UNIT – I
PART – A

1. What is J2EE?
The mission of J2EE is to provide a platform-independent, portable, multiuser, secure and standard enterprise class platform for server-side deployments written in the Java language.

2. What are the three different types Java Platform Editions?
Java Platform, Standard Edition (Java SE) - Develop secure, portable, high-performance applications for the widest range of computing platforms possible while boosting end-user productivity and dramatically reducing the cost of ownership of applications.
Java Platform, Enterprise Edition (Java EE) - Java Platform, Enterprise Edition (Java EE) is the standard in community-driven enterprise software. Java EE is developed using the Java Community Process, with contributions from industry experts, commercial and open source organizations, Java User Groups, and countless individuals. Each release integrates new features that align with industry needs, improves application portability, and increases developer productivity.
Java Platform, Micro Edition (Java ME) - Java Platform, Micro Edition (Java ME) provides a robust, flexible environment for applications running on mobile and embedded devices: mobile phones, set-top boxes, Blu-ray Disc players, digital media devices, M2M modules, printers and mor**e.**

3. Define Enterprise applications?
An enterprise application (EA) is a large software system platform designed to operate in a corporate environment such as business or government. EAs are complex, scalable, component-based, distributed and mission critical. EA software consists of a group of programs with shared business applications and organizational modeling utilities designed for unparalleled functionalities. EAs are developed using enterprise architecture.

4. What is a session bean?
A session bean is a relatively short-lived component. It has roughly the lifetime equivalent of a session or lifetime of the client code that is calling the session bean

5. What is a stateful session bean?
A stateful session bean is a bean that is designed to service business processes that span multiple method requests or transactions. To accomplish this, stateful session beans retain state on behalf of an individual client.

6. What is a stateless session bean?
A stateless session bean is a bean that holds conversations that span a single method call. They are stateless because they do not hold multihomed conversations with their clients.

7. What is the difference between authentication and authorization?
Authentication verifies the identity of a user and authorization is a process where you can check whether or not the identity has access rights to the system. In other words, you can say that authentication is a procedure of getting some credentials from the users and verify the user's identity against those credentials. Authorization is a procedure of granting access of particular resources to an authenticated user. You should note that authentication always takes place before authorization.

8. What is MVC?
Model-View-Controller architecture is used for interactive web-applications. This model minimizes the coupling between business logic and data presentation to web user. This model divides the web based application into three layers:

1. Model: Model domain contains the business logics and functions that manipulate the business data. It provides updated information to view domain and also gives response to query. And the controller can access the functionality which is encapsulated in the model.
2. View: View is responsible for presentation aspect of application according to the model data and also responsible to forward query response to the controller.
3. Controller: Controller accepts and intercepts user requests and controls the business objects to fulfill these requests. An application has one controller for related functionality. Controller can also be depends on the type of clients.

9. What are the containers available in J2EE architecture?
A web container for hosting Java Servlets and JSP pages
A EJB container for hosting enterprise java bean components
An applet container for hosting java applets
An application client container for hosting standard java applications

10. What are the services of J2EE container?
Declarative services, Services that the container interposes on the application, based on the deployment description provided for each application component, such as security, transaction etc.
Other container services, related to component lifecycle, resource pooling, garbage collection etc.,

11. What are technologies for developing components in J2EE platform?
Web component – which provides services to JSPs and servlets
EJB component - which provides services to EJBs. Enterprise JavaBeans (EJBs) reside in the business tier and are typically responsible for implementing the business logic of J2EE applications
XML – It influences how to view, process, transport and manage data.

12. Define Directory services?
A Directory services is a special type of database that is optimized for read access by using various indexing, and disk access techniques. The information in a directory service is described using a hierarchical information model.

13. What is LDAP?
LDAP stands for Lightweight Directory Access Protocol. It defines how client should access data on the server. It does not, however, specify how the data should be stored on the server.

14. What JNDI service providers?
A service provider is a set of Java classes that enable you to communicate with a directory service similar to the way in which a JDBC driver enables you to communicate to a database.

15. What is white page service?
White page services are services that enable someone to look up users based on attributes contained in their entries. For example, loop up Mark Wilcox's email address, and obtain the telephone number of the engineering office, the building number of Human resources, etc., They are called white pagesbecause this type of information is similar to the type of information find in the white pages of US telephone directory.

16. What are the 3 types of LDAP security?
Secure Socket Layer, Transport Layer Security and Simple Authentication and Security Layer

17. What is deployment descriptor?
A deployment descriptor defines the contract between the container and component. As application developers, we are required to specify a deployment descriptor for each group of application component.

18. What is naming service?
A naming service is a service that enables the creation of a standard name for a given set of data.

19. What is DIT?
Data in LDAP is organized in a hierarchical tree is called a Directory Information Tree(DIT). Each leaf in the DIT is called an entry in a DIT is called the root entry.

20. What are servlet engines and application servers?
These two the most common implementation tool for J2EE. Servlet engine or a servlet-container supports only the servlet API (including JSP, JSTL).
An application server supports the whole JavaEE - EJB, JMS, CDI, JTA, the servlet API (including JSP, JSTL), etc.

21. What is the difference between web client and EJB client?
Web client normally run in web browser. For these clients, the user interface is generated on the serverside as HTML or XML. And it is downloaded and rendered by the browsers.
EJB clients are applications that access EJB components in EJB containers.

22. Why LDAP is called light weight?
LDAP (Lightweight Directory Access Protocol) is a protocol for communications between LDAP servers and LDAP clients.
LDAP servers store "directories" which are access by LDAP clients.
LDAP is called lightweight because it is a smaller and easier protocol which was derived from the X.500 DAP

UNIT – II
Part-A

1. Define Struts?
Struts is open source software used to develop java based web page. Struts uses Jakarta Packages, Java Servlets, JavaBeans, ResourceBundles, and XML Struts takes the help of Model View Controller (MVC) architecture. Where Model is referring to business or database, View is referring to the Page Design Code, and Controller is referring to navigational code.

2. What is Action Class?
An Action class in the struts application is used to handle the request.

- It acts as interface or communication medium between the HTTP request coming to it and business logic used to develop the application.
- Action class consists of RequestProcessor which act as controller. This controller will choose the best action for each incoming request, generate the instance of that action and execute that action.
- This should be in thread-safe manner, because RequestProcessor uses the same instance for no. of requests at same time.

3. What is Struts Validator Framework?
Struts Validator Framework enables us to validate the data of both client side and server side. When some data validation is not present in the Validator framework, then programmer can generate own validation logic, this User Defined Validation logic can be bind with Validation Framework.Validation Framework consist of two XML configuration Files:Validator-Rules.xml file, Validation.xml file

4. What is the need of Struts?
We need Struts in Java because of following reasons:

- Helps in creation and maintenance of the application.
- Make use of Model View Controller (MVC) architecture. Where Model is referring to business or database, View is referring to the Page Design Code, and Controller is referring to navigational code.
- Enables developer to make use of Jakarta Packages, Java Servlets, JavaBeans, ResourceBundles, and XML

5. What are the classes used in Struts?
Struts Framework consists of following classes:

- **Action Servlets:** used to control the response for each incoming request.
- **Action Class:** used to handle the request.
- **Action Form:** it is java bean, used to referred to forms and associated with action mapping
- **Action Mapping:** used for mapping between object and action.
- **Action Forward:** used to forward the result from controller to destination.

6. What design patterns are used in Struts?
There are following types of design patterns are used in Struts: Service to Worker, Dispatcher View, Composite View (Struts Tiles), Front Controller, View Helper, Synchronizer Token

7. What are the different actions available in Struts?
The different kinds of actions in Struts are: ForwardAction, IncludeAction, DispatchAction, LookupDispatchAction, SwitchAction

8. What is the difference between ForwardAction and IncludeAction?
The difference between ForwardAction and InculdeAction are:IncludeAction is used when any other action is going to intake that action whereas ForwardAction is used move the request from one resource to another resource.

9. How is the MVC design pattern used in Struts framework?
In the MVC design pattern, application flow is mediated by a central Controller. The Controller delegates requests to an appropriate handler. The handlers are tied to a Model, and each handler acts as an adapter between the request and the Model. The Model represents, or encapsulates, an application's business logic or state. Control is usually then forwarded back through the Controller to the appropriate View. The forwarding can be determined by consulting a set of mappings, usually loaded from a database or configuration file. This provides a loose coupling between the View and Model, which can make an application significantly easier to create and maintain.

10. Describe validate() and reset() methods ?
validate() : Used to validate properties after they have been populated; Called before FormBean is handed to Action. Returns a collection of ActionError as ActionErrors. Following is the method signature for the validate() method.

11. What are the steps of Struts Installation?
In order to use Struts framework, we only need to add Struts.Jar file in our development environment. Once jar file is available in the CLASSPATH, we can use the framework and develop Strut based applications.

12. What is Hibernate?
Hibernate is a pure Java object-relational mapping (ORM) and persistence framework that allows you to map plain old Java objects to relational database tables using (XML) configuration files.Its purpose is to relieve the developer from a significant amount of relational data persistence-related programming tasks.

13. What are the Core interfaces are of Hibernate framework?
The five core interfaces are used in just about every Hibernate application. Using these interfaces, you can store and retrieve persistent objects and control transactions.**Session interface, SessionFactory interface, Configuration interface, Transaction interface, Query and Criteria interfaces**

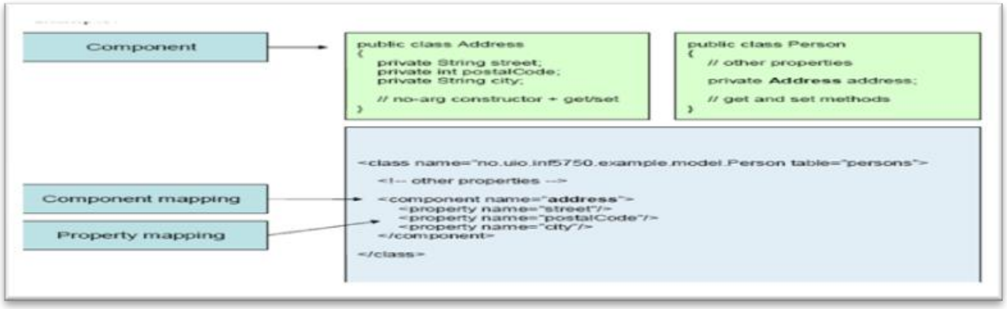
14. What is Hibernate Query Language (HQL)?
Hibernate offers a query language that embodies a very powerful and flexible mechanism to query, store, update, and retrieve objects from a database. This language, the Hibernate query Language (HQL), is an object-oriented extension to SQL.

15. What is the difference between load() and get()?

load()	**get()**
Only use the load() method if you are sure that the object exists.	If you are not sure that the object exists, then use one of the get() methods.
load() method will throw an exception if the unique id is not found in the database.	get() method will return null if the unique id is not found in the database.
load() just returns a proxy by default and database wont be hit until the proxy is first invoked.	get() will hit the database immediately.

16. Define cascade and inverse option in one-many mapping?
cascade - enable operations to cascade to child entities.
cascade="all|none|save-update|delete|all-delete-orphan"
inverse - mark this collection as the "inverse" end of a bidirectional association.
inverse="true|false"

17.What is component mapping in Hibernate?
- A component is an object saved as a value, not as a reference.
- A component can be saved directly without needing to declare interfaces or identifier properties
- Required to define an empty constructor
- Shared references not supported



```
public class Address
{
    private String street;
    private int postalCode;
    private String city;

    // no-arg constructor + get/set
}
```

```
public class Person
{
    // other properties

    private Address address;

    // get and set methods
}
```

```
<class name="no.uio.inf5750.example.model.Person table="persons">

  <!-- other properties -->

  <component name="address">
    <property name="street"/>
    <property name="postalCode"/>
    <property name="city"/>
  </component>

</class>
```

18. Define HibernateTemplate?
org.springframework.orm.hibernate.HibernateTemplate is a helper class which provides different methods for querying/retrieving data from the database. It also converts checked HibernateExceptions into unchecked DataAccessExceptions.

19. What are the Collection types in Hibernate?
The collection types in Hibernate are Bag, Set, List, Array, Map

20. What do you mean by fetching strategy?
A fetching strategy is the strategy Hibernate will use for retrieving associated objects if the application needs to navigate the association. Fetch strategies may be declared in the O/R mapping metadata, or over-ridden by a particular HQL or Criteria query.

UNIT – III
Part -A

1. What is LAMP?
***L** inux, **A** pache, **M** ySQL and **P** HP* , an open-source Web development platform that uses Linux as the operating system , Apache as the Web server , MySQL as the RDBMS and PHP as the object-oriented scripting language. Perl or Python is often substituted for PHP. LAMP has become a de facto development standard for Open Source Web Developers And Php developers. The combination of these technologies is used primarily to define a web server infrastructure, define a programming paradigm of developing software, and establish a software distribution package.

2. What are the tools for LAMP development?
- Linux
- Apache web server
- MySQL database application
- PHP scripting language

3. What are the benefits of LAMP stack?
Web application development using the LAMP stack is the preferred option for developers for creating a stable, reliable and highly efficient application.
- **Easy to code**: Ask all developers and they will tell you that coding is a breeze on LAMP. What this is does is that it ensures that coding is relatively bug free and doesn't have to go through an exhaustive and time consuming process of fixing the bugs.
- **Easy deployment**: For many developers, it's the deployment of a web application that becomes a tricky exercise; especially when the programming language cannot be easily integrated with the server and database. But, there are no such problems with LAMP as PHP is a standard Apache module. This makes it easier to deploy LAMP web applications.
- **Local Development** – Another huge advantage of using LAMP is that a developer can build an app locally and then deployed it onto the web. LAMP is definitely popular, but not all developers are able to optimize its use. You need to be able to choose the kind of developer who is both an expert and experienced, on using the LAMP stack.

4. What is PHP ?
PHP (recursive acronym for *PHP: Hypertext Preprocessor*) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

5. What is the difference between == and === in PHP?
When comparing values in PHP for equality you can use either the == operator or the === operator. What's the difference between the 2? Well, it's quite simple. The == operator just checks to see if the left and right values are equal. But, the === operator (note the extra "=") actually checks to see if the left and right values are equal, and also checks to see if they are of the same variable type (like whether they are both booleans, ints, etc.).

6. How would you return an array from a function in PHP?
If you have a function and you want to return multiple values from that function then you can easily return an array from the function. This is what it would look like:
Example:
```
Function someFunc() {
$aVariable = 10;
$aVariable2 = 10;
Return array($aVariable, $aVariable2
}
```
7. What is perl?
Perl, the Practical Extraction and Report Language, was the creation of linguist and programmer Larry Wall. Perl is a script programming language that is similar in syntax to the C language and that includes a number of popular UNIX facilities such as sed, awk, and tr. Perl is an interpreted language that can optionally be compiled just before execution into either C code or cross-platform byte code. When compiled, a Perl program is almost (but not quite) as fast as a fully precompiled C language program. Perl is regarded as a good choice for developing common gateway interface (CGI) programs because it has good text manipulation facilities (although it also handles binary files).

8. Why is LAMP a popular choice?
Think of a scenario wherein your business is unable to manage its organizational data. You want a solution to the comprehensive data flow taking place throughout your organization and you want this solution in double quick time. In cases like these, where a business or an organization cannot dedicate a lot of time to problem solving, the LAMP stack is the preferred platform for development. This is because developers can build an application quickly and ensure its reliability and stability. It's actually a win-win situation for both you and the developers. Both sides save time, and the well-defined development process ensures a highly efficient application.

9. What is MySQL?
MySQL is an SQL based relational database management system (DBMS) that runs under a broad array of operating systems. MySQL is frequently used by PHP and Perl scripts. MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).
SQL is the most popular language for adding, accessing and managing content in a database. It is most noted for its quick processing, proven reliability, ease and flexibility of use. MySQL is an essential part of almost every open source PHP application.

10. What are the features of MySQL?
- Cross-platform support
- Stored procedures
- A set of SQL Mode options to control runtime behavior, including a strict mode to better adhere to SQL standards.
- X/Open XA distributed transaction processing (DTP) support; two phase commit

11. What is Phython?
Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

12. How to define functions in phython?
The keyword def introduces a function *definition*. It must be followed by the function name and the parenthesized list of formal parameters. The statements that form the body of the function start at the next line, and must be indented.
Example: >>> def fib(n)

13. What is Lamda expressions?
Small anonymous functions can be created with the lambda keyword. This function returns the sum of its two arguments: lambda a, b: a+b. Lambda functions can be used wherever function objects are required. They are syntactically restricted to a single expression. Semantically, they are just syntactic sugar for a normal function definition. Like nested function definitions, lambda functions can reference variables from the containing scope:
Example:
```
>>> def make_incrementor(n):
return lambda x: x + n
```

14. What is pickling and unpickling.
pickle is a standard module which serializes & de-serializes a python object structure.
pickle module accepts any python object converts it into a string representation & dumps it into a file(by using dump() function) which can be used later, process is called **pickling**. Whereas **unpickling** is process of retrieving original python object from the stored string representation for use.

15. Explain how python is interpreted.
Python program runs directly from the source code. Each type Python programs are executed code is required. Python converts source code written by the programmer into intermediate language which is again translated it into the native language / machine language that is executed. So Python is an Interpreted language.

16. State some programming language features of Python.
Simple & Easy: Python is simple language & easy to learn.
Free/open source: it means everybody can use python without purchasing license.
High level language: when coding in Python one need not worry about low-level details.
Portable: Python codes are Machine & platform independent.
Extensible: Python program supports usage of C/ C++ codes.
Embeddable Language: Python code can be embedded within C/C++ codes & can be used a scripting language.
Standard Library: Python standard library contains prewritten tools for programming.
Build-in Data Structure: contains lots of data structure like lists, numbers & dictionaries.

17. How is memory managed in Python?
Like other programming language python also has garbage collector which will take care of memory management in python.

18. Why are Python strings immutable?
One is performance: knowing that a string is immutable means we can allocate space for it at creation time, and the storage requirements are fixed and unchanging. This is also one of the reasons for the distinction between tuples and lists.
Another advantage is that strings in Python are considered as "elemental" as numbers. No amount of activity will change the value 8 to anything else, and in Python, no amount of activity will change the string "eight" to anything else.

19. Is there a scanf() or sscanf() equivalent?
Not as such. For simple input parsing, the easiest approach is usually to split the line into whitespace-delimited words using the split() method of string objects and then convert decimal strings to numeric values using int() or float(). split() supports an optional "sep" parameter which is useful if the line uses something other than whitespace as a separator.For more complicated input parsing, regular expressions more powerful than C's sscanf() and better suited for the task.

20. What is Python good for?
Python is a high-level general-purpose programming language that can be applied to many different classes of problems.
The language comes with a large standard library that covers areas such as string processing (regular expressions, Unicode, calculating differences between files), Internet protocols (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, CGI programming), software engineering (unit testing, logging, profiling, parsing Python code), and operating system interfaces (system calls, filesystems, TCP/IP sockets).

UNIT – IV
Part - A

1. What is C#?
C# is a computer-programming language developed by Microsoft Corporation. C# is a fully object-oriented language like java and it is the first component –oriented language. It has been designed to support the key features of .NET Frame work’s# is used to build robust, reliable, and durable components for real-world applications.

2. C# is a modern language. Comment.
C# supports the following properties: Automatic garbage collection, Rich intrinsic model for error handling, Decimal data type for financial applications, Modern approach to debugging , Robust security model for the purpose of protection.

3. Why is C# called type – safe language ?
C# is called a type-safe language because of the following reasons :
(i)All dynamically allocated objects and arrays are initialized to zero (ii)Use of ant uninitialized variables produces an error message by the compiler (iii)Access to arrays are range-checked and warned (iv)C# does not permit unsafe casts (v)Reference parameters that are type safe

4. What is .NET framework ?
The .NET framework provides an environment for building, deploying and running web services and other applications. It consists of the following three technologies: (i) Common Language Run time(CLR) (ii) Framework Base Classes (iii) User and Program Interfaces (containing ASP.NET and Winforms) techniques.

5. What are the services provided by the CLR ?
The services provided by the CLR Loading and execution of programs, Memory isolation for applications, Verification of type safety, Memory management (automatic garbage collection), Interoperability with other systems, Managing exceptions and errors

6. What is managed code?
The CLR is responsible for managing the execution of various code compiled for the .NET platform. The code that satisfies the CLR at runtime in order to execute is referred to as managed code. The managed code generated by the c# compiler is the IL code.

7. Why do we use the using directive in a C# program ?
The using directive can be used to import namespace 'System' into the program. Once a namespace is imported , we can use the elements of that namespace using the namespace as prefix. Eg . *using System* ; // System is a namespace

8. List the type of literals supported by C#.
The literals supported by C# are Numeric literals, Boolean Literals, Character Literals. Numeric Literals contains two types Integer Literals and Real Literals. Character Literals contains two types Single Character Literals and String Literals.

9. What is the need for operator overloading?
Operators can be defined to work with the user defined data types such as structs and classes in much the same way as the built-in types. Operator overloading helps us to generate more readable and intuitive code in a number of situations such as mathematical modeling, graphical programs, etc. These are the functionalities of operator overloading.

10. What are overloaded constructors?
Similar to method overloading we have overloaded constructor wherein we have more than one constructors having same name as that of the class. These constructors have different definition, with different parameter list. The difference may be in either the number or type of arguments. Each parameter list must be unique.

11. What are the restrictions the static method has?
The restrictions that the static method has are (i) They can only call other static methods.(ii)They can only access static data and (iii) They cannot refer to this or base in any way. These are the restrictions of a static method or static function in C#.

12. What is a copy constructor?
A copy constructor creates an object by copying variable and its values from another object.

```
Eg: public Item (Item item)
    {
    code=item.code;
    price=item.price;
    }
```

The copy constructor is invoked by instantiating an object of type Item and passing it the object to be copied. Eg: *Item item2=new Item(item1);* now item2 is a copy of item1.

13. Distinguish between zero place holder and pound (#) space holder in producing integer outputs.
In using zero placeholders, if the number has a digit in the position at which the 0 appears in the format string, the digit will appear in the output; otherwise zero will appear. The pound (#) space holder works similar to the zero placeholder, except that a blank (instead of zero) appears if there is no digit in that position

14. Distinguish between the Write and Write Line methods
Write( ) method outputs one or more values to the screen without a newline character. WriteLine( ) method outputs one or more values to the screen but adds a newline character at the end of the output. These are the output methods used by the programmer to print values on screen.

15. What is the use of checked and unchecked operators?
To check for exceptions like Stack overflow, if the operation is checked the exception will be thrown. If the operation is not checked then no exception will be raised and we will lose data. *checked* and *unchecked*are the operators used for this special purpose.

16. What is an assembly?
An assembly is a collection of files thatappears to the user to be a single dynamic link library (DLL) or executable (EXE). DLLs are collections of classes and methods that are linked into your running program only when they are needed. Assemblies are the .NET unit of reuse, versioning, security, and deployment.

17. What do you mean by versioning?
Each assembly has a version number, and versions cannot transcend the boundary of the assembly. That is, a version can refer only to the contents of a single assembly. All types andresources within the assembly change versions together.

18. What is a single-module assembly?
A single-module assembly has a single file that can be an EXE or DLL file. This single module contains all the types and implementations for the application. The assembly manifest is embedded within this module.

19. Define multi-module assembly.
A multi-module assembly consists of multiple files (zero or one EXE and zero or more DLL files, though you must have at least one EXE or DLL). The assembly manifest in this case can reside in a standalone file, or it can be embedded in one of the modules. When the assembly is referenced, the runtime loads the file containing the manifest and then loads the required modules as needed.

20. How will you specify version number for an assembly?
A version number for an assembly might look like this: 1:0:2204:21 (four numbers, separated by colons). The first two numbers (1:0) are the major and minor version. The third number (2204) is the build, and the fourth (21) is the revision.

UNIT – V
Part - A

1. What is ASP?
Active Server Pages (ASP), also known as Classic ASP, is a Microsoft's server-side technology, which helps in creating dynamic and user-friendly Web pages. It uses different scripting languages to create dynamic Web pages, which can be run on any type of browser. The Web pages are built by using either VBScript or JavaScript and these Web pages have access to the same services as Windows application, including ADO (ActiveX Data Objects) for database access, SMTP (Simple Mail Transfer Protocol) for e-mail, and the entire COM (Component Object Model) structure used in the Windows environment. ASP is implemented through a dynamic-link library (asp.dll) that is called by the IIS server when a Web page is requested from the server.

2. What is ASP.NET?
ASP.NET is a specification developed by Microsoft to create dynamic Web applications, Web sites, and Web services. It is a part of .NET Framework. You can create ASP.NET applications in most of the .NET compatible languages, such as Visual Basic, C#, and J#. The ASP.NET compiles the Web pages and provides much better performance than scripting languages, such as VBScript. The Web Forms support to create powerful forms-based Web pages. You can use ASP.NET Web server controls to create interactive Web applications. With the help of Web server controls, you can easily create a Web application.

3. What is the basic difference between ASP and ASP.NET?
The basic difference between ASP and ASP.NET is that ASP is interpreted; whereas, ASP.NET is compiled. This implies that since ASP uses VBScript; therefore, when an ASP page is executed, it is interpreted. On the other hand, ASP.NET uses .NET languages, such as C# and VB.NET, which are compiled to Microsoft Intermediate Language (MSIL).

4. What is an ASP.NET Web Form?
ASP.NET Web forms are designed to use controls and features that are almost as powerful as the ones used with Windows forms, and so they are called as Web forms. The Web form uses a server-side object model that allows you to create functional controls, which are executed on the server and are rendered as HTML on the client. The attribute, runat="server", associated with a server control indicates that the Web form must be processed on the server.

5. What is IIS? Why is it used?
Internet Information Services (IIS) is created by Microsoft to provide Internet-based services to ASP.NET Web applications. It makes your computer to work as a Web server and provides the functionality to develop and deploy Web applications on the server. IIS handles the request and response cycle on the Web server. It also offers the services of SMTP and FrontPage server extensions. The SMTP is used to send emails and use FrontPage server extensions to get the dynamic features of IIS, such as form handler.

6. Advantage of ASP.net?
ASP.Net is the next generation of Microsoft's Active Server Page (ASP) technology platform.
ASP.Net is superior to ASP in the following ways:
Compiled Code
Language Support
Strict Coding Requirements
Event-Driven Programming Model
Third-Party Controls
User Authentication
Easier Configuration & Deployment
Object and Page Caching
Higher Scalability

7. What are different methods of session maintenance in ASP.NET?
In-Process Storage
The default location for session state storage is in the ASP.NET process itself.
Session State Service
As an alternative to using in-process storage for session state, ASP.NET provides the ASP.NET State Service. The State Service gives you an out-of-process alternative for storing session state that is not tied quite so closely to ASP. Net's own process.
Microsoft SQL Server
The final choice for storing state information is to save it in a Microsoft SQL Server database.
8. What is ViewState?
The ViewState is a feature used by ASP.NET Web page to store the value of a page and its controls just before posting the page. Once the page is posted, the first task by the page processing is to restore the ViewState to get the values of the controls.

9. Why use Silverlight?
- Support for the .NET Framework – if you are already a .NET developer, it is easy to start programming on Silverlight.
- Support for managed code – you can write programs in your favorite language which .NET CLR supports like C#, VB.NET, dynamic languages (IronPython, IronRuby).

- Better development tools -Visual Studio 2010, Expression Blend.
- Large community- More learning resources available compared to Flash.
- Integration with Enterprise based technologies like WPF, LINQ etc…
- Silverlight integrates the XAML declarative language with the .NET framework.
- It is a cross-browser, cross-platform technology which provides a consistent user experience everywhere it runs.

10. Which browsers does Silverlight support?

- Microsoft - Internet Explorer 6, 7, 8
- Mozilla - Firefox 2 and 3
- Apple - Safari 3 and 4
- Google - Chrome

11. What are the main features and benefits of Silverlight?

- Compelling cross-platform user experiences.
- Flexible programming model with collaboration tools.
- High-quality media, low-cost delivery.
- Connected to data, servers, and services.

12. What are the tools required to develop Silverlight applications?

Microsoft Expression Studio - This tool is meant for web designers to create rich visual elements for Silverlight applications. Expression Studio is recommended for web designers who create rich internet applications with enhanced visual content and graphics. There are several features provided for creating enhanced graphics elements, with lot of options to pick color, font, etc.

Microsoft Visual Studio - This is the integrated development environment from Microsoft to develop .NET applications. Programmers can use Visual Studio to develop Silverlight applications which require programming. Visual Studio allows programmers to develop sophisticated Silverlight applications in any .NET language (like C#, VB.NET etc).

13. When to use Silverlight, ASP.NET, or both?

Do you really need a rich interactive user experience?, Can you add Silverlight islands to your existing ASP.NET application instead of mixing the contents?, Will this be a good chance to revise your architecture?

14. What are the main components of Silverlight application?

Following are the main component of Silverlight application.
a)Input – It handles input from devices like keyboard, mouse etc.
b)UI core –It manages rendering of bitmap images, vector graphics, text and animations.
c)Media – It handles the playback of MP3, WMA Standard, WMV7, WMV8 streams.
d)XAML – This manages UI layout to be created by XAML markup language

15. What is Easing Functions in Silverlight?

Easing functions is used in Silverlight to utilise custom mathematical formulas to animations. For example,if we want an object to oscillate we could use we can use a corresponding mathmetical function to accurately depict this motion.

16. What is XAML ?

Extensible Application Markup Language (XAML, pronounced zammel) is a declarative XML-based language created by Microsoft which is used to initialize structured values and objects.

17. What is the difference between WPF and Silverlight?
Silverlight uses a particular implementation of a XAML parser, with that parser being part of the Silverlight core install. In some cases, the parsing behavior differs from the parsing behavior in Windows Presentation Foundation (WPF), which also has a particular implementation.

18. What methods are fired during the page load?
Init() - when the page is instantiated,
Load() - when the page is loaded into server memory,
PreRender() - the brief moment before the page is displayed to the user as HTML,
Unload() - when page finishes loading.

19. What is the difference between Server.Transfer and Response.Redirect?
In **Server.Transfer** page processing transfers from one page to the other page without making a round-trip back to the client's browser. This provides a faster response with a little less overhead on the server. The clients url history list or current url Server does not update in case of Server.Transfer. **Response.Redirect** is used to redirect the user's browser to another page or site. It performs trip back to the client where the client's browser is redirected to the new page. The user's browser history list is updated to reflect the new address.

20. What is MVC?
MVC is a framework used to create web applications. The web application base builds on Model-View-Controller pattern which separates the application logic from UI, and the input and events from the user will be controlled by the Controller.

UNIT – I
PART – B

1. List and explain the enterprise architecture styles

Business Architecture

Any architectural discussion should begin with Business Architecture. The Business Architecture aligns an organization's operating model, strategies, and objectives with IT; it also creates a business case for IT transformations and provides a business-centric view of the enterprise from a functional perspective. This part of the framework provides the following three key areas of information about the business:

• **Business Strategy:** Key business requirements, objectives, strategies, key performance indicators, business risks, and the business-operating model (how processes and systems are centralized versus decentralized across the business).

• **Business Function:** The key business services, processes, and capabilities that will be affected by the enterprise architecture effort.

• **Business Organization:** The high-level nature of the organizational structures, business roles (internal audiences, external customers and partners), the decision-making process, and the organizational budget information

Application Architecture

The Application Architecture provides an application- and services-centric view of an organization that ties business functions and services to application processes and services to application components in alignment with the application strategy. The Application Architecture's scope, strategy, standards are a consequence of the Business Architecture. The Application Architecture is composed of the following content categories:

• **Application Strategy:** The key application architecture principles (Build versus Buy, Hosted versus In-House, Open Standards versus .NET, etc.), application governance and portfolio management, and a set of reference application architectures relevant to the customer.

• Application Services: An inventory of the key application services exposed to internal and external audiences that support the business services.

• Application Processes: A series of application-specific processes that support the business processes in the Business Architecture.

• Logical Components: An inventory of the relevant product-agnostic enterprise application systems that is relevant to the stated business objectives.

• **Physical Components:** The actual products that support the logical application components and their relationships to the relevant components and services in the information and technology architectures.

Information Architecture

The Information Architecture describes all of the moving pieces and parts for managing information across the enterprise, and the sharing of that information to the right people at the right time to realize the business objectives stated in the business architecture. The key components for describing the information architecture are:

• **Information Strategy:** The information architecture principles, information governance and compliance requirements, canonical data models, and industry data model support strategy and a set of reference information exchange as well as dissemination patterns and reference models.

• **Information Assets:** A catalog of critical business data types and models (such as customer profile, purchase order, product data, supply chain, etc.) and the relationships between those business data types and all the services and processes that interact with that data. The Information Architecture provides an information- and data-centric view of an organization, focusing on key information assets that are used to support critical business functions.

Technology Architecture

The Technology Architecture describes how the infrastructure underlying the business, application, and information architectures is organized. The key components are:

• **Technology Strategy:** The technology architecture principles, technology asset governance and portfolio management strategy, and technology standards, patterns, and reference architectures used for developing specific technology solutions.
• **Technology Services:** An inventory of the specific technology services and their relationships and the business services, application services, information assets and logical or physical technology components that realize those services.
• **Logical Components:** The product-agnostic components that exist at the technology infrastructure tier to support each technology service.
• **Physical Components:** The set of technology products that exists behind each of the logical technology components to implement the technology service.

2. **Explain about the J2EE container architecture**

The component-based and platform-independent J2EE architecture makes J2EE applications easy to write because business logic is organized into reusable components and the J2EE server provides underlying services in the form of a container for every component type. Because you do not have to develop these services yourself, you are free to concentrate on solving the business problem at hand.

Containers and Services

Component are installed in their containers during deployment and are the interface between a component and the low-level platform-specific functionality that supports the component. Before a web, enterprise bean, or application client component can be executed, it must be assembled into a J2EE application and deployed into its container.

The assembly process involves specifying container settings for each component in the J2EE application and for the J2EE application itself. Container settings customize the underlying support provided by the J2EE Server, which include services such as security, transaction management, Java Naming and Directory Interface (JNDI) lookups, and remote connectivity. Here are some of the highlights:

- The J2EE security model lets you configure a web component or enterprise bean so system resources are accessed only by authorized users.
- The J2EE transaction model lets you specify relationships among methods that make up a single transaction so all methods in one transaction are treated as a single unit.
- JNDI lookup services provide a unified interface to multiple naming and directory services in the enterprise so application components can access naming and directory services.
- The J2EE remote connectivity model manages low-level communications between clients and enterprise beans. After an enterprise bean is created, a client invokes methods on it as if it were in the same virtual machine.

The fact that the J2EE architecture provides configurable services means that application components within the same J2EE application can behave differently based on where they are deployed. For example, an enterprise bean can have security settings that allow it a certain level of access to database data in one production environment and another level of database access in another production environment.

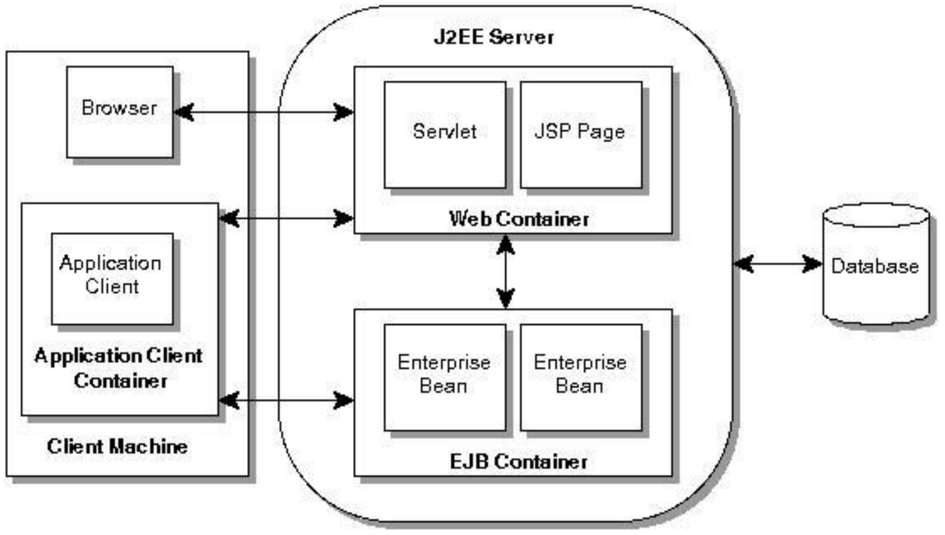
The container also manages non-configurable services such as enterprise bean and servlet life cycles, database connection resource pooling, data persistence, and access to the J2EE platform APIs described in J2EE APIs. Although data persistence is a non-configurable service, the J2EE architecture lets you override container-managed persistence by including the appropriate code in your enterprise bean implementation when you want more control than the default container-managed persistence provides. For example, you might use bean-managed persistence to implement your own finder (search) methods or to create a customized database cache.

Container Types

The deployment process installs J2EE application components in the following types of J2EE containers.

- An Enterprise JavaBeans (EJB) container manages the execution of all enterprise beans for one J2EE application. Enterprise beans and their container run on the J2EE server.
- A web container manages the execution of all JSP page and servlet components for one J2EE application. Web components and their container run on the J2EE server.

- An application client container manages the execution of all application client components for one J2EE application. Application clients and their container run on the client machine.
- An applet container is the web browser and Java Plug-in combination running on the client machine.



3. Explain about LDAP operations with relevant example

LDAP defines operations for accessing and modifying directory entries such as:
Binding and unbinding
Searching for entries meeting user-specified criteria
Adding an entry, Deleting an entry and Modifying an entry
Modifying the distinguished name or relative distinguished name of an entry (move) Comparing an entry
LDAP operations can be divided into the following three categories:
Query: Includes the search and compare operations used to retrieve information from a irectory
Update - Includes the add, delete, modify, and modify RDN operations used to update stored information in a directory
Authentication: Includes the bind, unbind, and abandon operations used to connect and disconnect to and from an LDAP server, establish access rights and protect information
Search: To perform a search, the following parameters must be specified
Base: A DN that defines the starting point, called the base object, of the search. The base object is a node within the DIT.
Scope: Specifies how deep within the DIT to search from the base object. There are three choices: baseObject, singleLevel, and wholeSubtree. If baseObject is specified, only the base object is examined. If singleLevel is specified only the immediate children of the base object are examined; the base object itself is not examined. If wholeSubtree is specified, the base object and all of its descendants are examined.
Search Filter: Specifies the criteria an entry must match to be returned from a search. The search filter is a Boolean combination of attribute value assertions
Limits: Searches can be very general, examining large subtrees and causing many entries to be returned. The user can specify time and size limits to prevent wayward searching from consuming too many resources
Some example searches expressed informally in English are:
• Find the postal address for cn=John Smith,o=IBM,c=DE.
• Find all the entries that are children of the entry ou=ITSO,o=IBM,c=US.
• Find the e-mail address and phone number of anyone in IBM whose last name contains the characters “miller” and who also has a fax number.

4. With example how will you store and retrieve the Java objects in LDAP

Storing Java objects in LDAP

There are numerous tools available for persisting Java objects to a SQL database, such as Hibernate. But what if you want to persist your objects to an LDAP directory, such as Fedora Directory Server? Perhaps the easiest way is to use the Java Naming and Directory Interface (JNDI). JNDI is an abstraction for working with all kinds of naming services, such as DNS, file systems, and LDAP.

However, most times where I see people working with JNDI and LDAP, they end up repeating a lot of work, or create some LDAPUtil class for performing the lookup and storing of objects. A lesser-known method that is built into JNDI is the use of object and state factories. For example, if I have a Java class calledGroupAccount that is used to store the names of group members, storing this object in an LDAP directory would consist of the following steps:

Create a directory context, using InitialDirContext

1. Create a BasicAttributes object
2. Create a BasicAttribute object for the object class, as well as any required attributes as defined in the LDAP schema
3. For each member of the group, add the value to the corresponding BasicAttribute
4. Add each BasicAttribute instance to the BasicAttributes object

Search the server for an entry

LDAP boasts feature-rich search facilities. The simplest level of search can use the JNDI naming service to perform the following types of searches:

- Context.lookup(), which returns a stored object corresponding to that DN in the server
- Context.list(), which returns the names bound and the class names of the objects bound in the context
- Context.listBindings(), which returns the names bound and the objects bound to them in the context

LDAP stores information in the tree and everything is specific to the node. Therefore, to search for an object in the LDAP server, you need to specify the node or the base from which to start -- the *scope* of the search. The scope defines exactly how much of the tree should be searched.

Different levels of scope

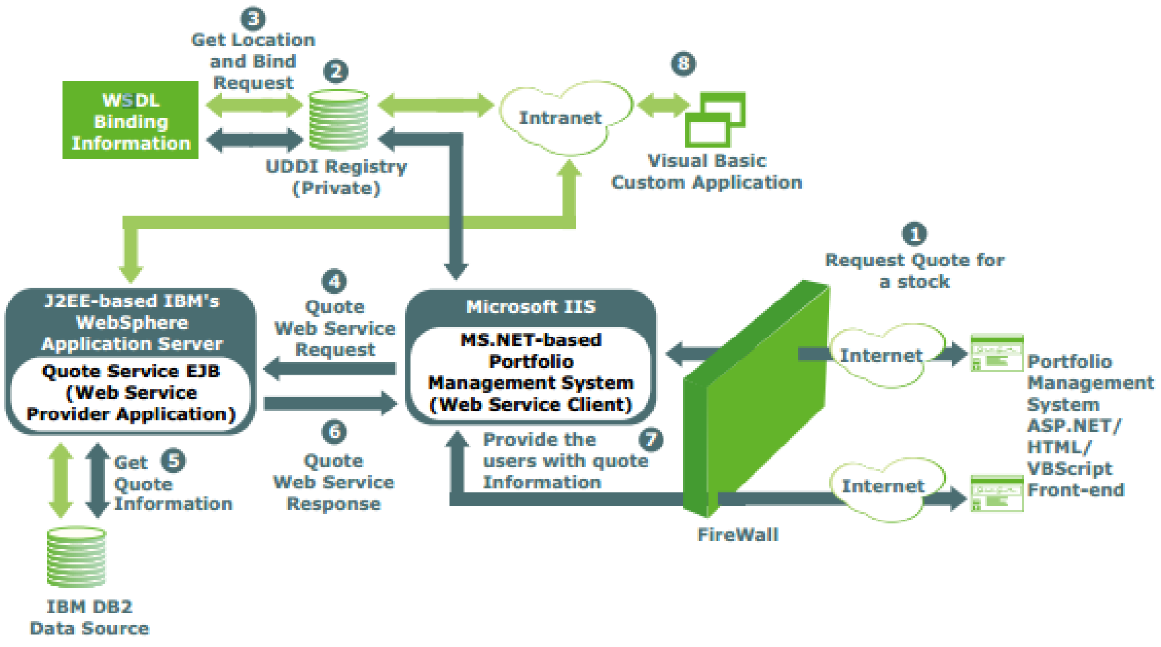
SUBTREE_SCOPE	Starts at the base entry; searches the base entry and everything below it
ONELEVEL_SCOPE	Searches only the entries below the base entry
OBJECT_SCOPE	Searches only the base entry; useful if you need to get attributes/value pair of just one entry

To perform the physical search, we invoke thesearch() method on the object that implements the DirContext interface (InitialDirContext). The search result is NamingEnumeration, an enumeration of SearchResultobjects. At minimum, you need the search base and the filter in order to perform a search, but other parameters can be specified to manage the results. The SearchContols object holds most additional parameters for the search, including scope, as seen below:

```
SearchControls ctls = new SearchControls();
setSearchScope(SearchControls.SUBTREE_SCOPE)
```

5. In detail explain about J2EE implementation functionality

The retail clients and in-house clients of a financial company use the portfolio management portal to monitor their investments. The front-end of the portal is built using Microsoft technology (Active Server Pages .NET (ASP.NET), Internet Information Services (IIS) Web Server, VB Script, etc.). One of the features provided within the portal application is quote information. Using this feature, the clients can retrieve real-time quotation for any stock. When a client requests a quote for any stock, the request is sent from the browser to the Web Server. As may potentially happen within any mid-to-large size company, we are assuming that the quote service is provided to multiple clients as a Web Service by a middleware application within the company, with our portfolio management portal being just one of those clients. Another client, as shown in the figure, is a VB application. The information about Web Services offered by this middleware application (which may be published by some other group within the company) is obtained from the private internal UDDI registry and invoked over the intranet. The implementation of the business methods exposed by the Web Service is provided by EJBs contained in another application server. This is a typical example of .NET-to-J2EE application server integration using Web Services. The binding information for frequently used Web Services, such as those for requesting quotes, can be cached by the client application, to avoid the resource intensive and time consuming dynamic binding. In this example, Web Services loosely integrates Microsoft technology-based portfolio management application with the J2EE-based middleware application that interfaces with the Mainframe to receive the quote.


3 Get Location and Bind Request
2
WSDL Binding Information
UDDI Registry (Private)
Intranet
8
Visual Basic Custom Application
1 Request Quote for a stock
J2EE-based IBM's WebSphere Application Server
Quote Service EJB (Web Service Provider Application)
4 Quote Web Service Request
Microsoft IIS
MS.NET-based Portfolio Management System (Web Service Client)
Internet
Portfolio Management System ASP.NET/ HTML/ VBScript Front-end
6 Quote Web Service Response
Get 5 Quote Information
Provide the users with quote Information 7
Internet
FireWall
IBM DB2 Data Source

The sequence of steps is as follows:

1. The user requests quotes for a specific company on an ASP.NET/VBScript/HTML front-end that is passed over to the portfolio management portal running within Microsoft IIS. Here for the sake of simplicity it is assumed that the user has already successfully logged into the application and has a valid session established.
2. The .NET-based portal application gets information about Web Services made available by the J2EE-based middleware application by performing a look up in the private UDDI registry.
3. The location of and WSDL binding information for Web Services is sent to the portal application as a SOAP-based message.
4. The portal application invokes the Web Service published by the middleware application, passing a stock symbol as part of a SOAP-based message.
5. The actual implementation of the Web Service is provided by EJBs running within a J2EE-based application server. The EJBs use the JDBC API to get information from the data source, which in this case is IBM's DB2.
6. The EJBs send the Web Services response to the portal application as a SOAP-based message.
7. The response is formatted in HTML (using XSLT) and sent back to the browser-based client application.
8. Another VB custom application within the company intranet invokes the same Web Service, thereby being another client of the quote Web Service. The communication happens based on SOAP

6. Explain the following

i. J2EE Packaging

J2EE components are packaged separately and bundled into a J2EE application for deployment. Each component, its related files such as GIF and HTML files or server-side utility classes, and a deployment descriptor (DD), are assembled into a module and added to the J2EE application. A J2EE application is composed of one or more enterprise bean, web, or application client component modules. The final enterprise solution can use one J2EE application or be made up of two or more J2EE applications depending on design requirements

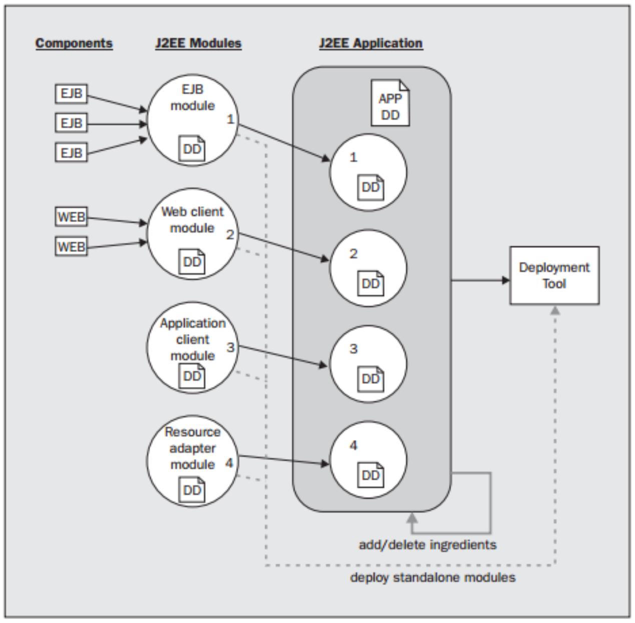
A J2EE application and each of its modules has its own deployment descriptor. A deployment descriptor is an Extensible Markup Language (XML) text-based file with an .xml extension that describes a component's deployment settings. An enterprise bean module deployment descriptor, for example, declares transaction attributes and security authorizations for an enterprise bean. Because deployment descriptor information is declarative, it can be changed without modifying the bean source code. At run time, the J2EE server reads the deployment descriptor and acts upon the component accordingly.

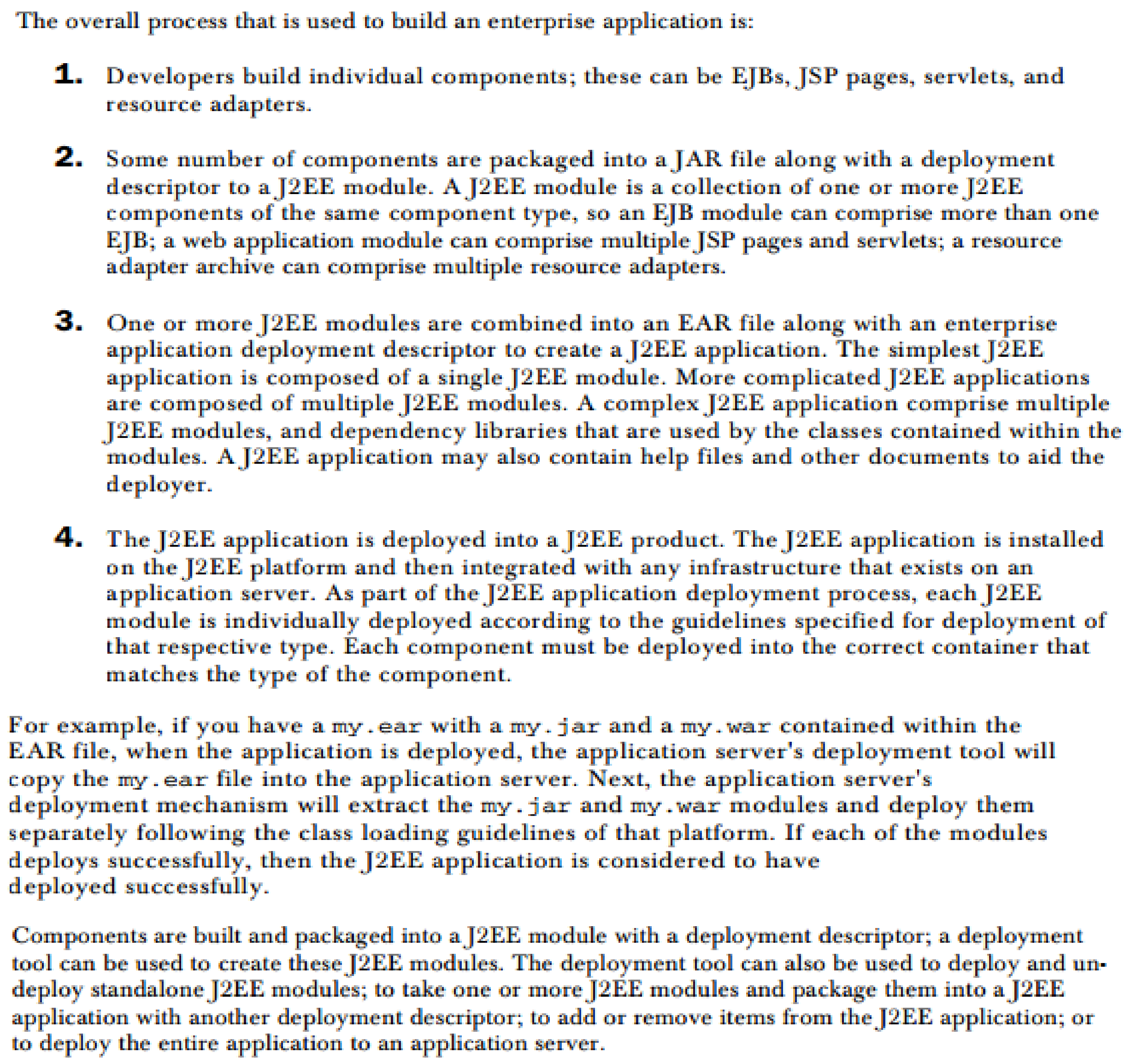
A J2EE application with all of its modules is delivered in an Enterprise ARchive (EAR) file. An EAR file is a standard JAR file with an .ear extension. In the GUI version of the J2EE SDK application deployment tool, you create an EAR file first and add JAR and WAR files to the EAR. If you use the command line packager tools, however, you create the Java ARchive (JARs) and Web ARchive (WAR) files first and create the EAR.
The J2EE SDK tools are described in Tools.

- Each EJB JAR file contains its deployment descriptor, related files, and the .class files for the enterprise bean.
- Each application client JAR file contains its deployment descriptor, related files, and the .class files for the application client.
- Each WAR file contains its deployment descriptor, related files, and the .class files for the servlet or .jsp files for a JSP page.

Using modules and EAR files makes it possible to assemble a number of different J2EE applications using some of the same components. No extra coding is needed; it is just a matter of assembling various J2EE modules into J2EE EAR files.

ii. J2EE Deployment





The overall process that is used to build an enterprise application is:

1. Developers build individual components; these can be EJBs, JSP pages, servlets, and resource adapters.

2. Some number of components are packaged into a JAR file along with a deployment descriptor to a J2EE module. A J2EE module is a collection of one or more J2EE components of the same component type, so an EJB module can comprise more than one EJB; a web application module can comprise multiple JSP pages and servlets; a resource adapter archive can comprise multiple resource adapters.

3. One or more J2EE modules are combined into an EAR file along with an enterprise application deployment descriptor to create a J2EE application. The simplest J2EE application is composed of a single J2EE module. More complicated J2EE applications are composed of multiple J2EE modules. A complex J2EE application comprise multiple J2EE modules, and dependency libraries that are used by the classes contained within the modules. A J2EE application may also contain help files and other documents to aid the deployer.

4. The J2EE application is deployed into a J2EE product. The J2EE application is installed on the J2EE platform and then integrated with any infrastructure that exists on an application server. As part of the J2EE application deployment process, each J2EE module is individually deployed according to the guidelines specified for deployment of that respective type. Each component must be deployed into the correct container that matches the type of the component.

For example, if you have a `my.ear` with a `my.jar` and a `my.war` contained within the EAR file, when the application is deployed, the application server's deployment tool will copy the `my.ear` file into the application server. Next, the application server's deployment mechanism will extract the `my.jar` and `my.war` modules and deploy them separately following the class loading guidelines of that platform. If each of the modules deploys successfully, then the J2EE application is considered to have deployed successfully.

Components are built and packaged into a J2EE module with a deployment descriptor; a deployment tool can be used to create these J2EE modules. The deployment tool can also be used to deploy and un-deploy standalone J2EE modules; to take one or more J2EE modules and package them into a J2EE application with another deployment descriptor; to add or remove items from the J2EE application; or to deploy the entire application to an application server.

7. Explain about J2EE technologies

Java 2 Platform, Standard Edition (J2SE) as a basis, Java 2 Platform, Enterprise Edition (J2EE) builds on top of this to provide the types of services that are necessary to build large scale, distributed, component based, multi-tier applications. Essentially, J2EE is a collection of APIs that can be used to build such systems, although this is only half of the picture. J2EE is also a standard for building and deploying enterprise applications, held together by the specifications of the APIs that it defines and the services that J2EE provides. In other words, this means that the "write once, run anywhere" promises of Java apply for enterprise applications too:

- Enterprise applications can be run on different platforms supporting the Java 2 platform.
- Enterprise applications are portable between application servers supporting the J2EE specification.
- **Servlet Technology**: It is considered as the foundation of web technologies. It overcomes the limitations of CGI technology. It is a server side component to serve the clients and to generate dynamic content. Servlets interacts with web clients using the paradigm ‘request-response’. All client requests are sent through web servers to the servlet container. The servelt then process the request and sends the response back to the client. Servlets are reliable, scalable, efficient and reusable server side components.
- **JSP Technology**: It is the extension of servlet technology. It is easy to author JSP without much knowledge of the supporting API. JSP can be used to work with HTTP requests and HTTP responses,

session management and so on. It is easy to combine both static and dynamic content with JSP. The factor to develop JSP technology is to use regular HTML tags. The JSP author can place servlet or simple java code in the page by using special designated tags. The entire JSP will be translated into Servlet and the servlet related code is communicated to container to run.

- **EJB Technology**: EJB is a server-side web component. It depends on other Java technologies for proper functionality such as Remote Method Invocation. RMI is used as a protocol between 2 enterprise beans and between an enterprise bean and its client. EJB encapsulates the business logic. All EJBs are developed, deployed and run only in an EJB container. This is similar to servlets and JSP run in a web container.

EJB applications are easy to develop because the applications developer can concentrate on business logic. The developer can utilize the services provided by the EJB container, like connection pooling and transactions.

- **RMI/IIOP:** RMI stands for Remote Method Invocation. IIOP stands for Inter Internet-ORB Protocol.

RMI/IIOP API is used to write distributed objects using Java technology, which enables communication between objects in the memory, across JVM and also physical devices.

RMI/IIOP yields the benefits of OOP such as inheritance, polymorphism and encapsulation and is platform independent. In RMI/IIOP, the code related to network is written by applying the interface but not the implementations. The operation can solely on the interface that object’s class.

RMI-IIOP relies on object serialization for passing parameters via remote method invocations.

- **JNDI (Java Naming and Directory Interface) API :** To enable java programs to access the naming and directory services, the JNDI API is utilized. Naming services emphasizes on the services that are to associate names with objects.

We are familiar with naming systems such as the file system which has a directory or path associated it. Surfing web is associated a name that is the URL called Domain Naming System. EJB components of a J2EE application server, user profiles are associated in LDAP(Lightweight Directory Access Protocol) directory.

For example, JNDI is the best API to write a java application that is used for search utility over network-enabled desktop, class-browser or an address book search utility.

- **JDBC (Java Database Connectivity) API :** Many java applications use a database and database accessing and programming is a significant role in web application development. JDBC is an API that enables the accessibility to a database in order to manipulate the database. The JDBC API supports both two-tier and three-tier models for database access.
    - Two-tier model -- a Java application interacts directly with the database.
    - Three-tier model -- introduces a middle-level server for execution of business logic: the middle tier to maintain control over data access.

The application that uses the JDBC implements the following sequence.
1. Importing Packages
2. Registering the JDBC Drivers
3. Opening a Connection to a Database
4. Creating a Statement Object
5. Executing a query and Returning a Result Set Object
6. Processing the Result Set
7. Closing the Result Set and Statement Objects
8. Closing the Connection

Java Mail API: Mailing and Messaging applications can be modelled or developed using Java Mail. The e-mail messaging applications can be developed for both high-level implementation and low-level implementation. That is, a small company which is heading towards growth can develop a solution which ensures the accessing to their mail server in an efficient manner. A blue-chip company can focus on providing the access to industry-level access which is a wider reach and provides vast support.

JMS: Java Message Services is an API that enables the components of an application, could it be JSE or JEE component to provide messaging services such as create, send, receive and read messages. Using the

loosely coupled, reliable and asynchronous communications are enabled by JMS. The JMS application contains the following parts:
JMS Provider: It implements the JMS interface and provides the administration and control services for the messages. J2EE 1.3 includes the JMS Provider service.
JMS Clients: These are the java applications that produce or consume the messages.
Messages: These are the objects that communicate the messages between clients.
Administered Objects: They are the JMS objects created by the administrator that can be used by the clients. They are namely destinations and connection factories.
Native Clients: These are the applications that use message client's native client API instead of JMS API.

8. What is N tier architecture? Explain its advantages with example.
Three-Tier Client/Server Architecture:
In this system, the client implements presentation logic (thin client). The application server implements the business logic and the data resides on database server.
Multi-Tier Architecture:
The front end component is responsible for providing portable presentation logic. The back-end component acts as database server. The middle tier component allows the users to share and control business logic by isolating it from actual application. This system is fat in the middle. The client system interacts with the middle tier through a standard protocol such as HTTP or RPC. The middle tier interacts with the backend server through standard database protocols such as SQL, ODBC and JDBC.
N-Tier Architecture:
With four or more tiers, each layer can be further decomposed to allow various parts of the system to scale independently. More sophiscated multi-tier solutions appear in this model. This architecture leads to reduction of network traffic, faster network communications, greater reliability and greater overall performance.
Benefits of N-Tier Architecture:

- Database drivers are installed and configured on the server-side, rather than on client machines. Hence deployment costs are low.
- Data base switching costs are low: There is a middle tier for data access. This enables the users to migrate database schemas, or change the different database drivers without redeploying the clients.
- Changing the business logic layer may not necessitate the redeploying the client tier.
- By placing a firewall between the presentation and business logic tiers, high security can be provided to the data easily.
- Rather than, the business components acquiring and releasing connections to the resources such as databases, the resources can be pooled and reused for different client requests. Resource pooling can also be applied to other resources such as threads and socket connections. Business components themselves can be pooled by multiple clients.
- Since there are many tiers and each tier is independent, the database images can be added while minimizing the changes and recompiling other tiers.
- If one tier is overloaded, other tier can still function properly there by improving the performance.
- If critical error occurs, it is localized to a single tier.

Application of N-Tier Architecture:
It plays a major role in internet and intranet services, transaction processing monitors, distributed computing and most other growing software technologies.
N-Tier provides a wide range of benefits to the companies longing for flexible and reliable solution to complex, and constantly changing problems. It also provides information and tools to solve some of the challenges faced by IT professionals.

9. List the difference between directory and database. Also explain LDAP.

Directories	**Relational Databases**
Read more frequently than written	Written more frequently than read
Handle small, simple units of data	Handle large, complex, transaction-oriented units of data
Distributed widely	Not distributed widely
Store information in hierarchically arranged entries	Store information as records in relational tables

LDAP Architecture

Lightweight Directory Access Protocol (also known as LDAP) is an application protocol. This protocol is used specifically for querying data as well as modifying said data. This is performed by using directory services –that is, a software system that stores, organises, and provides access to the information that is in a directory– running through a TCP/IP. The main function of any directory is to act as a set of objects with logically and hierarchically organised attributes –such as the telephone directory.

Benefits of LDAP

LDAP simplifies directory management in the following ways.

- It provides users and applications in an enterprise with a single, well-defined, standard interface to a single, extensible directory service.
- It reduces the need to manage and coordinate application-specific directories
- Its well-defined protocol and array of programmatic interfaces make it more practical to deploy internet-ready applications that leverage the directory.

LDAP defines the content of messages exchanged between an LDAP client and an LDAP server. The messages specify the operations requested by the client (that is, search, modify, and delete), the responses from the server, and the format of data carried in the messages. LDAP messages are carried over TCP/IP, a connection-oriented protocol, so there are also operations to establish and disconnect a session between the client and server.

The general interaction between an LDAP client and an LDAP server takes the following form:

1. The client establishes a session with an LDAP server. This is known as binding to the server. The client specifies the host name or IP address and TCP/IP port number where the LDAP server is listening.
2. The client can provide a user name and a password to properly authenticate with the server, or the client can establish an anonymous session with default access rights. The client and server can also establish a session that uses stronger security methods such as encryption of data.
3. The client then performs operations on directory data. LDAP offers both read and update capabilities. This allows directory information to be managed as well as queried. LDAP also supports searching the directory for data meeting arbitrary user-specified criteria. Searching is a very common operation in LDAP. A user can specify what part of the directory to search and what information to return. A search filter that uses Boolean conditions specifies what directory data matches the search.
4. When the client is finished making requests, it closes the session with the server. This is also known as unbinding.

10. What is application server? Develop a program for accessing the entries in LDAP.

An **application server** is a component-based product that resides in the middle-tier of a **server** centric architecture. It provides middleware services for security and state maintenance, along with data access and persistence. Java **application servers** are based on the Java 2 Platform, Enterprise Edition (J2EE).

LDAP stands for Lightweight Directory Access Protocol. It is usually used to fetch (and sometimes update) data in a directory of people. For example the employees and students of a University.

For example, Active Directory, which is used in Microsoft Windows based networks to hold the accounts of all he users, provides a way to access it via LDAP.

Using Net::LDAP can provide a way to interact with this database. For example you might build an in-house web application and instead of managing your own user database, you could let the users authenticate using their username/password in the Windows network.

```
#!/usr/bin/env perl
use strict;
use warnings;

use Net::LDAP;
my $server = "ldap.itd.umich.edu";
my $ldap = Net::LDAP->new( $server ) or die $@;
$ldap->bind;

my $result = $ldap->search(
    base   => "",
    filter => "(&(cn=Jame*) (sn=Woodw*))",
);

die $result->error if $result->code;

printf "COUNT: %s\n", $result->count;

foreach my $entry ($result->entries) {
    $entry->dump;
}
print "===============================================\n";

foreach my $entry ($result->entries) {
    printf "%s <%s>\n",
        $entry->get_value("displayName"),
        ($entry->get_value("mail") || ");
}

$ldap->unbind;
```

Running the above code will print:

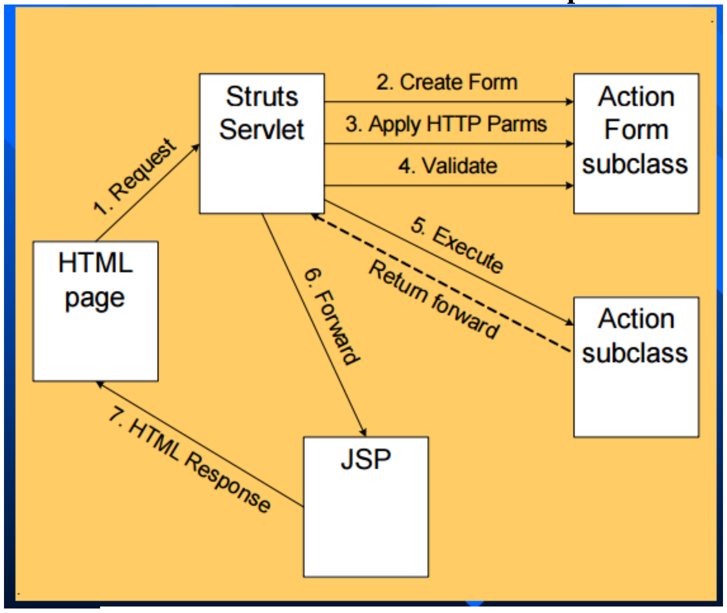
```
COUNT: 2
------------------------------------------------------------------------
dn:uid=jrwood,ou=People,dc=umich,dc=edu

   objectClass: umichPerson
                top
                person
                inetOrgPerson
                posixAccount
           uid: jrwood
            sn: Woodworth
            cn: James R Woodworth
                James Woodworth
     uidNumber: 75786
     gidNumber: 10
 homeDirectory: /users/jrwood
```

```
RealtimeBlockList: TRUE
               ou: Alumni
      displayName: James R Woodworth
          krbName: jrwood@umich.edu
--------------------------------------------------------------------
dn:uid=jwoodnh,ou=People,dc=umich,dc=edu

          krbName: jwoodnh@umich.edu
        uidNumber: 99417910
       loginShell: /bin/csh
               sn: Woodward
               ou: Alumni
             mail: jwoodnh@umich.edu
    homeDirectory: /users/jwoodnh
RealtimeBlockList: TRUE
      displayName: James Alan Woodward
              uid: jwoodnh
        gidNumber: 10
               cn: James Alan Woodward
                   James A Woodward
                   James Woodward
      objectClass: umichPerson
                   inetOrgPerson
                   organizationalPerson
                   person
                   top
                   posixAccount
==========================================
James R Woodworth <>
James Alan Woodward <jwoodnh@umich.edu>
```

UNIT – II

PART – B

With neat sketch explain about struts framework



- Struts is an open-source framework for building more flexible, maintainable and structured front-ends in Java web applications
- There are two key components in a web application:
  - the data and business logic performed on this data
  - the presentation of data
- Struts
  - helps structuring these components in a Java web app.
  - controls the flow of the web application, strictly separating these components
  - unifies the interaction between them
- This separation between presentation, business logic and control is achieved by implementing the Model-View-Controller (MVC) Design Pattern

Model A model represents an application’s data and contains the logic for accessing and manipulating that data. Any data that is part of the persistent state of the application should reside in the model objects. The services that a model exposes must be generic enough to support a variety of clients. By glancing at the model's public method list, it should be easy to understand how to control the model's behavior. A model groups related data and operations for providing a specific service; these group of operations wrap and abstract the functionality of the business process being modeled. A model’s interface exposes methods for accessing and updating the state of the model and for executing complex processes encapsulated inside the model. Model services are accessed by the controller for either querying or effecting a change in the model state. The model notifies the view when a state change occurs in the model.

View The view is responsible for rendering the state of the model. The presentation semantics are encapsulated within the view, therefore model data can be adapted for several different kinds of clients. The view modifies itself when a change in the model is communicated to the view. A view forwards user input to the controller.

Controller The controller is responsible for intercepting and translating user input into actions to be performed by the model. The controller is responsible for selecting the next view based on user input and the outcome of model operations.

1. Explain about struts classes with suitable example

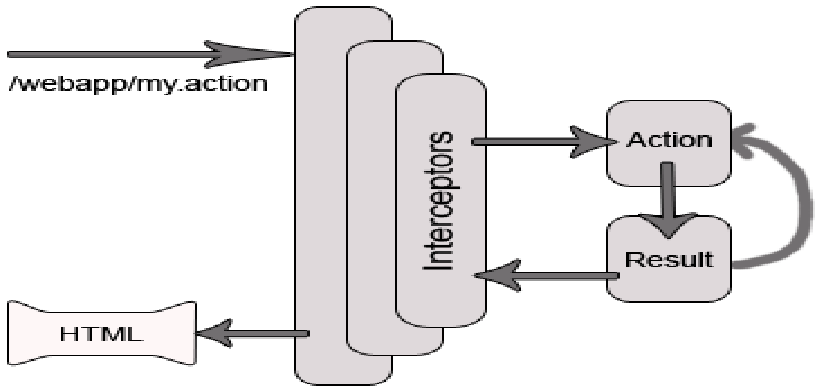
- **Action -** The basic action class in which we implement our business logic.
- **Include Action -** Similar as include page directive in jsp.
- **Forward Action -** Used in case we need to forward the request from one JSP to another. If we directly forward the request from one jsp it violates the MVC architecture. Hence an action class is used to do this job.
- **Dispatch Action -** Handles multiple operations in multiple methods. It is better to have one method per operation instead of merging the entire business logic in a single execute method of an action class.
- **Look up Dispatch Action -** Same as dispatch action but recommended not to use.
- **Switch Action -** used to switch between different modules in struts application.

Most commonly used action classes are:

- Action
- Dispatch Action

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
  "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<constant name="struts.devMode" value="true" />
  <package name="helloworld" extends="struts-default">

    <action name="index">
        <result >/index.jsp</result>
    </action>

    <action name="hello"
        class="com.tutorialspoint.struts2.HelloWorldAction"
        method="execute">
        <result name="success">/HelloWorld.jsp</result>
    </action>
  </package>
</struts>
```

2. Explain about struts lifecycle with suitable diagram



Steps for request lifecycle in strut two applications:-

1) In first step user sends a request to the server for some resource.
2) The filterDispatcher accept the request and then it determines the appropriate action.
3) The interceptors configured for applying the common functionalities like workflow, validation, and file upload etc and these functionalities are automatically to the request. The Interceptors are helps to specify the "request-processing lifecycle" for an action. Interceptors are configured to apply the common functionalities like workflow, validation etc to the request.
4) The action method executed to perform the database related operations such as storing and retrieving data from the database.
5) The result rendered the output.
6) The request returns through the interceptors in the reverse order. The retuning request permits us to perform the clean up or additional processing.
7) The final step is returned to the servlet container which sends the output to the user browser.

3. Explain about the struts flow with sample application for employee search (Any two option)

The sample application is called *Mini HR* and it will have a basic opening page that links to an Employee Search page. From the Employee Search page, users can search for employees by name or social security number. After executing the search, the Search page will be redisplayed with a list of employees that match the search criteria. Although quite simple, and limited in scope, this example illustrates the key features common to any Struts-based Web application.

The Mini HR Application Files

All Struts applications are comprised of several files, which contain the various parts of a Struts program. Some are Java source files, but others contain JSP and XML. A properties file is also required. Because of the relatively large number of files required by a Struts application, we will begin by examining the files required by Mini HR. The same general types of files will be needed by just about any Struts application. The following table lists each file required by Mini HR and its purpose.

File	**Description**
index.jsp	Contains the JSP that is used as a gateway page for the Mini HR application and provides a link to the Employee Search page.
search.jsp	Contains the JSP that is used for performing employee searches and displaying the search results.
SearchForm.java	Contains the class that captures and transfers data to and from the Search page. This is a View class.
SearchAction.java	Contains the class code that processes requests from the Search page. This is a Controller class.
EmployeeSearchService.java	Contains the class that encapsulates the business logic and data access involved in searching for employees. This is a Model class.
Employee.java	Contains the class that represents an employee and encapsulates all of an employee's data. This is a Model class.
web.xml	Contains the XML that is used to configure the Java Web application properties for the Mini HR application.
struts-config.xml	Contains the XML that is used to configure the Struts framework for this application.
ApplicationResources.properties	Contains properties that are used to externalize application strings, labels, and messages so that they can be changed without having to recompile the application. This file is also used for internationalizing the application.

The following sections examine each of the Mini HR application files in detail, and in many cases line by line. First, though, it's necessary to explain where each file should be placed in a directory hierarchy. Because this application (and all other Struts applications) will be deployed to a J2EE servlet container, the application files have to be arranged in the standard J2EE Web Archive (**.war**) format, which is simply a Java Archive (**.jar**) file with a different extension (**.war**). The Web Archive format also specifies a few key requirements for the **.jar** file:

- There must be a directory at the root level of the archive named **WEB-INF**. At run time this is a protected directory and thus any files beneath it will be inaccessible to browsers.
- There must be a Web application deployment descriptor file named **web.xml** beneath the **WEB-INF** directory. This file will be explained later in this chapter in the 'web.xml' section.
- Any libraries needed by the application should be under a directory called **lib** located beneath the **WEB-INF** directory.
- Any class files needed by the application, which are not already packaged in a **.jar** file, should be under a directory called **classes** located beneath the **WEB-INF** directory.

For the Mini HR application, you will create a directory called **MiniHR**. In principle, you can place this directory anywhere, but to follow along with this example, put it at **c:\java**. You'll use the **c:\java\MiniHR** directory as the root of your Web application so that you can easily create a Web Archive file later. Following is the layout of the **c:\java\MiniHR** directory, shown in Figure 2-1, and the location of each file examined in this section. You will need to place the files in this exact structure.

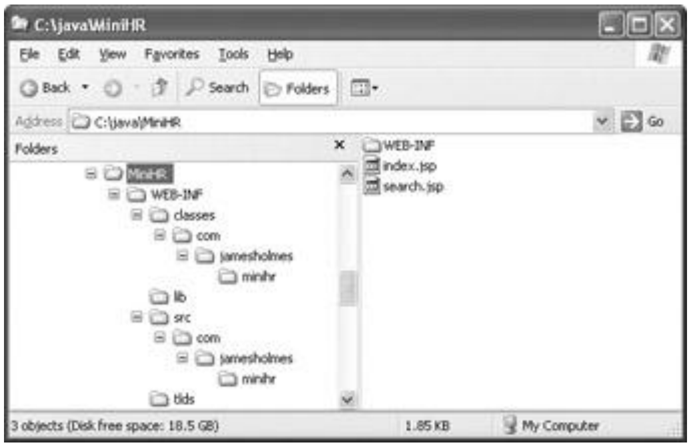


Figure 2-1: The c:\java\MiniHR directory layout

c:\java\MiniHR\index.jsp
c:\java\MiniHR\search.jsp
c:\java\MiniHR\WEB-INF\web.xml
c:\java\MiniHR\WEB-INF\struts-config.xml
c:\java\MiniHR\WEB-INF\classes\com\jamesholmes\minihr\ApplicationResources.properties
c:\java\MiniHR\WEB-INF\lib
c:\java\MiniHR\WEB-INF\src\com\jamesholmes\minihr\Employee.java
c:\java\MiniHR\WEB-INF\src\com\jamesholmes\minihr\EmployeeSearchService.java
c:\java\MiniHR\WEB-INF\src\com\jamesholmes\minihr\SearchAction.java
c:\java\MiniHR\WEB-INF\src\com\jamesholmes\minihr\SearchForm.java
c:\java\MiniHR\WEB-INF\tlds

index.jsp

The **index.jsp** file, shown here, is a very simple JSP that is used to render Mini HR's opening screen:

```
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>

<html>
<head>
<title>ABC, Inc. Human Resources Portal</title>
</head>
<body>
```

```
<font size="+1">ABC, Inc. Human Resources Portal</font><br>
<hr width="100%" noshade="true">

&#149; Add an Employee<br>
&#149; <html:link forward="search">Search for Employees</html:link><br>

</body>
</html>
```

You'll notice that **index.jsp** is comprised mostly of standard HTML, with the exception of the JSP tag library definition at the top of the file and the 'Search for Employees' link. The **index.jsp** file uses Struts' HTML Tag Library to render the Search link. Before you can use the HTML Tag Library, you have to 'import' it into the JSP with the following line at the top of the JSP:

```
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>
```

This line associates the tag library located at **/WEB-INF/tlds/struts-html.tld** with a prefix, or 'handle,' of html. That way, any time a tag from the Struts HTML Tag Library is used, it will be prefixed with html. In **index.jsp**'s case, the **link** tag is used with the following line:

```
<html:link forward="search">Search for Employees</html:link>
```

Of course, if you wanted to use another prefix for the tag library, you could do so by updating the **prefix** attribute of the tag library import on the first line of the file.

The HTML Tag Library's **link** tag is used for rendering an HTML link, such as http://www.jamesholmes.com/. The **link** tag goes beyond basic HTML, though, by allowing you to access link, or *forward* definitions, from the Struts configuration file, **struts-config.xml**, which is covered later in this chapter in the 'struts-config.xml' section. In this case, the tag looks for a forward definition named 'search' defined in the **struts-config.xml** file to use for the link being generated. If you skip ahead to the 'struts-config.xml' section of this chapter, you'll see that the forward definition is as follows:

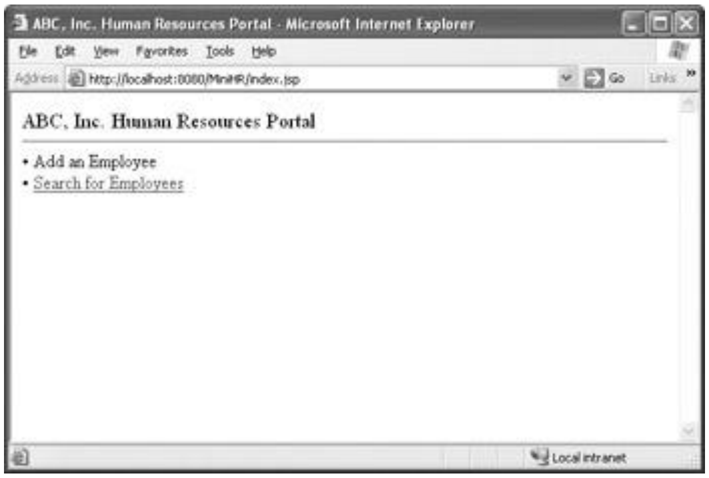
```
<!-- Global Forwards Configuration -->
<global-forwards>
  <forward name="search" path="/search.jsp"/>
</global-forwards>
```

Forward definitions allow you to declaratively configure the location to which a link points instead of hard-coding that information into your JSP or application. As you'll see in Chapter 5, forward definitions are used throughout Struts to direct the flow of an application from the Struts configuration file.

The following is the source code generated after **index.jsp** has been requested in the browser. Notice that the Search page link has been converted into a standard HTML link.

```
<html>
<head>
<title>ABC, Inc. Human Resources Portal</title>
</head>
<body>

<font size="+1">ABC, Inc. Human Resources Portal</font><br>

<hr width="100%" noshade="true">

&#149; Add an Employee<br>
&#149; <a href="/MiniHR/search.jsp">Search for Employees</a><br>

</body>
</html>
```

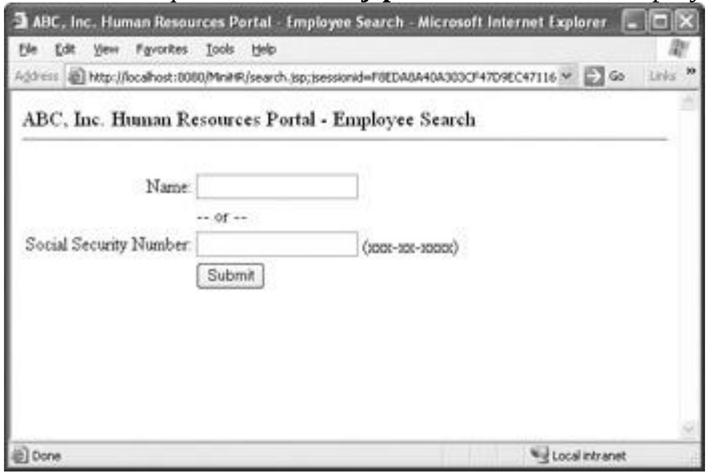
Here is how **index.jsp** looks in the browser.


ABC, Inc. Human Resources Portal

• Add an Employee
• Search for Employees

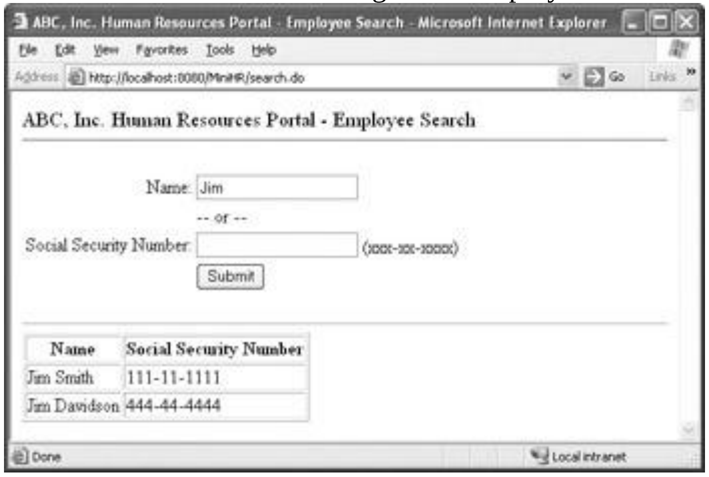
search.jsp

The **search.jsp** file is responsible for the bulk of the Employee Search functionality in the Mini HR application. When the Employee Search link is selected from the **index.jsp** page, **search.jsp** is executed. This initial request for **search.jsp** renders the basic Employee Search screen shown here:


ABC, Inc. Human Resources Portal - Employee Search

Name:
-- or --
Social Security Number: (xxx-xx-xxxx)

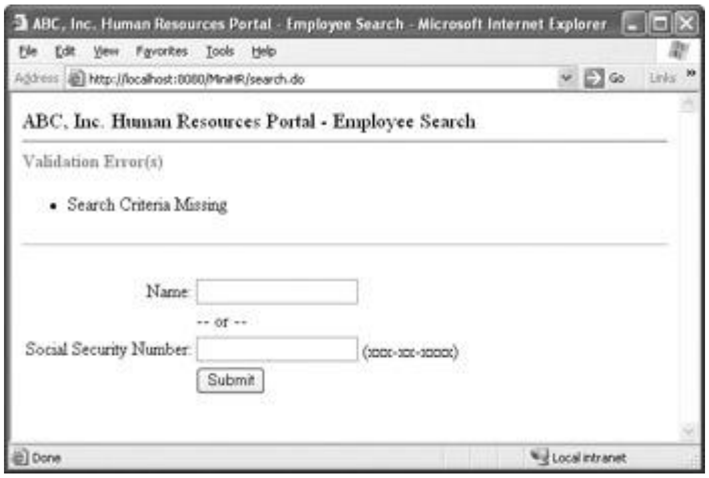
Each time a search is performed, Struts' Controller servlet is executed and eventually **search.jsp** is executed to handle the rendering of the Employee Search screen, with the search results, as shown here:


ABC, Inc. Human Resources Portal - Employee Search

Name: Jim
-- or --
Social Security Number: (xxx-xx-xxxx)

Name	Social Security Number
Jim Smith	111-11-1111
Jim Davidson	444-44-4444

Similarly, if there are any errors with the search criteria when the search is submitted, **search.jsp** is executed to report the errors, as shown here:


ABC, Inc. Human Resources Portal - Employee Search - Microsoft Internet Explorer

http://localhost:8080/MiniHR/search.do

ABC, Inc. Human Resources Portal - Employee Search

Validation Error(s)

- Search Criteria Missing

Name:

-- or --

Social Security Number: (xxx-xx-xxxx)

Submit

The contents of **search.jsp** are

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic" %>

<html>
<head>
<title>ABC, Inc. Human Resources Portal - Employee Search</title>
</head>
<body>

<font size="+1">
ABC, Inc. Human Resources Portal - Employee Search
</font><br>
<hr width="100%" noshade="true">

<html:errors/>

<html:form action="/search">

<table>
<tr>
<td align="right"><bean:message key="label.search.name"/>:</td>
<td><html:text property="name"/></td>
</tr>
<tr>
<td></td>
<td>-- or --</td>
</tr>
<tr>
<td align="right"><bean:message key="label.search.ssNum"/>:</td>
<td><html:text property="ssNum"/> (xxx-xx-xxxx)</td>
</tr>
<tr>
<td></td>
<td><html:submit/></td>
</tr>
</table>
```

```
</html:form>

<logic:present name="searchForm" property="results">

<hr width="100%" size="1" noshade="true">

<bean:size id="size" name="searchForm" property="results"/>
<logic:equal name="size" value="0">
<center><font color="red"><b>No Employees Found</b></font></center>
</logic:equal>

<logic:greaterThan name="size" value="0">
<table border="1">
<tr>
<th>Name</th>
<th>Social Security Number</th>
</tr>
<logic:iterate id="result" name="searchForm" property="results">
<tr>
<td><bean:write name="result" property="name"/></td>
<td><bean:write name="result" property="ssNum"/></td>
</tr>
</logic:iterate>
</table>
</logic:greaterThan>

</logic:present>

</body>
</html>
```

Because of its size and importance, we will examine it closely, line by line.
Similar to **index.jsp**, **search.jsp** begins by declaring the JSP tag libraries that will be used by the JSP:

```
<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic" %>
```

In addition to the HTML Tag Library used by **index.jsp**, **search.jsp** uses Struts' Bean and Logic libraries. These additional tag libraries contain utility tags for working with Java beans and using conditional logic in a page, respectively.
The next several lines are comprised of basic HTML tags:

```
<html>
<head>
<title>ABC, Inc. Human Resources Portal - Employee Search</title>
</head>
<body>

<font size="+1">
ABC, Inc. Human Resources Portal - Employee Search
</font><br>
<hr width="100%" noshade="true">
```

Immediately following this basic HTML is this **errors** tag definition:

```
<html:errors/>
```

Recall that **search.jsp** is used to render any errors that occur while validating that the search criteria are sound. The HTML Tag Library's **errors** tag will emit any errors that are passed to the JSP from the **SearchForm** object. This is covered in more detail in the 'SearchForm.java'section in this chapter.
The next several lines of **search.jsp** are responsible for rendering the HTML for the search form:

```
<html:form action="/search">

<table>
<tr>
<td align="right"><bean:message key="label.search.name"/>:</td>
<td><html:text property="name"/></td>
</tr>
<tr>
<td></td>
<td>-- or --</td>
</tr>
<tr>
<td align="right"><bean:message key="label.search.ssNum"/>:</td>
<td><html:text property="ssNum"/> (xxx-xx-xxxx)</td>
</tr>
<tr>
<td></td>
<td><html:submit/></td>
</tr>
</table>

</html:form>
```

Before discussing the specifics of the search form, let's review the use of the Bean Tag Library in this snippet. This snippet uses the library's **message** tag, as shown here:

<td align="right"><**bean:message key="label.search.name"/**>:</td>

The **message** tag allows externalized messages from the **ApplicationResources .properties** file to be inserted into the JSP at run time. The **message** tag simply looks up the key passed to it in **ApplicationResources.properties** and returns the corresponding message from the file. This feature is especially useful to internationalize a page and to allow easy updating of messages outside the JSP. *Internationalization* is the process of providing content specific to a language, locale, or region. For instance, internationalization would be to create both English and Spanish versions of the same JSP.

Now, it's time to examine the form. Struts' HTML Tag Library has a tag for each of the standard HTML form tags, such as

```
<form>
<input type=">
```

and so on. Instead of using the standard HTML tags, you'll use the HTML Tag Library's equivalent tag, which ties the form to Struts. For example, the **text** tag (<**html:text**>) renders an <**input type='text' …**> tag. The **text** tag goes one step further, though, by allowing a property to be associated with the tag, as shown here:

<td><**html:text property="name"/**></td>

The property 'name' here corresponds to the field named **name** in the **SearchForm** object. That way, when the tag is executed, it places the value of the **name** field in the HTML at run time. Thus, if the **name** field had a value of 'James Holmes' at run time, the output from the tag would look like this:

<td><**input type="text" name="name" value="James Holmes"**></td>

At the beginning of this next snippet, the HTML Tag Library's **form** tag is used to render a standard HTML <**form**> tag. Notice, however, that it specifies an **action** parameter of '/search' as shown here:

```
<html:form action="/search">
```

The **action** parameter associates an **Action** object mapping from the **struts-config.xml** file with the form. That way, when the form is submitted, the processing will be handled by the specified **Action** object.

The final section of the **search.jsp** file contains the logic and tags for rendering search results:

```
<logic:present name="searchForm" property="results">

<hr width="100%" size="1" noshade="true">

<bean:size id="size" name="searchForm" property="results"/>
<logic:equal name="size" value="0">
<center><font color="red"><b>No Employees Found</b></font></center>
</logic:equal>

<logic:greaterThan name="size" value="0">
<table border="1">
<tr>
<th>Name</th>
<th>Social Security Number</th>
</tr>
<logic:iterate id="result" name="searchForm" property="results">
<tr>
<td><bean:write name="result" property="name"/></td>
<td><bean:write name="result" property="ssNum"/></td>
</tr>
</logic:iterate>
</table>
</logic:greaterThan>

</logic:present>
```

The beginning of this snippet uses Struts' Logic Tag Library for using conditional logic in a JSP. The Logic Library's **present** tag checks an object to see if a particular property is present. In this case, the **logic** tag checks to see if the **results** field of the **SearchForm** has been set. If so, then all of the HTML and JSP tags inside the <**logic:present …**> tag will be executed. Otherwise, they will be ignored.

The rest of the tags in this snippet are responsible for rendering the search results. First, the Bean Library's **size** tag gets the size of the **results ArrayList** from the **SearchForm** object. Next, the size is checked to see if it is 0 using the Logic Library's **equal** tag. If the size is in fact 0, then a 'No Employees Found' message will be rendered. Otherwise, each of the employees returned from the search will be displayed. The Logic Library's **iterate** tag is used to iterate over each of the search results. Each search result is assigned to a variable named **result** by the**iterate** tag. Inside the **iterate** tag the Bean Library's **write** tag is used to access the **result** variable's name and **ssNum** fields.

SearchForm.java

The **SearchForm** class, shown next, is a View class that is used to capture and transfer data to and from the Employee Search page. When the HTML form on the Search page is submitted, Struts' **ActionServlet** will populate this class with the data from the form. Notice that there will be a one-to-one mapping between fields on the page and fields in the class with getter and setter methods. Struts uses encapsulation and Java's reflection mechanism to call the method corresponding to each field from a page. Additionally, when **SearchAction** (the Controller class for the Search page) executes, it will populate this object with the search results so that they can be transferred back to the Search page.

```
package com.jamesholmes.struts;
```

```
import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class SearchForm extends ActionForm
{
  private String name = null;
  private String ssNum = null;
  private List results = null;

  public void setName(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }

  public void setSsNum(String ssNum) {
    this.ssNum = ssNum;
  }

  public String getSsNum() {
    return ssNum;
  }

  public void setResults(List results) {
    this.results = results;
  }

  public List getResults() {
    return results;
  }

  // Reset form fields.
  public void reset(ActionMapping mapping, HttpServletRequest request)
  {
    name = null;
    ssNum = null;
    results = null;
  }

  // Validate form data.
  public ActionErrors validate(ActionMapping mapping,
    HttpServletRequest request)
```

```
{
  ActionErrors errors = new ActionErrors();

  boolean nameEntered = false;
  boolean ssNumEntered = false;

  // Determine if name has been entered.
  if (name != null && name.length() > 0) {
    nameEntered = true;
  }

  // Determine if social security number has been entered.
  if (ssNum != null && ssNum.length() > 0) {
    ssNumEntered = true;
  }

  /* Validate that either name or social security number
     has been entered. */
  if (!nameEntered && !ssNumEntered) {
    errors.add(null,
      new ActionError("error.search.criteria.missing"));
  }

  /* Validate format of social security number if
     it has been entered. */
  if (ssNumEntered && !isValidSsNum(ssNum.trim())) {
    errors.add("ssNum",
      new ActionError("error.search.ssNum.invalid"));
  }

  return errors;
}

// Validate format of social security number.
private static boolean isValidSsNum(String ssNum) {
  if (ssNum.length() < 11) {
    return false;
  }

  for (int i = 0; i < 11; i++) {
    if (i == 3 || i == 6) {
      if (ssNum.charAt(i) != '-') {
        return false;
      }
    } else if ("0123456789".indexOf(ssNum.charAt(i)) == -1) {
      return false;
    }
  }

  return true;
}
```

}

ActionForm subclasses, including **SearchForm**, are basic Java beans with a couple of extra Struts-specific methods: **reset( )** and **validate( )**. The **reset( )** method is used to clear out, or 'reset,' an **ActionForm**'s data after it has been used for a request. Because Struts reuses**ActionForm**s instead of creating new ones for each request, this method is necessary to ensure that data from different requests is not mixed. Typically, this method is used to just set class fields back to their initial states, as is the case with **SearchForm**. However, as you'll see inChapter 4, this method can be used to perform other necessary logic for resetting an **ActionForm** object.

The **validate( )** method of **ActionForm** is called to perform basic validations on the data being transferred from an HTML form. In **SearchForm**'s case, the **validate( )** method first confirms that a name and social security number have been entered. If a social security number has been entered, **SearchForm** goes one step further and validates the format of the social security number with the **isValidSsNum( )** method. The **isValidSsNum( )** method simply ensures that an 11-character string was entered and that it conforms to the following format: three digits, hyphen, two digits, hyphen, four digits (e.g., 111-22-3333). Note that business-level validations, such as looking up a social security number in a database to make sure it is valid, are considered business logic and should be in a Model layer class. The validations in an **ActionForm**are meant to be very basic, such as just confirming that data was entered, and should not be used for performing any real business logic.

You'll notice that the **validate( )** method returns an **ActionErrors** object and the validations inside the method populate an **ActionErrors** object if any validations fail. The **ActionErrors** object is used to transfer validation error messages to the screen. Remember from the discussion of **search.jsp** that the HTML Tag Library's **errors** tag will emit any errors in a JSP if they are present. Following is the snippet from **search.jsp**:

```
<html:errors/>
```

Here in the **ActionForm** class, you simply place the keys for messages into the **ActionErrors** object, such as 'error.search.criteria.missing'. The **errors** tag will use these keys to load the appropriate messages from the **ApplicationResources.properties** file, discussed in the section of the same name later in this chapter.

Note that the **validate( )** method will be invoked only if the **validate** parameter of the <**action**> tag is set to 'true' in the **struts-config.xml** file.

SearchAction.java

The **SearchAction** class, shown next, is a Controller class that processes requests from the Search page:

```
package com.jamesholmes.struts;

import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public final class SearchAction extends Action
{
  public ActionForward execute(ActionMapping mapping,
    ActionForm form,
    HttpServletRequest request,
    HttpServletResponse response)
    throws Exception
```

```
 {
   EmployeeSearchService service = new EmployeeSearchService();
   ArrayList results;

   SearchForm searchForm = (SearchForm) form;

   // Perform employee search based on what criteria was entered.
   String name = searchForm.getName();
   if (name != null && name.trim().length() > 0) {
     results = service.searchByName(name);
   } else {
     results = service.searchBySsNum(searchForm.getSsNum().trim());
   }

   // Place search results in SearchForm for access by JSP.
   searchForm.setResults(results);

   // Forward control to this Action's input page.
   return mapping.getInputForward();
 }
}
```

Remember from the discussion of **search.jsp** that the HTML form on the page is set to post its data to the '/search' action. The **strut-config.xml** file maps the search action to this class so that when **ActionServlet** (Controller) receives a post from the Search page, it delegates processing for the post to this **Action** subclass. This mapping is shown here:

```
<!-- Action Mappings Configuration -->
<action-mappings>
 <action path="/search"
        type="com.jamesholmes.struts.SearchAction"
        name="searchForm"
        scope="request"
        validate="true"
        input="/search.jsp">
   <forward name="results" path="/results.jsp"/>
 </action>
</action-mappings>
```

Struts' **Action** subclasses manage the processing of specific requests. You can think of them as mini servlets assigned to manage discreet Controller tasks. For instance, in the preceding example, **SearchAction** is responsible for processing employee search requests and acts as a liaison between the Model (**EmployeeSearchService**) and the View (**search.jsp**).

SearchAction begins by overriding Struts' **Action** class **execute( )** method. The **execute( )** method is the single point of entry for an **Action** class by Struts' **ActionServlet**. You'll notice that this method takes an **HttpServletRequest** object and an **HttpServletResponse** object as parameters, similar to a servlet's **service( )**, **doGet( )**, and **doPost( )** methods. Additionally, **execute( )** takes a reference to the **ActionForm** associated with this **Action** and an **ActionMapping** object reference. The **ActionForm** reference passed to this **Action** will be an instance of**SearchForm**, as discussed in the previous section, 'SearchForm.java.' The **ActionMapping** reference passed to this **Action** will contain all of the configuration settings from the **struts-config.xml** file for this **Action**.

The **execute( )** method begins by instantiating a few objects, and then the real work gets underway with a check to see what search criteria was entered by the user. Notice that the **ActionForm** object passed in is cast to its native type: **SearchForm**. Casting the object allows you to access **SearchForm**'s methods for retrieving the search criteria. Based on the criteria entered, one of **EmployeeSearchService**'s methods will be invoked to perform the employee search. If an employee name was entered, the **searchByName( )** method will be invoked. Otherwise, the **searchBySsNum( )** method will be invoked. Both search methods return an **ArrayList** containing the search results. This **results ArrayList** is then added to the **SearchForm** instance so that **search.jsp** (View) can access the data.

The **execute( )** method concludes by forwarding control to **SearchAction**'s input page: **search.jsp**. The input page for an action is declared in the **struts-config.xml** file, as shown here for **SearchAction**, and is used to allow an action to determine from which page it was called:

```
<action path="/search"
        type="com.jamesholmes.struts.SearchAction"
        name="searchForm"
        scope="request"
        validate="true"
        input="/search.jsp">
```

EmployeeSearchService.java

EmployeeSearchService is a Model class that encapsulates the business logic and data access routines involved in searching for employees. The **SearchAction** Controller class uses this class to perform an employee search and then shuttles the resulting data to the View layer of the Mini HR application. **EmployeeSearchService** is shown here:

```
package com.jamesholmes.struts;

import java.util.ArrayList;

public class EmployeeSearchService
{
  /* Hard-coded sample data. Normally this would come from a real data
     source such as a database. */
  private static Employee[] employees =
  {
    new Employee("Bob Davidson", "123-45-6789"),
    new Employee("Mary Williams", "987-65-4321"),
    new Employee("Jim Smith", "111-11-1111"),
    new Employee("Beverly Harris", "222-22-2222"),
    new Employee("Thomas Frank", "333-33-3333"),
    new Employee("Jim Davidson", "444-44-4444")
  };

  // Search for employees by name.
  public ArrayList searchByName(String name) {
    ArrayList resultList = new ArrayList();

    for (int i = 0; i < employees.length; i++) {
      if(employees[i].getName().toUpperCase().indexOf(name.toUpperCase())
         != -1)
      {
        resultList.add(employees[i]);
      }
    }

    return resultList;
  }
  // Search for employee by social security number.
  public ArrayList searchBySsNum(String ssNum) {
    ArrayList resultList = new ArrayList();

    for (int i = 0; i < employees.length; i++) {
      if (employees[i].getSsNum().equals(ssNum)) {
        resultList.add(employees[i]);
      }
    }

    return resultList;
  }
}
```

In order to simplify Mini HR, the **EmployeeSearchService** class will not actually communicate with a real data source, such as a database, to query employee data. Instead, **EmployeeSearchService** has some sample **Employee** data hard-coded at the top of the class, as shown here:

```
/* Hard-coded sample data. Normally this would come from a real data
   source such as a database. */
private static Employee[] employees =
{
  new Employee("Bob Davidson", "123-45-6789"),
  new Employee("Mary Williams", "987-65-4321"),
  new Employee("Jim Smith", "111-11-1111"),
  new Employee("Beverly Harris", "222-22-2222"),
  new Employee("Thomas Frank", "333-33-3333"),
  new Employee("Jim Davidson", "444-44-4444")
};
```

The sample data is comprised of a few **Employee** objects. As you'll see in the next section, the **Employee** class is a simple class for encapsulating employee data.

The **searchByName( )** and **searchBySsNum( )** methods use the hard-coded data when performing a search. The **searchByName( )** method loops through each of the **Employee** objects in the **employees** array looking for any employees that match the name specified. If a match is found, it is added to the return **ArrayList** that will eventually be used by **search.jsp** to display the results. Note that the name search is case insensitive by virtue of uppercasing the **String**s before comparison. You should also note that the use of **String**'s **indexOf( )** method allows for partial matches instead of only exact matches.

Similar to the **searchByName( )** method, **searchBySsNum( )** loops through the hard-coded employee list looking for any employees that match the specified social security number. Note that **searchBySsNum( )** will capture only exact matches. Because social security numbers are unique to an individual, only one match should ever be returned for a social security number-based search.

Employee.java

The **Employee** class, shown next, is a simple bean for encapsulating the data for an employee. The class is straightforward, comprised simply of setters and getters for the **Employee** class data.

```
package com.jamesholmes.struts;

public class Employee
{
  private String name;
  private String ssNum;
  public Employee(String name, String ssNum) {
    this.name = name;
    this.ssNum = ssNum;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getName() {
    return name;
  }

  public void setSsNum(String ssNum) {
    this.ssNum = ssNum;
  }

  public String getSsNum() {
    return ssNum;
  }
}
```

This class is used by **EmployeeSearchService** for transferring employee search results data from the Model (**EmployeeSearchService**) to the View (**search.jsp**). Oftentimes, this 'transfer' object is referred to as a Data Transfer Object (DTO) or Value Object (VO) and has the simple responsibility of being a data container and abstracting the Model from the View.

web.xml

The **web.xml** file, shown next, is a standard Web Archive deployment descriptor used to configure the Mini HR application. Because the file contains several configuration details, it will be reviewed section by section.

```
<?xml version="1.0"?>

<!DOCTYPE web-app PUBLIC
  "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
  "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

  <!-- Action Servlet Configuration -->
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <!-- Action Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>

  <!-- The Welcome File List -->
  <welcome-file-list>
    <welcome-file>/index.jsp</welcome-file>
  </welcome-file-list>

  <!-- Struts Tag Library Descriptors -->
  <taglib>
    <taglib-uri>/WEB-INF/tlds/struts-bean.tld</taglib-uri>
    <taglib-location>/WEB-INF/tlds/struts-bean.tld</taglib-location>
  </taglib>
  <taglib>
    <taglib-uri>/WEB-INF/tlds/struts-html.tld</taglib-uri>
    <taglib-location>/WEB-INF/tlds/struts-html.tld</taglib-location>
  </taglib>
  <taglib>
    <taglib-uri>/WEB-INF/tlds/struts-logic.tld</taglib-uri>
    <taglib-location>/WEB-INF/tlds/struts-logic.tld</taglib-location>
  </taglib>

</web-app>
```

The following is the first section of the **web.xml** file. It declares the Struts Controller servlet, **ActionServlet**, and configures it.

```
<!-- Action Servlet Configuration -->
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

This declaration starts by assigning a name to the servlet that will be used in the next section for mapping the servlet to specific application requests. After defining the servlet's name and class, the **config** initialization parameter is defined. This parameter tells Strut's **ActionServlet**where to find its central configuration file: **struts-config.xml**. Finally, the <**load-on-startup**> tag is used to instruct the servlet engine how many instances of the servlet should be instantiated when the server starts.

The second section of the **web.xml** file causes **ActionServlet** to respond to certain URLs:

```
<!-- Action Servlet Mapping -->
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Notice that the <**servlet-name**> tag references the same name declared in the previous section. This associates the previous servlet declaration with this mapping. Next, the <**url-pattern**> tag is used to declare the URLs that **ActionServlet** will respond to. In this case, it is saying that**ActionServlet** should process any requests for pages that end in **.do**. So, for example, a request to
http://localhost:8080/MiniHR/page.do

or a request to
http://localhost:8080/MiniHR/dir1/dir2/page2.do
will be routed to Struts' **ActionServlet** for processing.
The next section of the **web.xml** file declares the Welcome File list that the Mini HR application will use:

```
<!-- The Welcome File List -->
<welcome-file-list>
  <welcome-file>/index.jsp</welcome-file>
</welcome-file-list>
```

The Welcome File list is a list of files that the Web server will attempt to respond with when a given request to the Web application goes unfulfilled. For example, in Mini HR's case, you can enter a URL of **http://localhost:8080/MiniHR/** and **index.jsp** will be executed, because no page has been specified in the URL. The servlet engine detects this and references the Welcome File list for pages that should be tried to respond to the request. In this case, the servlet engine will try to respond with a page at **/index.jsp**. If that page is unavailable, an error will be returned. Note that the Welcome File list can span several pages. In that case, the servlet engine will iterate through the list until a file is found that can be served for the request.
The final section of the **web.xml** file declares the JSP tag libraries that should be available to JSPs in the Mini HR application:

```
<!-- Struts Tag Library Descriptors -->
<taglib>
  <taglib-uri>/WEB-INF/tlds/struts-bean.tld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/struts-bean.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>/WEB-INF/tlds/struts-html.tld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/struts-html.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>/WEB-INF/tlds/struts-logic.tld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/struts-logic.tld</taglib-location>
</taglib>
```

A tag library definition associates a URI (or simple identifier) with the actual location of a ***.tld** file beneath a Web application, so essentially they are aliases. Using these aliases allows JSPs to reference an alias for a Tag Library Descriptor instead of the actual descriptor location. That way, the actual location of the tag library definitions can change without each JSP having to be changed as long as the aliases stay consistent.
Notice in the **web.xml** file that the URI (alias) and the actual location of the TLD files are the same. This is done for simplicity's sake and is a common practice. Note, despite the fact that the URL and location are the same, the definitions in **web.xm**l are necessary for JSPs to access the tag libraries.

struts-config.xml

The **struts-config.xml** file, shown next, is the central location for all of a Struts application's configuration settings. Remember from the previous description of the **web.xml** file that the **struts-config.xml** file is used by **ActionServlet** to configure the application. The basic configuration information is covered here, but a complete description will have to wait until you know more about Struts. (A complete discussion of configuration is found in Chapter 16.)

```
<?xml version="1.0"?>

<!DOCTYPE struts-config PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
  "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
```

```
<struts-config>

  <!-- Form Beans Configuration -->
  <form-beans>
    <form-bean name="searchForm"
               type="com.jamesholmes.minihr.SearchForm"/>
  </form-beans>

  <!-- Global Forwards Configuration -->
  <global-forwards>
    <forward name="search" path="/search.jsp"/>
  </global-forwards>

  <!-- Action Mappings Configuration -->
  <action-mappings>
    <action path="/search"
            type="com.jamesholmes.minihr.SearchAction"
            name="searchForm"
            scope="request"
            validate="true"
            input="/search.jsp">
    </action>
  </action-mappings>

  <!-- Message Resources Configuration -->
  <message-resources
    parameter="com.jamesholmes.minihr.ApplicationResources"/>

</struts-config>
```

Struts configuration files are XML-based and should conform to the Struts Configuration Document Type Definition (DTD). The **struts-config.xml** file just shown begins by declaring its use of the Struts Configuration DTD:

```
<!DOCTYPE struts-config PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
  "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
```

Next, is the Form Beans Configuration section, which is used to specify all of the **ActionForm** objects used in your Struts application. In this case, you're only using one Form Bean: **SearchForm**. The definition of the Form Bean, shown here, allows you to associate a name or alias of 'searchForm' with the **SearchForm** object:

```
<form-bean name="searchForm"
          type="com.jamesholmes.minihr.SearchForm"/>
```

That way, the application code (i.e., JSPs, **Action** objects, and so on) will reference 'searchForm' and not 'com.jamesholmes.minihr.SearchForm'. This allows the class definition to change without causing the code that uses the definition to change.

The next section of the file, Global Forwards Configuration, lists the forward definitions that your application will have. Forward definitions are a mechanism for assigning a name to the location of a page. For example, for the Mini HR application, you assign the name 'search' to the'search.jsp' page:

```
 <forward name="search" path="/search.jsp"/>
```

Similar to Form Beans, the use of forward definitions allows application code to reference an alias and not the location of pages. Note that this section of the file is dedicated to 'Global' forwards, which are made available to the entire Struts application. You can also specify action-specific forwards that are nested in an <**action**> tag in the config file:

```
<action ...>
  <forward .../>
</action>
```

The issue of action-specific forward definitions is examined later in this book.

After the Global Forwards Configuration section comes the Action Mappings Configuration section of the file. This section is used to define the **Action** classes used in your Struts application. Remember from the previous section on **SearchAction.java** that **Action** objects are used to handle discreet Controller tasks. Because the **SearchAction** mapping, shown here, has many settings, each is examined in detail.

```
<action path="/search"
        type="com.jamesholmes.minihr.SearchAction"
        name="searchForm"
        scope="request"
        validate="true"
        input="/search.jsp">
</action>
```

The first part of the Action Mappings Configuration section defines the path associated with this action. This path corresponds to the URL used to access your Struts application. Remember from the 'web.xml' section that your application is configured to have any URLs ending in **.do**be handled by **ActionServlet**. Setting the path to '/search' for this action essentially says that a request to '/search.do' should be handled by **SearchAction**. Struts removes the **.do** from the URL (resulting in '/search') and then looks in the **struts-config.xml** settings for an Action Mapping that corresponds to the URL.

The next <**action**> attribute, **type**, specifies the **Action** class that should be executed when the **path** is requested. The **name** attribute corresponds to the name of a Form Bean defined in the **struts-config.xml** file. In this case, 'searchForm' corresponds to the Form Bean you set up earlier. Using the **name** attribute tells Struts to populate the specified Form Bean with data from the incoming request. The **Action** object will then have access to the Form Bean to access the request data.

The next two attributes, **scope** and **validate**, are related to the Form Bean defined with the **name** attribute. The **scope** attribute sets the scope for the Form Bean associated with this action. For example, use 'request' for request scope or 'session' for session scope. The **validate**attribute is used to specify whether or not the Form Bean defined with the **name** attribute should have its **validate( )** method called after it has been populated with request data.

The final <**action**> attribute, **input**, is used to inform the **Action** object what page is being used to 'input' data to (or execute) the action; in this case, it is 'search.jsp'.

The last section of the file, Message Resources Configuration, is used to define the location of the **ApplicationResources.properties** file. Notice that the file is specified using Java's package mechanism: package.package.class (i.e., 'com.jamesholmes.minihr .ApplicationResources'). This allows **ActionServlet** to load the properties file from the same place that classes are loaded.

ApplicationResources.properties

The **ApplicationResources.properties** file, shown next, is based on Java's Resource Bundle functionality for externalizing and internationalizing application strings, messages, and labels.

```
# Label Resources
label.search.name=Name
label.search.ssNum=Social Security Number

# Error Resources
error.search.criteria.missing=<li>Search Criteria Missing</li>
error.search.ssNum.invalid=<li>Invalid Social Security Number</li>
errors.header=<font color="red"><b>Validation Error(s)</b></font><ul>
errors.footer=</ul><hr width="100%" size="1" noshade="true">
```

4. Explain about the struts tiles framework with neat sketch

Tiles in Struts helps to provide an easy and common look for Struts application. Struts Tiles is a user interface framework. The steps for creating a tiles application are explained below.

Tiles support the 2 different concepts.

- Reuse
- Developers can define a template for the web site and then use this layout to change the content of the web site.

Eg: – Consider we develop a web site having 1000 of pages static content and dynamically generate pages. If we want to change the layout of the web site we use the tiles framework to design template for the web site. This template helps to change the content of our web site. In future we want to change any layout we made a change only in one page and this change will affect whole pages in our web site.

We can specify the template in three different ways.

1. Define anonymously in a JSP page by using the tiles insert tag.
2. Define in a JSP using the tiles definition tag.
3. Define through XML file and loaded from a factory using the tiles insert tag.

Features of struts tiles

1) Screen definitions: –
   1. Create a screen by assembling tiles like header, footer, menu, body, etc.
   2. Definitions can take place :
      i. in a centralized XML file
      ii. directly in JSP pages
      iii. programmatically in Actions
   3. Definitions provide an inheritance mechanism: Definition can extend another one, and override parameters.
2) Simplify the creation of similar pages.
3) Layouts
   - ✓ Define common page layouts and reuse them across your website.
   - ✓ Define menu layouts, and use them by passing lists of items and links.
   - ✓ Define a portal layout; use it by passing a list of tiles (pages) to show.
   - ✓ Reuse existing layouts, or define our own.
4) Increase flexibility and ease of maintenance compared to <jsp:include…/>
5) Dynamic page building
   - ✓ Tiles can be gathered dynamically during page reload. It is possible to change any attribute: layout, list of tiles in portal, list of menu items, etc.
6) Reuse of tiles Components
   - ✓ If well defined, a tile can be reused across multiple applications.
   - ✓ Dynamic attributes are used to parameterize tiles.
   - ✓ It is possible to define a library of reusable tiles.
   - ✓ Build a page by assembling predefined components, giving them appropriate parameters.
   - ✓ It avoid the repetitions
7) Internationalization
   - ✓ It is possible to load different tiles according to Locale.
   - ✓ A mechanism similar to Java properties files is used for definitions files: you can have one definition file per Locale. The appropriate definition is loaded according to the current Locale.
8) Multi-channels
   - ✓ It is possible to load different Tiles according to a key.
   - ✓ For example, the key could represent user privileges, browser type, arbitrary name stored in session, etc.

- ✓ A mechanism similar to Java properties files is used for definitions files: you can have one definition file per key. The appropriate definition is loaded according to the key.

5. Explain the struts validation framework with neat diagram

In Struts,validation is provided by the Struts Validation framework.Struts Framework provides the functionality to validate the form data. It can be use to validate the data on the users browser as well as on the server side. Struts Framework emits the java scripts and it can be used to validate the form data on the client browser. Server side validation of the form can be accomplished by sub classing your From Bean with DynaValidatorForm class.

The Validator framework was developed by David Winterfeldt as third-party add-on to Struts. Now the Validator framework is a part of Jakarta Commons project and it can be used with or without Struts. The Validator framework comes integrated with the Struts Framework and can be used without doing any extra settings.

Using Validator Framework

Validator uses the XML file to pickup the validation rules to be applied to an form. In XML validation requirements are defined applied to a form. In case we need special validation rules not provided by the validator framework, we can plug in our own custom validations into Validator.

The Validator Framework uses two XML configuration files validator-rules.xml and validation.xml. Thevalidator-rules.xml defines the standard validation routines, these are reusable and used in validation.xml. to define the form specific validations. The validation.xml defines the validations applied to a form bean.

6. How struts and hibernate are integrated with suitable example

We can integrate any **struts application with hibernate**. There is no requirement of extra efforts.

Struts hibernate integration is a common integration mechanism of the struts application. Hibernate is one of the powerful Object-Oriented mapping tool that maps the object view of data into relational database for java and it includes efficient persistence services such as create, read, update and delete (CRUD). It also provides powerful, ultra-high performance OR persistence and Database query service for java web based applications. Normal strut hibernate integration architecture is given below.

Struts (Web page) <—> Spring DI <–> Hibernate (DAO) <—> Database

The applications starts with the blank struts framework and we need to add the Hibernate library files into them. The additional and the important thing we need is the integration Plugin that should be defined in the struts-config.xml file. Steps for implementing strut hibernate integration.

1) Create a new Hibernate Struts plug-in file to set the Hibernate session factory in servlet context, and include this file in struts-config.xml file.

2) In Struts, get the Hibernate session factory from servlet context, and do whatever Hibernate task you want.

1) Hibernate Struts Plug-in:- Create a Hibernate Struts Plug-in, to get the Hibernate session factory and store it into the servlet context for later user servlet.getServletContext().setAttribute(KEY_NAME, factory);

Features of Hibernate plug-in integration with Struts

1. Hibernate validator integration: – This feature allow the use of the fantastic Hibernate Validator Framework with very simple use of annotations in your Actions.
2. Hibernate core sessions injection capability: – This feature provides an Interceptor that open and sets in the value stack a Hibernate Core Session and closes (or not) this after the results are rendered for the client.
3. Hibernate transaction injection capability: – This feature provides an Interceptor that opens and sets in the value stack a Transaction and commit this after the results are rendered for the client if a rollback method was not invoked.

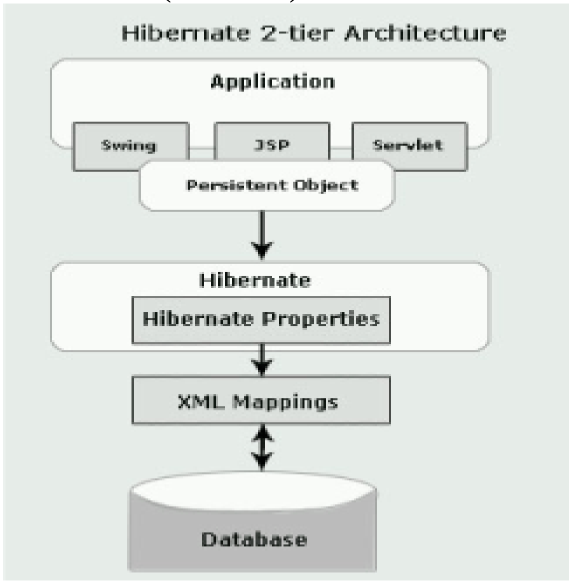
4. Hibernate core configuration management web tool: – This feature provides a web front-end tool for view and reload your Hibernate Core configurations.
5. Struts Spring Hibernate Integration is possible: – We can integrate spring and hibernate together. For this integration we need to integrate the following.

1) Integrate spring with Hibernate with spring's "LocalSessionFactoryBean" class.
2) Integrate spring with Struts via spring's ready make Struts plug-in – "ContextLoaderPlugIn".

7. Explain about the architecture of the hibernate with neat diagram

Hibernate Architecture

Unlike other technologies, Hibernate provides persistence as a service, rather than as a framework. It integrates flawlessly with various application architectures.
There are two common (recommended) architectures can be seen including Hibernate as a persistence layer.
The following diagram describes the Web (two tiered) Architecture of



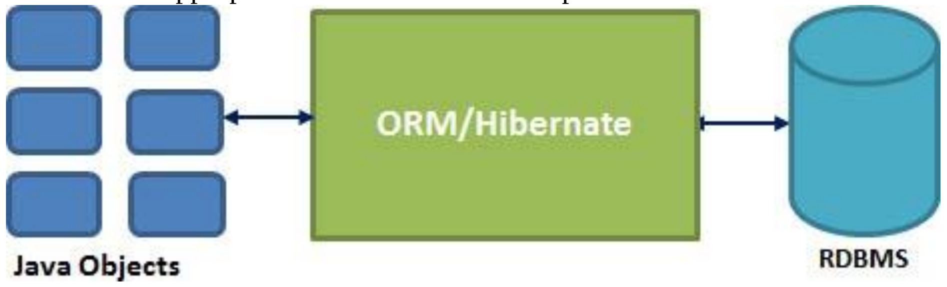
Hibernate: The above diagram shows that Hibernate is using **XML mapping** to configure data connectivity to database tables, and map classes for providing persistence services (and persistent objects) to the application.
To use Hibernate, it is required to create Java classes that represent the table in the database and then map the instance variable in the class with the columns in the database. Once the mapping is complete, various operations like **select, insert, update** and **delete** the records can be performed by the Hibernate on the table of database. Hibernate automatically creates the query to perform these operations. It may also be used to persist **JavaBeans** used by **servlets/JSPs** in Model View Controller (MVC) architecture.
Now, the following diagram describes the Enterprise (three-tiered) Architecture of Hibernate: The above diagram shows that Hibernate is using the database and configuration data to provide persistence services (and persistent objects) to the application.
To use Hibernate, it is required to create Java classes that represents the table in the database and then map the instance variable in the class with the columns in the database. Then Hibernate can be used to perform operations on the database like select, insert, update and delete the records in the table. Hibernate automatically creates the query to perform these operations.
Hibernate architecture has three main components:
Connection Management - Hibernate Connection management service provide efficient management of the database connections. Database connection is the most expensive part of interacting with the database as it requires a lot of resources of open and close the database connection.

Transaction management – Transaction management service provide the ability to the user to execute more than one database statements at a time.
Object relational mapping: Object relational mapping is technique of mapping the data representation from an object model to a relational data model. This part of the hibernate is used to select, insert, update and delete the records form the underlying table. When we pass an object to a **Session.save()** method, Hibernate reads the state of the variables of that object and executes the necessary query.

8. What is ORM? Explain about hibernate OR mapping in detail

Object-relational mapping (**ORM**, **O/RM**, and **O/R mapping**) in computer science is a programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. There are both free and commercial packages available that perform object-relational mapping, although some programmers opt to create their own ORM tools.

Hibernate is an Object-Relational Mapping(ORM) solution for JAVA and it raised as an open source persistent framework created by Gavin King in 2001. It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.
Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.
Hibernate sits between traditional Java objects and database server to handle all the work in persisting those objects based on the appropriate O/R mechanisms and patterns.



Hibernate Advantages:
- Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code.
- Provides simple APIs for storing and retrieving Java objects directly to and from the database.
- If there is change in Database or in any table then the only need to change XML file properties.
- Abstract away the unfamiliar SQL types and provide us to work around familiar Java Objects.
- Hibernate does not require an application server to operate.
- Manipulates Complex associations of objects of your database.
- Minimize database access with smart fetching strategies.
- Provides Simple querying of data.

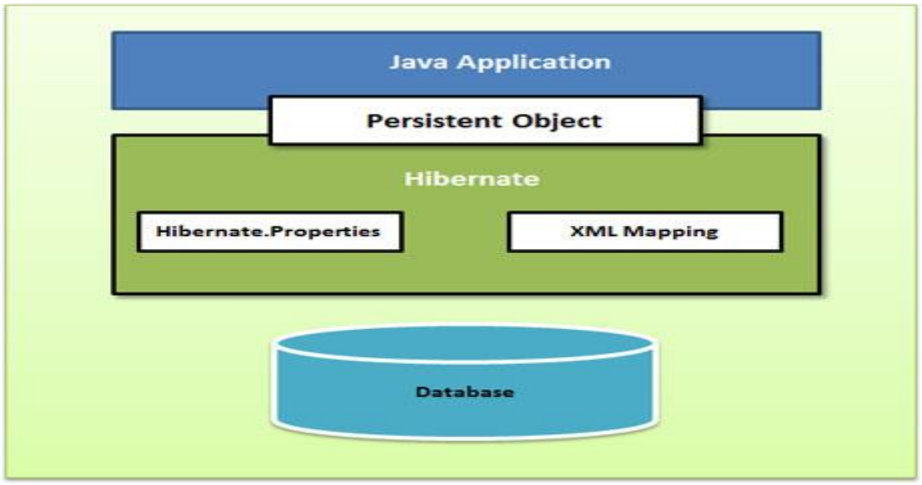
Supported Databases:
Hibernate supports almost all the major RDBMS. Following is list of few of the database engines supported by Hibernate.

- HSQL Database Engine
- DB2/NT
- MySQL
- PostgreSQL
- FrontBase
- Oracle
- Microsoft SQL Server Database
- Sybase SQL Server

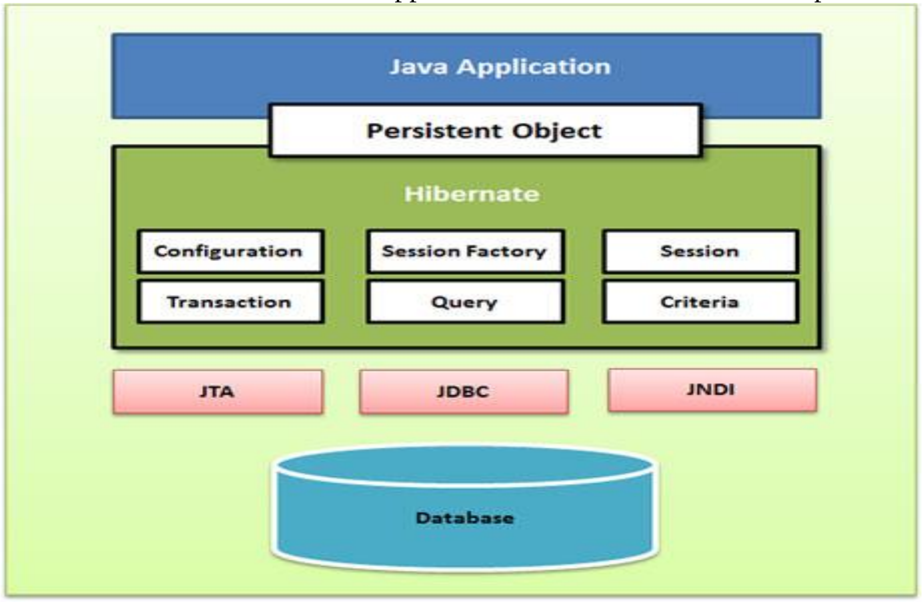
- Informix Dynamic Server

9. Explain about the core interfaces of hibernate framework.

The Hibernate architecture is layered to keep you isolated from having to know the underlying APIs. Hibernate makes use of the database and configuration data to provide persistence services (and persistent objects) to the application.



Following is a detailed view of the Hibernate Application Architecture with few important core classes.



Hibernate uses various existing Java APIs, like JDBC, Java Transaction API(JTA), and Java Naming and Directory Interface (JNDI). JDBC provides a rudimentary level of abstraction of functionality common to relational databases, allowing almost any database with a JDBC driver to be supported by Hibernate. JNDI and JTA allow Hibernate to be integrated with J2EE application servers.

Configuration Object:

The Configuration object is the first Hibernate object you create in any Hibernate application and usually created only once during application initialization. It represents a configuration or properties file required by the Hibernate. The Configuration object provides two keys components:

- **Database Connection:** This is handled through one or more configuration files supported by Hibernate. These files are **hibernate.properties** and**hibernate.cfg.xml**.
- **Class Mapping Setup**

This component creates the connection between the Java classes and database tables..

SessionFactory Object:

Configuration object is used to create a SessionFactory object which inturn configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all the threads of an application.

The SessionFactory is heavyweight object so usually it is created during application start up and kept for later use. You would need one SessionFactory object per database using a separate configuration file. So if you are using multiple databases then you would have to create multiple SessionFactory objects.

Session Object:

A Session is used to get a physical connection with a database. The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database. Persistent objects are saved and retrieved through a Session object.

The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed.

Transaction Object:

A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality. Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA).

This is an optional object and Hibernate applications may choose not to use this interface, instead managing transactions in their own application code.

Query Object:

Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects. A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.

Criteria Object:

Criteria object are used to create and execute object oriented criteria queries to retrieve objects.

UNIT –III
PART – B

1. Write down the steps how will you host a web site using LAMP

Create your Server

The first step to getting your site online is creating your VPS. You can find the information on how to launch your cloud server

Step Two—Basic Server Setup

After you have created your first server, you can start to secure it and create new users. Although this step is not required, nor are all the changes made in the tutorial necessary, it will begin to secure your server:

Step Three—Install LAMP

Subsequently, we need to start setting up the server programs. A strong foundation for the server is the traditional LAMP stack: Linux (automatically installed on the server), Apache, MySQL, and PHP, a web server, database server and programming language, respectively.

Prior to setting up our personal site, we need to be sure that LAMP Stack is installed on the server.

Step Four—Set Up Your Domain

In order to ensure that your site will be up and visible, you should set up your DNS to point your domain towards your new server. The basic setup is almost all automatic, requiring you to only enter your domain and the IP address of the server that you are looking to host your site on.

You can find more information on Setting Up a Hostname by following the link.

Step Five—Connect with SFTP

Now your site should be up and connected, and, once the records have propagated, you should be able to see the standard Apache, "It Works!" page by visiting your domain in the browser. If you would like to visit your site before the records have been updated, you can access it by typing your server's IP address in the browser. Now it is time to bring out the big guns: ie. Customizing your setup so that you can upload your own pages to your site. In order to do this, you need to install and configure an SFTP client.

Filezilla is a great program to use.

You can download Filezilla from their site.

Once it’s downloaded and installed, start it running.

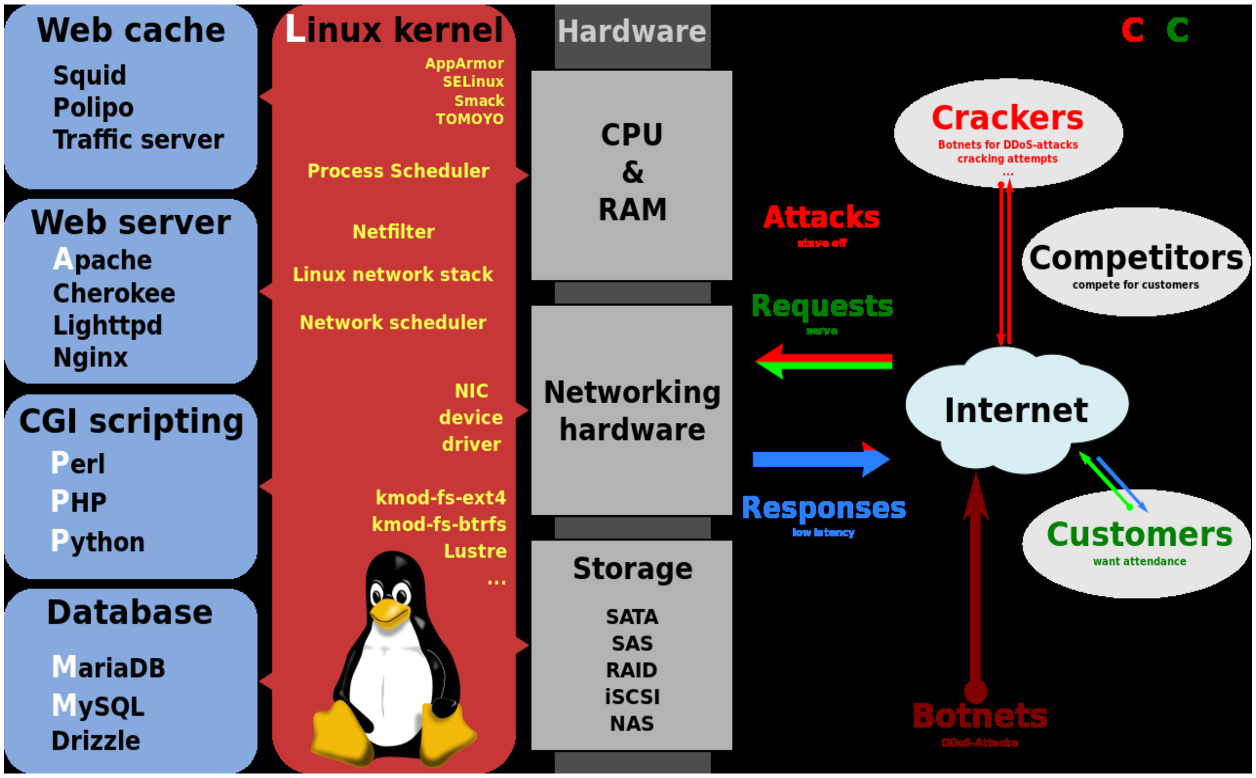
To begin transferring files, open up the Filezilla and within the file manager, open up the site manager options:

1. Click on the new site button. You can double click the name of the entry to change the name
2. Put your domain name (or IP address if your domain has not propagated yet) in the host field
3. Fill in your username and password with the username “root” and the password you set earlier
4. Click connect

Clicking connect will now allow you to transfer files from your computer to your server and build your site: be sure to drag them from your local directory on your left into your web directory (very likely /var/www) in frame on the right:

2. Define LAMP. Explain about the LAMP.

LAMP is an archetypal model of web service solution stacks, named as an acronym of the names of its original four components: the Linux operating system, the Apache HTTP Server, the MySQL relational database management system (RDBMS), and the PHP programming language. The LAMP components are largely interchangeable and not limited to the original selection. As a solution stack, LAMP is suitable for building dynamic web sites and web applications.



The LAMP platform consists of four components that are structured in a layered way. Each layer provides a critical part of the entire software stack:

Linux. Linux is the lowest-level layer and provides the operating system. Linux actually runs each of the other components. You are not specifically limited to Linux, however; you can easily run each of the other components on Microsoft®; Windows®;, Mac OS X, or UNIX® if you need to.

Apache. The next layer is Apache, the Web server. Apache provides the mechanics for getting a Web page to a user. Apache is a stable, mission-critical-capable server, and it runs more than 65 percent of all Web sites on the Internet. The PHP component actually sits inside Apache, and you use Apache and PHP together to create your dynamic pages.

MySQL. MySQL provides the data-storage side of the LAMP system. With MySQL, you have access to a very capable database suitable for running large and complex sites. Within your Web application, all your data, products, accounts, and other types of information will reside in this database in a format that you can easily query with the SQL language.

PHP. PHP is a simple and efficient programming language that provides the glue for all the other parts of the LAMP system. You use PHP to write dynamic content capable of accessing the data in the MySQL database and some of the features that Linux provides.

3. Explain about Python data structures with example

List:

The list data type has some more methods. Here are all of the methods of list objects:
list.append(*x*): Add an item to the end of the list; equivalent to a[len(a):] = [x].
list.extend(*L*): Extend the list by appending all the items in the given list; equivalent to a[len(a):] = L.
list.insert(*i*, *x*): Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list, and a.insert(len(a), x) is equivalent to a.append(x).
list.remove(*x*): Remove the first item from the list whose value is *x*. It is an error if there is no such item.
list.pop([*i*]): Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the *i* in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)
list.index(*x*): Return the index in the list of the first item whose value is *x*. It is an error if there is no such item.
list.count(*x*): Return the number of times *x* appears in the list.
list.sort(*cmp=None*, *key=None*, *reverse=False*): Sort the items of the list in place (the arguments can be used for sort customization, see sorted() for their explanation).
list.reverse(): Reverse the elements of the list, in place.
Tuples and sequence:
They are two examples of *sequence* data types (Sequence Types — str, unicode, list, tuple, bytearray, buffer, xrange). Since Python is an evolving language, other sequence data types may be added. There is also another standard sequence data type: the *tuple*.
A tuple consists of a number of values separated by commas, for instance:

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
>>> # Tuples are immutable:
... t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> # but they can contain mutable objects:
... v = ([1, 2, 3], [3, 2, 1])
>>> v
([1, 2, 3], [3, 2, 1])
```

Sets
Python also includes a data type for *sets*. A set is an unordered collection with no duplicate elements. Basic uses include membership testing and eliminating duplicate entries. Set objects also support mathematical operations like union, intersection, difference, and symmetric difference.
Curly braces or the set() function can be used to create sets. Note: to create an empty set you have to use set(), not {}; the latter creates an empty dictionary, a data structure that we discuss in the next section.
Here is a brief demonstration:

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> fruit = set(basket)              # create a set without duplicates
>>> fruit
set(['orange', 'pear', 'apple', 'banana'])
```

```
>>> 'orange' in fruit                 # fast membership testing
True
>>> 'crabgrass' in fruit
False

>>> # Demonstrate set operations on unique letters from two words
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                              # unique letters in a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b                           # letters in a but not in b
set(['r', 'd', 'b'])
>>> a | b                           # letters in either a or b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b                            # letters in both a and b
set(['a', 'c'])
>>> a ^ b                           # letters in a or b but not both
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

Dictionaries

Dictionaries are sometimes found in other languages as “associative memories” or “associative arrays”. Unlike sequences, which are indexed by a range of numbers, dictionaries are indexed by *keys*, which can be any immutable type; strings and numbers can always be keys. Tuples can be used as keys if they contain only strings, numbers, or tuples; if a tuple contains any mutable object either directly or indirectly, it cannot be used as a key. You can’t use lists as keys, since lists can be modified in place using index assignments, slice assignments, or methods like append() and extend().

It is best to think of a dictionary as an unordered set of *key: value* pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}. Placing a comma-separated list of key:value pairs within the braces adds initial key:value pairs to the dictionary; this is also the way dictionaries are written on output.

The main operations on a dictionary are storing a value with some key and extracting the value given the key. It is also possible to delete a key:value pair with del. If you store using a key that is already in use, the old value associated with that key is forgotten. It is an error to extract a value using a non-existent key.

The keys() method of a dictionary object returns a list of all the keys used in the dictionary, in arbitrary order (if you want it sorted, just apply the sorted() function to it). To check whether a single key is in the dictionary, use the in keyword.

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}
>>> tel.keys()
['guido', 'irv', 'jack']
>>> 'guido' in tel
True
```

4. Explain about the exceptions and errors in python

Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them −

Exception Handling: This would be covered in this tutorial. Here is a list standard Exceptions available in Python: Standard Exceptions.

Assertions: This would be covered in Assertions in Python tutorial.

ist of Standard Exceptions −

EXCEPTION NAME	**DESCRIPTION**
Exception	Base class for all exceptions
StopIteration	Raised when the next() method of an iterator does not point to any object.
SystemExit	Raised by the sys.exit() function.
StandardError	Base class for all built-in exceptions except StopIteration and SystemExit.
ArithmeticError	Base class for all errors that occur for numeric calculation.
OverflowError	Raised when a calculation exceeds maximum limit for a numeric type.
FloatingPointError	Raised when a floating point calculation fails.
ZeroDivisonError	Raised when division or modulo by zero takes place for all numeric types.

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.
When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

Handling an exception

If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the try: block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

Syntax
Here is simple syntax of *try....except...else* blocks −

```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

Here are few important points about the above-mentioned syntax −

- A single try statement can have multiple except statements. This is useful when the try block contains statements that may throw different types of exceptions.
- You can also provide a generic except clause, which handles any exception.
- After the except clause(s), you can include an else-clause. The code in the else-block executes if the code in the try: block does not raise an exception.
- The else-block is a good place for code that does not need the try: block's protection.

Example
This example opens a file, writes content in the, file and comes out gracefully because there is no problem at all −

```
#!/usr/bin/python

try:
   fh = open("testfile", "w")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
else:
   print "Written content in the file successfully"
   fh.close()
```

This produces the following result −

```
Written content in the file successfully
```

Example
This example tries to open a file where you do not have write permission, so it raises an exception −

```
#!/usr/bin/python

try:
   fh = open("testfile", "r")
   fh.write("This is my test file for exception handling!!")
except IOError:
   print "Error: can\'t find file or read data"
else:
   print "Written content in the file successfully"
```

This produces the following result −

```
Error: can't find file or read data
```

The except Clause with No Exceptions

You can also use the except statement with no exceptions defined as follows −

```
try:
   You do your operations here;
   ......................
except:
   If there is any exception, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

This kind of a **try-except** statement catches all the exceptions that occur. Using this kind of try-except statement is not considered a good programming practice though, because it catches all exceptions but does not make the programmer identify the root cause of the problem that may occur.

The except Clause with Multiple Exceptions

You can also use the same *except* statement to handle multiple exceptions as follows −

```
try:
   You do your operations here;
   ......................
except(Exception1[, Exception2[,...ExceptionN]]]):
   If there is any exception from the given exception list,
   then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

The try-finally Clause

You can use a **finally:** block along with a **try:** block. The finally block is a place to put any code that must execute, whether the try-block raised an exception or not. The syntax of the try-finally statement is this −

```
try:
   You do your operations here;
   ......................
   Due to any exception, this may be skipped.
finally:
   This would always be executed.
   ......................
```

Note that you can provide except clause(s), or a finally clause, but not both. You cannot use *else* clause as well along with a finally clause.

Example

```
#!/usr/bin/python

try:
   fh = open("testfile", "w")
   fh.write("This is my test file for exception handling!!")
finally:
   print "Error: can\'t find file or read data"
```

If you do not have permission to open the file in writing mode, then this will produce the following result:

```
Error: can't find file or read data
```

Same example can be written more cleanly as follows −

```
#!/usr/bin/python

try:
   fh = open("testfile", "w")
   try:
      fh.write("This is my test file for exception handling!!")
   finally:
      print "Going to close the file"
      fh.close()
```

```
except IOError:
   print "Error: can\'t find file or read data"
```

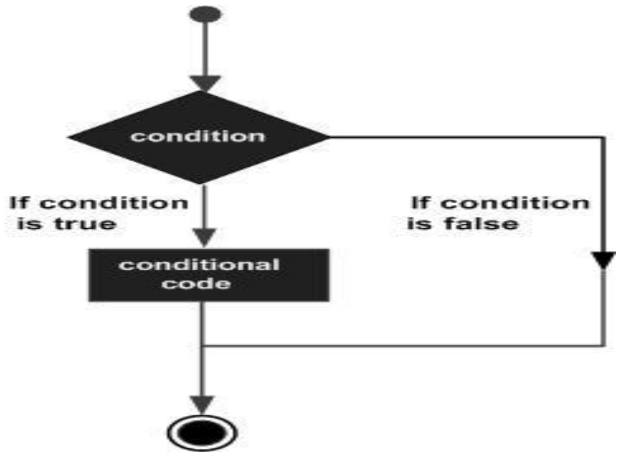
When an exception is thrown in the *try* block, the execution immediately passes to the *finally* block. After all the statements in the *finally* block are executed, the exception is raised again and is handled in the *except* statements if present in the next higher layer of the *try-except* statement.

5. Explain about the control flow statements in python

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.

Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.

Following is the general form of a typical decision making structure found in most of the programming languages −



Python programming language assumes any **non-zero** and **non-null** values as TRUE, and if it is either **zero** or **null**, then it is assumed as FALSE value.

Python programming language provides following types of decision making statements. Click the following links to check their detail.

Statement	**Description**
if statements	An **if statement** consists of a boolean expression followed by one or more statements.
if...else statements	An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE.
nested if statements	You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).

Let us go through each decision making briefly −

Single Statement Suites

If the suite of an **if** clause consists only of a single line, it may go on the same line as the header statement.

Here is an example of a **one-line if** clause −

```
#!/usr/bin/python

var = 100

if ( var  == 100 ) : print "Value of expression is 100"

print "Good bye!"
```

When the above code is executed, it produces the following result −

```
Value of expression is 100
Good bye!
```

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Python supports the following control statements. Click the following links to check their detail.

Control Statement	**Description**
break statement	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
pass statement	The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

6. List and explain the steps how to compose the web page in python

To make an initial very simple "Hello World" web application and project directory using lpthw.web. First, make your project directory:

```
$ cd projects
$ mkdir gothonweb
$ cd gothonweb
$ mkdir bin gothonweb tests docs templates
$ touch gothonweb/__init__.py
$ touch tests/__init__.py
```

You'll be taking the game from Exercise 43 and making it into a web application, so that's why you're calling it gothonweb. Before you do that, we need to create the most basic lpthw.web application possible. Put the following code into bin/app.py:

```
import web

urls = (
  '/', 'index'
)

app = web.application(urls, globals())

class index:
    def GET(self):
        greeting = "Hello World"
        return greeting

if __name__ == "__main__":
    app.run()
```

Then run the application like this:

```
$ python bin/app.py
http://0.0.0.0:8080/
```

However, if you did this:

```
$ cd bin/   # WRONG! WRONG! WRONG!
$ python app.py  # WRONG! WRONG! WRONG!
```

Then you are doing it *wrong*. In all Python projects you do not cd into a lower directory to run things. You stay at the top and run everything from there so that all of the system can access all the modules and files. Go reread Exercise 46 to understand a project layout and how to use it if you did this.

Finally, use your web browser and go to http://localhost:8080/ and you should see two things. First, in your browser you'll see Hello, world!. Second, you'll see your Terminal with new output like this:

```
$ python bin/app.py
http://0.0.0.0:8080/
127.0.0.1:59542 - - [13/Jun/2011 11:44:43] "HTTP/1.1 GET /" - 200 OK
127.0.0.1:59542 - - [13/Jun/2011 11:44:43] "HTTP/1.1 GET /favicon.ico" - 404 Not Found
```

Those are log messages that lpthw.web prints out so you can see that the server is working, and what the browser is doing behind the scenes. The log messages help you debug and figure out when you have problems. For example, it's saying that your browser tried to get /favicon.ico but that file didn't exist so it returned 404 Not Found status code.

Create Basic Templates

Break your lpthw.web application, but did you notice that "Hello World" isn't a very good HTML page? This is a web application, and as such it needs a proper HTML response. To do that you will create a simple template that says "Hello World" in a big green font.

The first step is to create a templates/index.html file that looks like this:

```
$def with (greeting)

<html>
    <head>
        <title>Gothons Of Planet Percal #25</title>
    </head>
<body>

$if greeting:
    I just wanted to say <em style="color: green; font-size: 2em;">$greeting</em>.
$else:
    <em>Hello</em>, world!

</body>
</html>
```

If you know what HTML is, then this should look fairly familiar. If not, research HTML and try writing a few web pages by hand so you know how it works. This HTML file, however, is a *template*, which means that lpthw.web will fill in "holes" in the text depending on variables you pass in to the template. Every place you see $greeting will be a variable you'll pass to the template that alters its contents.

To make your bin/app.py do this, you need to add some code to tell lpthw.web where to load the template and to render it. Take that file and change it like this:

```
import web

urls = (
  '/', 'Index'
)

app = web.application(urls, globals())

render = web.template.render('templates/')

class Index(object):
    def GET(self):
        greeting = "Hello World"
        return render.index(greeting = greeting)

if __name__ == "__main__":
    app.run()
```

Pay close attention to the new render variable and how I changed the last line of index.GET so it returns render.index() passing in your greeting variable.

Once you have that in place, reload the web page in your browser and you should see a different message in green. You should also be able to do a View Source on the page in your browser to see that it is valid HTML.

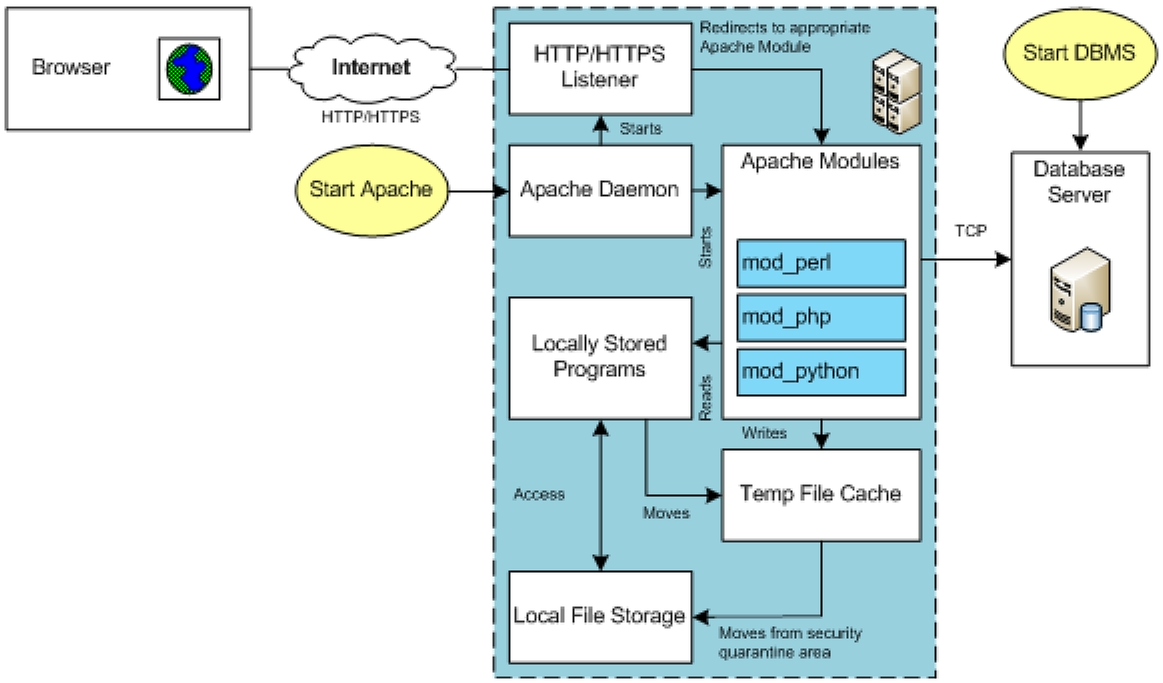
This may have flown by you very fast, so let me explain how a template works:

1. In your bin/app.py you've added a new variable, render, which is a web.template.render object.
2. This render object knows how to load .html files out of the templates/ directory because you passed that to it as a parameter.
3. Later in your code, when the browser hits the index.GET like before, instead of just returning the string greeting, you call render.index and pass the greeting to it as a variable.
4. This render.index method is kind of a *magic* function where the render object sees that you're asking for index, goes into the templates/ directory, looks for a page named index.html, and then "renders" it, or converts it.

5. In the templates/index.html file you see the beginning definition that says this template takes a greeting parameter, just like a function. Also, just like Python this template is indentation sensitive, so make sure you get them right.
6. Finally, you have the HTML in templates/index.html that looks at the greeting variable and, if it's there, prints one message using the $greeting, or a default message.

To get deeper into this, change the greeting variable and the HTML to see what effect it has. Also create another template named templates/foo.html and render that using render.foo() instead of render.index() like before. This will show you how the name of the function you call on render is just matched to an .html file in templates/.

7. Explain about the LAMP architecture with neat diagram



LAMP (Linux, Apache, MySQL, Perl/PHP/Python) Architecture is very flexible. All the components can be positioned on the same server or different servers. The servers are divided into two types. The types are known as the Application or database tiers. Generally, the application tier holds the Apache Server, any Apache Modules, and local copies of Server Side Includes (SSI) programs.
In many development environments, you also deploy the client to the same machine. This means a single machine runs the database server, the application server, and the browser. The lab for this section assumes these configurations.

Before you test an installation, you should make sure that you've started the database and Apache server. In an Oracle LAMP configuration (known as an OLAP – Oracle, Linux, Apache, Perl/PHP/Python), you must start both the Oracle Listener and database. MySQL starts the listener when you start the database. You must also start the Apache Server. The Apache Server also starts an Apache Listener, which listens for incoming HTTP/HTTPS requests. It listens on Port 80 unless you override that setting in the httpd.conf file.

The URI reaches the server and is redirected to an Apache Module based on configuration information found in the httpd.conf file. Spawned or child processes of the Apache Module then read programs into memory from the file system and run them. If you've uploaded a file the locally stored program can move it from a secure cache location to another local area for processing. The started programs can run independently or include other files as libraries, and they can communicate to the database server.

8. Explain the features of Lamp Stack

Benefits of LAMP

The benefits of the LAMP stack are illustrated by the number of proponents of this system. Benefits of using LAMP include:

- **Easy to code**: Ask all developers and they will tell you that coding is a breeze on LAMP. What this is does is that it ensures that coding is relatively bug free and doesn't have to go through an exhaustive and time consuming process of fixing the bugs.
- **Easy deployment**: For many developers, it's the deployment of a web application that becomes a tricky exercise; especially when the programming language cannot be easily integrated with the server and database. But, there are no such problems with LAMP as PHP is a standard Apache module. This makes it easier to deploy LAMP web applications.
- **Local Development** – Another huge advantage of using LAMP is that a developer can build an app locally and then deployed it onto the web.

LAMP is definitely popular, but not all developers are able to optimize its use. You need to be able to choose the kind of developer who is both an expert and experienced, on using the LAMP stack.

The team of highly proficient developers at PLAVEB can use LAMP in a way such that its use is optimized to benefit client applications. Our **web application development** team has successfully worked on diverse web applications that have been developed with the help of this particular software bundle. This helps us cater to all kinds of client requirements and ensure that our web applications meet the highest standards of quality, functionality and all-round efficiency.

Specific solutions are required for web sites that serve large numbers of requests, or provide services that demand high uptime. High-availability approaches for the LAMP stack may involve multiple web and database servers, combined with additional components that perform logical aggregation of resources provided by each of the servers, as well as distribution of the workload across multiple servers. The aggregation of web servers may be provided by placing a load balancer in front of them, for example by using Linux Virtual Server (LVS). For the aggregation of database servers, MySQL provides internal replication mechanisms, which implement a master/slave relationship between the original database (master) and its copies (slaves).

Such high-availability setups can improve the availability of LAMP instances by providing various forms of redundancy, making it possible for a certain number of an instance's components (separate servers) to experience downtime without interrupting the users of services provided by a LAMP instance. Also, such redundant setups allow for hardware failures resulting in data loss on individual servers, without the collectively stored data actually becoming lost. Beside higher availability, such LAMP setups can provide almost linear improvements in performance for services where the number of internal database read operations is much higher than the number of write/update operations

9. Explain about the PHP arrays and control structures with example

Control Structures are loops, like **for**, and conditionals, such as **if**. Most of these will be familiar to the javascript programmer, and this page is basically a list. The following exist in php:

- if and else
- elseif
- while
- do...while
- for
- foreach
- break and continue
- switch

if and else

These are exactly the same as javascript and other languages.

```
if ($a==$b) {
print "The same";
}
else {
print "They aren't the same";
}
```

There is an alternative way of writing the if statement. For instance:

```
<script type="text/javascript">
<?php
$a=5;
if($a==5):?>alert("It's five!");
<?php
endif;
?>
//result: alert saying "It's five"
</script>
```

The condition, **if ($a==5)** is followed by a colon (**:**), and the end of the block of statements is followed by **endif** (in php tags). In this way, the statement is not php, and in this case it is javascript.

In the alternative styles, instead of having two curly brackets **{}** the code replaces the first with a colon and the last with an **end...** in the case of if, it is **endif**. In other cases, **endwhile**, **endfor**, etc.

If there are other conditions, such as **ifelse**, then this new condition acts as the closing curly bracket for the if-statement, and the the endif, at the end of the block of code, acts as the final bracket. For instance:

```
if($a==5): //note colon
print"a equals 5";
print"...";
elseif($a==6): // the elseif effectively ends the if-statement and replaces a }

//note the colon after ifelse(...)!

print"a equals 6";

print"!!!";
else : //This acts as the end bracket } for the elseif

print"a is neither 5 nor 6";

endif; // The endif acts as the final curly bracket for the BLOCK of statements
```

Below is a more complex example:

```
<script type="text/javascript">
<?php
$a=6;
if($a==5): //note colon
?>
document.write("It's five!") ;
<?php
elseif($a==6): //colon!
?>
document.write("It's six!");
<?php
else:
?>
document.write("It's neither six nor five");
<?php
endif;
//result: It's six!
?>
</script>
```

Fundamentally, you can put **elseif** within the if block between the colon (:) and **endif;**

elseif

elseif, which you can also write **else if**, gives you a second if-statement. The first if-condition, if met, is run, and any other elseif conditions, even if met are not run. So in the example below, the if-statement is met, so the program outputs **Hello Tom**, and not **Hello Tom 2**, even though the relevant elseif condition has been met. That is, elseif acts like a switch, and the control structure is exited at the first match.

```
$a="Tom";
if($a=="Tom"){
print"Hello Tom";}
elseif($a=="Beryl"){
print"Hello Beryl";
}
else if ($a=="Tom"){
print "Hello Tom 2";
}
else {
print"I don't know you!";
}
//result: Hello Tom
```

while

The while loop is the same as in javascript:

```
$i=1;
while ($i<10){
print "Hello";
$i++;
}
```

Ensure the while loop will exit! Forgetting $i++ means it will go on until the browser gets bored!

The next example uses the alternative style:

```
<?php
$i=1;
while($i<=10): //note the colon!
?>
```

```
Hello
<?php
$i++;
endwhile; // tells php the while is done
//result: HelloHelloHelloHelloHelloHelloHelloHelloHelloHello
?>
```

In the alternative style, the first bracket **{** is replaced with a colon: the final curly bracket **}** is replaced with **endwhile**.

do..while

The do..while loop is similar to the while loop, however, the while loop will not run if the condition is not met. The do..while loop, on the contrary, will always run at least once.

In the example below, the do..while loop runs once, even though the condition is never met (1 isn't bigger than 100):

```
$i=0;
do{
print "The number is $i";
}
while($i>100);
//result: The number is 0
```

for

The for loop is the same as javascript's:

```
for($i=1;$i<=2;$i++){
print "This is $i<br>";
}
//This is 1
//This is 2
```

The for loop can be written in various ways. One additional way is as follows:

```
for($i=1;$i<=3;print "This time it's $i ",$i++);
//result: This time it's 1 This time it's 2 This time it's 3
```

Quite neat isn't it?

foreach

This is used to access the items in an array. For instance:

```
$arr=array("cat","dog","aadvark");
foreach($arr as $value){
echo"Value: $value<br>\n";
//Value: cat

//Value: dog

//Value: aadvark
```

This example shows the keys and values of an associative array:

```
$a=array("fruit"=>"apple","meat"=>"ardvaak","vegetables"=>"potatoes","sweet"=>"pecan pie");
foreach($a as $k=>$v){
print"\$a[$k]=>$v.\n";
};
//result: $a[fruit]=>apple. $a[meat]=>ardvaak. $a[vegetables]=>potatoes. $a[sweet]=>pecan pie.
```

break and continue

Break and continue are familiar to javascript programmers. Break exits a for, while, or switch structure. And continue exits the current pass in the loop and begins with the next. You can determine how many loops to break out of, or to continue from.

switch

The switch statement is also the same as the one in javascript. A switch statement will continue to, in the example below, print all the statements after finding one that is true, so breaks are used to prevent this. Each statement is followed by a break, usually.

```
$i=1;
switch($i){ //the variable is in brackets and the cases are in curly brackets
case 0:
print "zero";
break;
case 1:
print "one";
break;
default: //You can do a default case where the variable is something else!
print "whatever it is, it isn't 0, 1 or 2";
}
//result: one
```

The switch statement can deal with any **simple type of data such as integers, floats and strings**. This next example is the same as before in structure, except it uses strings:

```
$i="tomatoes";
switch ($i){
case "potatoes":
print "spuds";
break;
case "tomatoes":
print "toms";
break;
default:
print "Well, it could be anything!";
}
//result: toms
```

Arrays

Getting information about a variable (print_r)

You can print to the screen information about a variable using print_r (r for readable by humans). With strings and numbers, it does the same as echo() and print(). With arrays it prints out the items.

For example:

```
$str="Hello, it's me!<br>";
print_r($str);
//Hello, it's me!
$furniture=array("chair","bed","carpet");
print_r($furniture);
//Array ( [0] => chair [1] => bed [2] => carpet )
```

Using **print_r** we can easily see what is in an array.

Making an array from a string (explode)

We can convert a string of items separated by a string into an array by using explode() as follows:

```
$animals = "cat dog elephant zebra";
$animals = explode(" ", $animals);
print_r($animals);
//Array ( [0] => cat [1] => dog [2] => elephant [3] => zebra )
```

We make the items in a string separated by a space into an array, and we demonstrate this has occurred by using print_r.

Making a string from an array (implode())

We can do the opposite to making an array, and make an array into a string using **implode()**:

```
$animals=implode(" and ",$animals);
print_r($animals);
echo "<br>";
//cat and dog and elephant and zebra
```

The first parameter of **implode** is the separator, a string, and the next one is the array. We used **print_r()** to display the string, with its repetitive and's.

Getting a list of variables from an array (list())

You can quickly assign elements in an array to a list of variables using list():

```
$animals=array("chair","bed","carpet");
list($a,$b,$c)=$animals;
echo "The big brown $a ate the furry $b instead of the $c<br>";
//The big brown chair ate the furry bed instead of the carpet
```

In the example, the variables $a, $b, $c, have the elements of the array $animals assigned to them, so writing $a becomes equivalent to writing "chair", or $animals[0].

Turning a string into an array using split()

split() is similar to explode() except it uses regular expressions. It is better to use explode() because explode uses fewer resources. The example shows its power:

```
$animals=split("[/.?]","cat.dog?pig/parrot");
print_r($animals)."<br>";
//Array ( [0] => cat [1] => dog [2] => pig [3] => parrot )
```

In the example split uses any of the separators **[/.?]** to separate the elements for the array. The next piece of code shows how a number can be added to split(), to limit the number of items.

```
$passwd_line="username:secretword:123:somethingelse";
list($user, $pass, $uid) =
split(":", $passwd_line, 3);
echo "\$user is: ".$user."<br>";
//$user is: username
```

Although there are 4 items in the string, we only use 3 of them, by using the optional third parameter of **split()**.

Accessing elements with foreach

This is the lazy persons way of accessing the elements of an array:

```
$arr = array("spades","hearts","diamonds","clubs");
foreach ($arr as $value) { // $value is any variable name
//the each value of the $arr is assigned to $value (or other name)
echo "Value = $value <br>";
}
/*Value = spades
Value = hearts
Value = diamonds
Value = clubs */
```

The foreach loop doesn't need a counter (as a for loop does). The basic syntax is:

```
foreach ($arr as $anything)
{
// action
}
```

Naturally, we can do also sorts of things where it says "action" above. For instance:

```
$arr = array("spades","hearts","diamonds","clubs");
foreach ($arr as $tmp) {
$value="The ".$tmp;
echo "Value = $tmp <br>";
}
/*Value = The spades
Value = The hearts
Value = The diamonds
Value = The clubs */
```

10. Create a PHP application that connects with the databases with suitable example

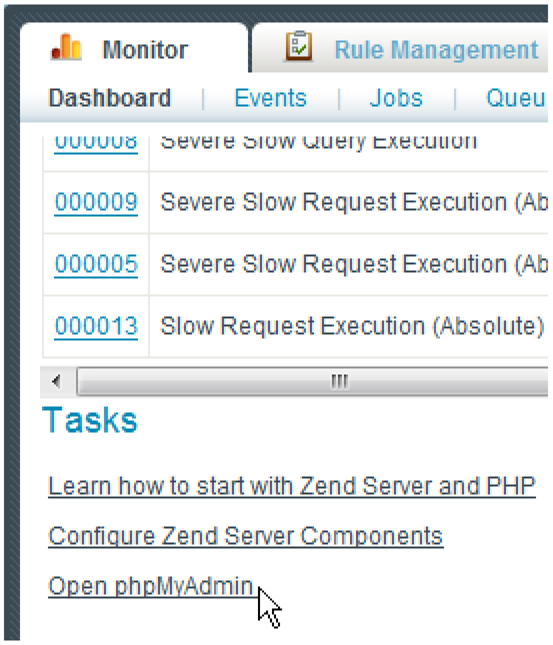
Step 1: Create a MySQL Databse Using phpMyAdmin

phpMyAdmin is a component that comes bundled with Zend Server in order to handle the administration of your MySQL databases.

Creating a database in MySQL is necessary in order to debug your Flex/PHP application.

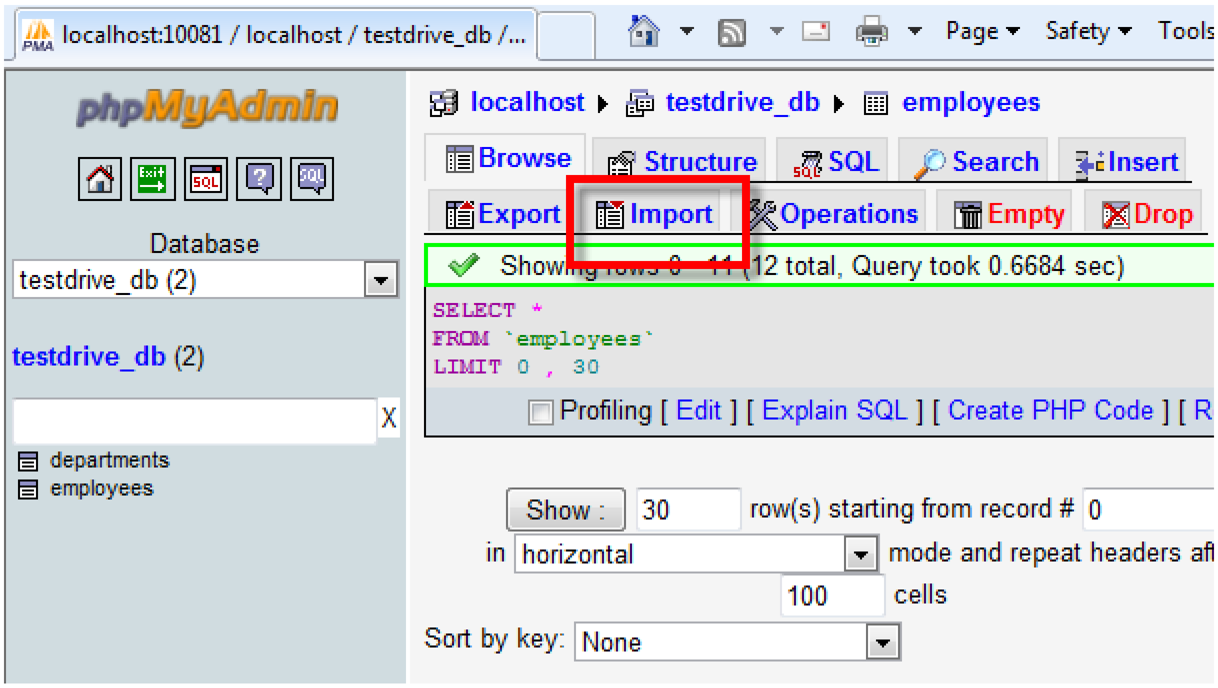
The PHP code in your project will use the database to access the information when needed, as defined by the code.

You will need the test_db.sql file located in the Database folder of the Building_A_PHP_App.zip file.
Open your Zend Server UI by entering "http://localhost:10081" into a browser, and enter your username and password.
If this is your first time opening Zend Server see Getting Started with Zend Server.
Go to **Monitor | Dashboard** and in the Tasks area click **Open phpMyAdmin**.


Monitor
Rule Management
Dashboard | Events | Jobs | Queu
000008 Severe Slow Query Execution
000009 Severe Slow Request Execution (Ab
000005 Severe Slow Request Execution (Ab
000013 Slow Request Execution (Absolute)
Tasks
Learn how to start with Zend Server and PHP
Configure Zend Server Components
Open phpMyAdmin

phpMyAdmin opens in a new browser.

4. Enter the username "root" and leave the password empty (as defined when setting up your phpMyAdmin during Zend Server installation) and click **Go**.



Showing rows 0 - 11 (12 total, Query took 0.6684 sec)

```
SELECT *
FROM `employees`
LIMIT 0 , 30
```

Profiling [Edit] [Explain SQL] [Create PHP Code] [R

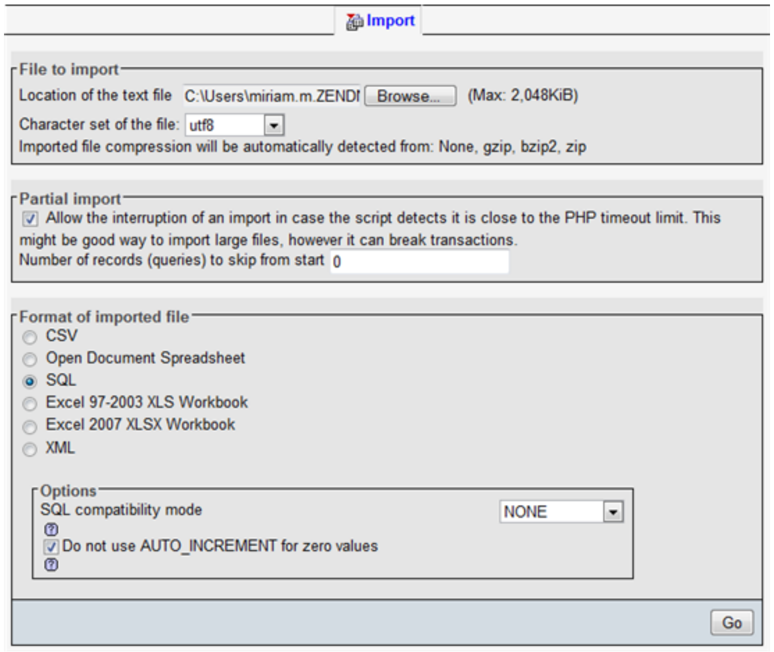
Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers af
100 cells
Sort by key: None

Go to the Import tab, browse to the test_db.sql file and click **Go**.

Figure 4: Import tab

The database has been created, and is already located on your machine.

To see the tables located in your database, select the database from the left menu of the phpMyAdmin UI and go to the Structure tab.



File to import
Location of the text file: C:\Users\miriam.m.ZENDI (Max: 2,048KiB)
Character set of the file: utf8
Imported file compression will be automatically detected from: None, gzip, bzip2, zip

Partial import
[✓] Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. This might be good way to import large files, however it can break transactions.
Number of records (queries) to skip from start: 0

Format of imported file
- CSV
- Open Document Spreadsheet
- [✓ SQL]
- Excel 97-2003 XLS Workbook
- Excel 2007 XLSX Workbook
- XML

Options
SQL compatibility mode: NONE
[✓] Do not use AUTO_INCREMENT for zero values

PART - B UNIT – IV

1. Describe the characteristics of C#. How does C# differ from C++ ?

Characteristics of C#

- Simple
- Consistent
- Modern
- Object-oriented
- Type-safe
- Versionable
- Compatible
- Interoperable
- Flexible

C# differ from C++

- C# compiles straight from source code to executable code, with no object files.
- C# does not separate class definition from implementation and therefore no need for header files.
- Class definition does not use a semicolon at the end.
- The first character of the Main() function is capitalized. The Main must return either int or void type value.
- C# does not provide #include statement.
- All data types in C# are inherited from the object super class and therefore they are objects.
- All the basic value types will have the same size on any system. This is not the case in C or C++. Thus C# is more suitable for writing distributed applications.
- In C#, data types belong to either value types or reference types.
- C# checks for uninitialized variables and gives error messages at compile time. In C++, an uninitialized variable goes undetected thus resulting in unpredictable output.
- In C#, structs are value types.

C++ features dropped

The following C++ features are missing from C#:

- Macros
- Multiple Inheritance
- Templates
- Pointers
- Global Variables
- Typedef statement
- Default arguments
- Constant member functions or parameters
- Forward declaration of classes

Enhancements to C++

- Modernizes C++ by adding the following new features:
- Automatic garbage collection
- Versioning support
- Strict type-safety
- Properties to access data members
- Delegates and events
- Boxing and unboxing
- Web services

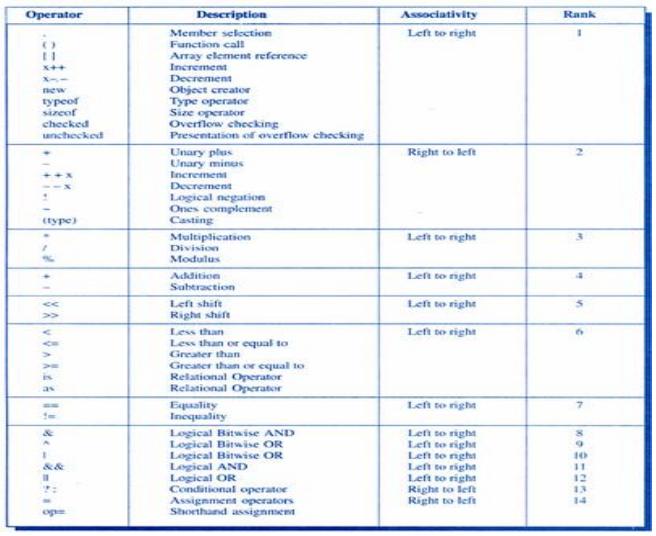
2. Discuss in detail about various types of operators.

Operators are symbols used in expressions to describe operations involving one or more operands.

Types

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Increment and Decrement Operators
- Conditional Operators
- Bitwise Operators
- Special Operators

Summary of C# Operators



Operator	Description	Associativity	Rank
.	Member selection	Left to right	1
( )	Function call		
[ ]	Array element reference		
x++	Increment		
x--,--	Decrement		
new	Object creator		
typeof	Type operator		
sizeof	Size operator		
checked	Overflow checking		
unchecked	Presentation of overflow checking		
+	Unary plus	Right to left	2
–	Unary minus		
+ + x	Increment		
– – x	Decrement		
!	Logical negation		
~	Ones complement		
(type)	Casting		
*	Multiplication	Left to right	3
/	Division		
%	Modulus		
+	Addition	Left to right	4
–	Subtraction		
<<	Left shift	Left to right	5
>>	Right shift		
<	Less than	Left to right	6
<=	Less than or equal to		
>	Greater than		
>=	Greater than or equal to		
is	Relational Operator		
as	Relational Operator		
==	Equality	Left to right	7
!=	Inequality		
&	Logical Bitwise AND	Left to right	8
^	Logical Bitwise OR	Left to right	9
|	Logical Bitwise OR	Left to right	10
&&	Logical AND	Left to right	11
||	Logical OR	Left to right	12
? :	Conditional operator	Right to left	13
=	Assignment operators	Right to left	14
op=	Shorthand assignment		

3. What are the major components of a commo lain the services provided by CLR.

It consists of 3 distinct technologies:

- Common Language Runtime
- Framework Base Classes
- User and program interfaces

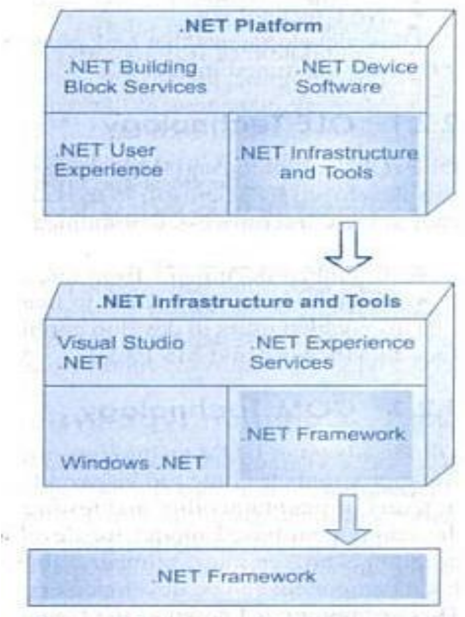


Figure 2.3 *Various Components of .NET platform*

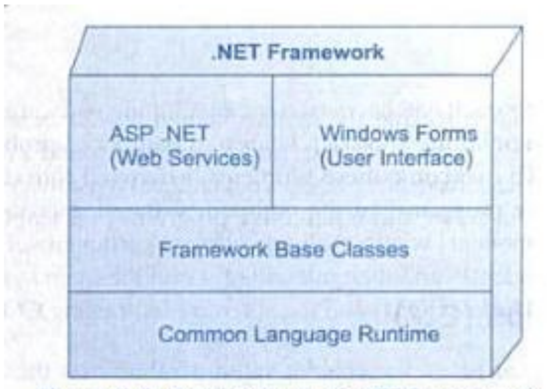
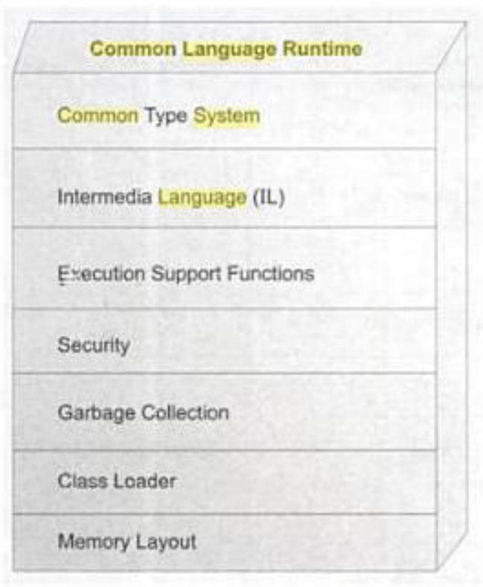


Figure 2.4 *Architecture of .NET Framework*

Components of CLR



Services provided by CLR

- Loading and execution of programs
- Memory isolation for applications
- Verification of type safety
- Compilation of IL into native executable code
- Providing metadata
- Memory management
- Enforcement of security
- Interoperability with other systems
- Managing exceptions and errors
- Support for tasks such as debugging and profiling

4. (i) Explain namespace from command line arguments in C# with examples.

We all know, that we can pass parameter to a function as argument but what about Main(string[] args) method? Do parameters can be passed to Main() method in C#? Yes, parameter(s) can be passed to a Main() method in C# and it is called **command line argument**.

Main() method is where program stars execution. Main method doesn't accept parameter from any method. It accept parameter through command line. It is an array type parameter that can accept n number of parameter in runtime.

In Main(string[] args) , args is a string type of array that can hold numerous parameter.

C# command line argument example:

To understand command line argument in C#, consider the following command line example.

1. Open Notepad and write the following code and save it with **command.cs**

```
using System;
 namespace command
{
  class Program
   {
     static void Main(string[] args)
      {
        Console.WriteLine("First Name is " + args[0]);
        Console.WriteLine("Last Name is " + args[1]);
        Console.ReadLine();
      }
   }
}
```

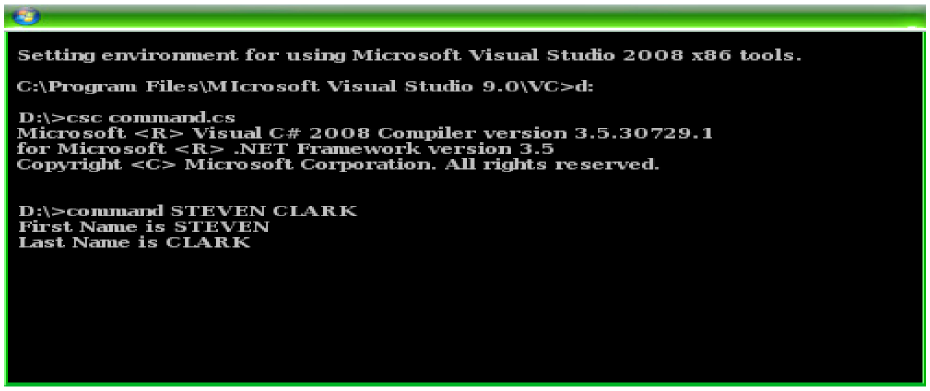
2. Open visual studio command prompt and compile the code as follow:

(i) Set current path, where your program is saved.

(ii) Compile it with **csc command.cs**

3. Now execute the program using following command line argument:

(a) command steven clark



```
Setting environment for using Microsoft Visual Studio 2008 x86 tools.

C:\Program Files\MIcrosoft Visual Studio 9.0\VC>d:

D:\>csc command.cs
Microsoft <R> Visual C# 2008 Compiler version 3.5.30729.1
for Microsoft <R> .NET Framework version 3.5
Copyright <C> Microsoft Corporation. All rights reserved.


D:\>command STEVEN CLARK
First Name is STEVEN
Last Name is CLARK
```

[illegible]

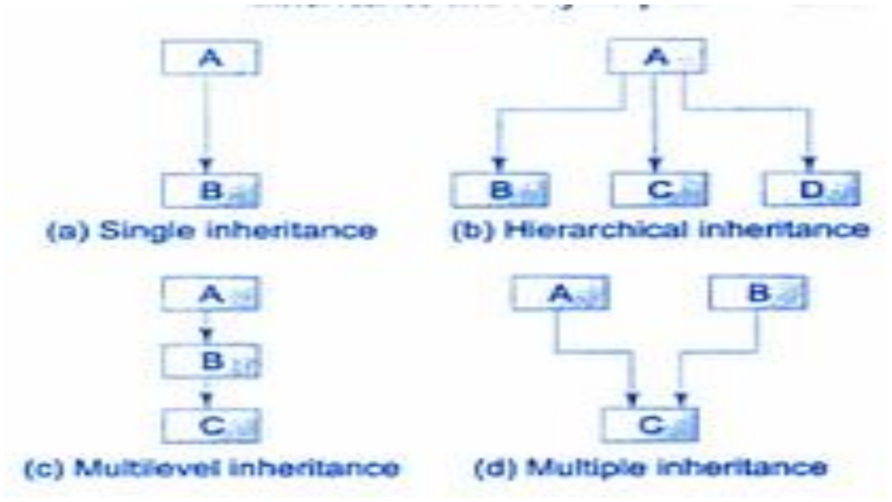
(ii) Give a C# program to compute the binominal co-efficient. The binominal co-efficient is given as nCr = $\frac{n!}{(n-r)!r!}$

using System;

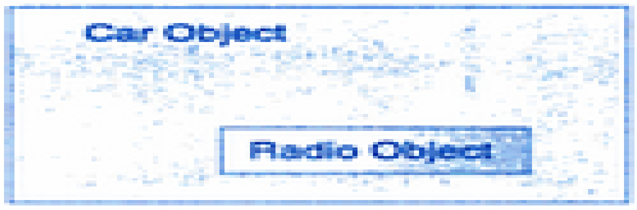
```
namespace BinomialCoefficients
{
    class Program
    {
        static void Main(string[] args)
        {
            ulong n = 1000000, k = 3;
            ulong result = biCoefficient(n, k);
            Console.WriteLine("The Binomial Coefficient of {0}, and {1}, is equal to: {2}", n, k, result);
            Console.ReadLine();
        }

        static int fact(int n)
        {
            if (n == 0) return 1;
            else return n * fact(n - 1);
        }

        static ulong biCoefficient(ulong n, ulong k)
        {
            if (k > n - k)
            {
                k = n - k;
            }

            ulong c = 1;
            for (uint i = 0; i < k; i++)
            {
                c = c * (n - i);
                c = c / (i + 1);
            }
            return c;
        }
    }
}
```

5. What is meant by Inheritance? Explain the types of Inheritance.
The mechanism of designing or constructing one class from another is called inheritance.
Two forms
- Classical form
- Containment form

Classical inheritance: -is-a-relationship
Different combinations



Containment Inheritance: -has-a relationship
Example



6. Define an Interface. How will you extend and implement an interface? Explain with an example.
An interface in C# is a reference type. Interfaces can define methods, properties, indexers, and events. To implement an interface, a class or structure must publicly implement all members of that interface.
General Form

```
interface
{
Member declarations;
}
```

Example

```
interface Example
{
int Aproperty
{get;}
event someEvent Changed;
void Display();
```

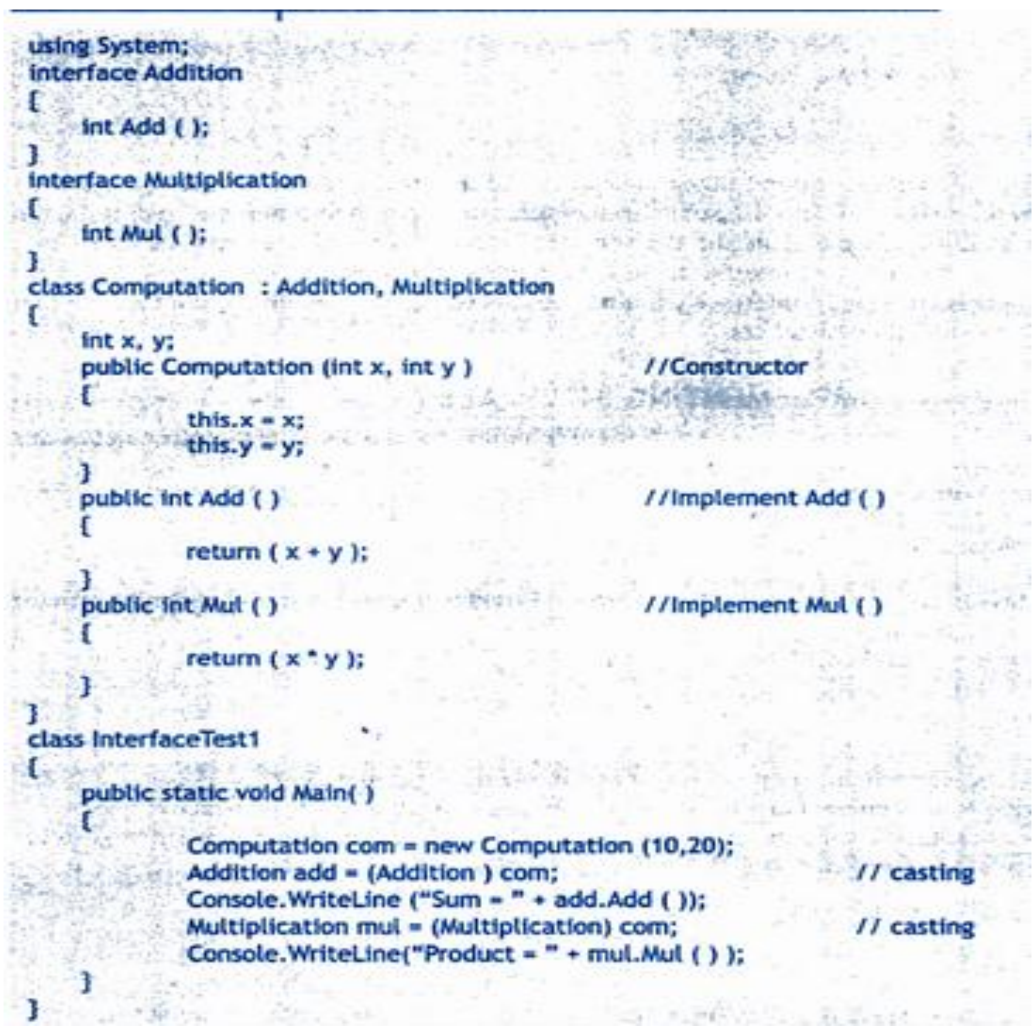
}
Extending an interface

An interface can be subinterfaced form other interfaces. The new subinterface will inherit all the members of the superinterface.

Example

```
interface Addition
{
 int Add(int x, int y);
}
interface Compute:Addition
{
int Sub(int x, int y);
}
```

Implementation of multiple interfaces



```
using System;
interface Addition
{
    int Add ( );
}
interface Multiplication
{
    int Mul ( );
}
class Computation  : Addition, Multiplication
{
    int x, y;
    public Computation (int x, int y )              //Constructor
    {
            this.x = x;
            this.y = y;
    }
    public int Add ( )                               //Implement Add ( )
    {
            return ( x + y );
    }
    public int Mul ( )                               //Implement Mul ( )
    {
            return ( x * y );
    }
}
class InterfaceTest1
{
    public static void Main( )
    {
            Computation com = new Computation (10,20);
            Addition add = (Addition ) com;                     // casting
            Console.WriteLine ("Sum = " + add.Add ( ));
            Multiplication mul = (Multiplication) com;          // casting
            Console.WriteLine("Product = " + mul.Mul ( ) );
    }
}
```

7. Explain operator overloading concept with example.

You can redefine or overload most of the built-in operators available in C#. Thus a programmer can use operators with user-defined types as well. Overloaded operators are functions with special names the keyword operator followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

Example

```
public static Box operator+ (Box b, Box c)
{
  Box box = new Box();
  box.length = b.length + c.length;
  box.breadth = b.breadth + c.breadth;
  box.height = b.height + c.height;
  return box;
}
```

Implementation of Operator Overloading

```
using System;

namespace OperatorOvlApplication
{
  class Box
  {
    private double length;      // Length of a box
    private double breadth;     // Breadth of a box
    private double height;      // Height of a box

    public double getVolume()
    {
      return length * breadth * height;
    }
    public void setLength( double len )
    {
      length = len;
    }

    public void setBreadth( double bre )
    {
      breadth = bre;
    }

    public void setHeight( double hei )
    {
      height = hei;
    }
    // Overload + operator to add two Box objects.
    public static Box operator+ (Box b, Box c)
    {
      Box box = new Box();
      box.length = b.length + c.length;
      box.breadth = b.breadth + c.breadth;
      box.height = b.height + c.height;
      return box;
    }
```

```
   }

   class Tester
   {
      static void Main(string[] args)
      {
         Box Box1 = new Box();         // Declare Box1 of type Box
         Box Box2 = new Box();         // Declare Box2 of type Box
         Box Box3 = new Box();         // Declare Box3 of type Box
         double volume = 0.0;          // Store the volume of a box here

         // box 1 specification
         Box1.setLength(6.0);
         Box1.setBreadth(7.0);
         Box1.setHeight(5.0);

         // box 2 specification
         Box2.setLength(12.0);
         Box2.setBreadth(13.0);
         Box2.setHeight(10.0);

         // volume of box 1
         volume = Box1.getVolume();
         Console.WriteLine("Volume of Box1 : {0}", volume);

         // volume of box 2
         volume = Box2.getVolume();
         Console.WriteLine("Volume of Box2 : {0}", volume);

         // Add two object as follows:
         Box3 = Box1 + Box2;

         // volume of box 3
         volume = Box3.getVolume();
         Console.WriteLine("Volume of Box3 : {0}", volume);
         Console.ReadKey();
      }
   }
}
```
Output:
```
Volume of Box1 : 210
Volume of Box2 : 1560
Volume of Box3 : 5400
```

Overloadable and Non-Overloadable Operators

Operators	Description
+, -, !, ~, ++, --	These unary operators take one operand and can be overloaded.
+, -, *, /, %	These binary operators take one operand and can be overloaded.
==, !=, <, >, <=, >=	The comparison operators can be overloaded
&&, ||	The conditional logical operators cannot be overloaded

	directly.
+=, -=, *=, /=, %=	The assignment operators cannot be overloaded.
=, ., ?:, ->, new, is, sizeof, typeof	These operators cannot be overloaded.

8. Explain Exception handling in detail with an example.

An exception is a problem that arises during the execution of a program. A C# exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C# exception handling is built upon four keywords: try, catch, finally and throw.

try: A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

catch: A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.

finally: The finally block is used to execute a given set of statements, whether an exception is thrown or not thrown. For example, if you open a file, it must be closed whether an exception is raised or not.

throw: A program throws an exception when a problem shows up. This is done using a throw keyword.

Syntax

```
try
{
  // statements causing exception
}
catch( ExceptionName e1 )
{
  // error handling code
}
catch( ExceptionName e2 )
{
  // error handling code
}
catch( ExceptionName eN )
{
  // error handling code
}
finally
{
  // statements to be executed
}
```

Exception Classes in C#

C# exceptions are represented by classes. The exception classes in C# are mainly directly or indirectly derived from the System.Exception class. Some of the exception classes derived from the System.Exception class are the System.ApplicationException and System.SystemExceptionclasses.

The System.ApplicationException class supports exceptions generated by application programs. So the exceptions defined by the programmers should derive from this class.

The System.SystemException class is the base class for all predefined system exception.

The following table provides some of the predefined exception classes derived from the Sytem.SystemException class:

Exception Class	Description
System.IO.IOException	Handles I/O errors.
System.IndexOutOfRangeException	Handles errors generated when a method refers to an array index out of range.
System.ArrayTypeMismatchException	Handles errors generated when type is mismatched

	with the array type.
System.NullReferenceException	Handles errors generated from deferencing a null object.
System.DivideByZeroException	Handles errors generated from dividing a dividend with zero.
System.InvalidCastException	Handles errors generated during typecasting.
System.OutOfMemoryException	Handles errors generated from insufficient free memory.
System.StackOverflowException	Handles errors generated from stack overflow.

Handling Exceptions

C# provides a structured solution to the exception handling problems in the form of try and catch blocks. Using these blocks the core program statements are separated from the error-handling statements.

9. (i) Briefly explain assemblies and versioning.

The basic unit of .NET programming is the *assembly*. An assembly is a collection of files thatappears to the user to be a single dynamic link library (DLL) or executable (EXE). DLLs arecollections of classes and methods that are linked into your running program only when theyare needed.Assemblies are the .NET unit of reuse, versioning, security, and deployment.

PE Files

On disk, assemblies are Portable Executable (PE) files. PE files are not new. The format of a .NET PE file is exactly the same as a normal Windows PE file. PE files are implemented asDLLs or EXEs. Logically (as opposed to physically), assemblies consist of one or more*modules*. Note, however, that an assembly must have exactly one entry point -- DLLMain,WinMain, or Main. DLLMain is the entry point for DLLs, WinMain is the entry point forWindows applications, and Main is the entry point for DOS and Console applications.Modules are created as DLLs and are the constituent pieces of assemblies. Standing alone,modules cannot be executed; they must be combined into assemblies to be useful.Deploy and reuse the entire contents of an assembly as a unit. Assemblies are loaded ondemand and will not be loaded if not needed.

Metadata

Metadata is information stored in the assembly that describes the types and methods of the assembly and provides other useful information about the assembly. Assemblies are said to be*self-describing* because the metadata fully describes the contents of each module.

Security Boundary

Assemblies form security boundaries as well as type boundaries. That is, an assembly is the scope boundary for the types it contains, and types cannot cross assemblies. It is possible to refer to types across assembly boundaries by adding a reference to the requiredassembly, either in the Integrated Development Environment (IDE) or on the command line,at compile time.

Versioning

Each assembly has a version number, and versions cannot transcend the boundary of theassembly. That is, a version can refer only to the contents of a single assembly. All types andresources within the assembly change versions together.

Manifests

As part of its metadata, every assembly has a *manifest*. This describes what is in the assembly, including identification information (name, version, etc.), a list of the types and resources inthe assembly, a map to connect public types with the implementing code, and a list of assemblies referenced by this assembly.

Other Required Assemblies

The assembly manifest also contains references to other required assemblies. Each suchreference includes the name of the other assembly, the version number and required culture, and optionally, the other assembly's originator. The originator is a digital signature for thedeveloper or company that provided the other assembly.

Multi-Module Assemblies

A single-module assembly has a single file that can be an EXE or DLL file. This singlemodule contains all the types and implementations for the application. The assembly manifestis embedded within this module.

A multi-module assembly consists of multiple files (zero or one EXE and zero or more DLL files, though you must have at least one EXE or DLL). The assembly manifest in this case canreside in a standalone file, or it can be embedded in one of the modules. When the assembly isreferenced, the runtime loads the file containing the manifest and then loads the requiredmodules as needed.

Private Assemblies

Assemblies come in two flavors: *private* and *shared* . Private assemblies are intended to be used by only one application; shared assemblies are intended to be shared among manyapplications.

Shared Assemblies

It is possible to create assemblies that can be shared by other applications. First, your assembly must have a *strong name*. Strong names are globally unique.Second, your shared assembly must be protected against newer versions trampling over it, andso it must have version control.Finally, to share your assembly, place it in the *Global Assembly Cache (GAC)* (pronouncedGACK). This is an area of the filesystem set aside by the Common Language Runtime (CLR)to hold shared assemblies.

(ii) Explain the various types of assemblies.

Private Assemblies

Assemblies come in two flavors: *private* and *shared* . Private assemblies are intended to be used by only one application; shared assemblies are intended to be shared among manyapplications.
All the assemblies you've built so far are private. By default, when you compile a C#application, a private assembly is created. The files for a private assembly are all kept in thesame folder (or in a tree of subfolders). This tree of folders is isolated from the rest of thesystem, as nothing other than the one application depends on it, and you can redeploy thisapplication to another machine just by copying the folder and its subfolders.

A private assembly can have any name you choose. It does not matter if that name clashes with assemblies in another application; the names are local only to a single application.In the past, DLLs were installed on a machine and an entry was made in the WindowsRegistry. It was difficult to avoid corrupting the Registry, and reinstalling the program onanother machine was nontrivial. With assemblies, all of that goes away. With privateassemblies, installing is as simple as copying the files to the appropriate directory.

Shared Assemblies

It is possible to create assemblies that can be shared by other applications. You might want to do thisif you have written a generic control or a class that might be used by other developers. If youwant to share your assembly, it must meet certain stringent requirements.

First, your assembly must have a *strong name*. Strong names are globally unique.

Second, your shared assembly must be protected against newer versions trampling over it, and so it must have version control.

Finally, to share your assembly, place it in the *Global Assembly Cache (GAC)* (pronouncedGACK). This is an area of the filesystem set aside by the Common Language Runtime (CLR)to hold shared assemblies.

10. Explain in detail about .Net framework and its architecture.

- Common Language Runtime
- Windows Forms
- ASP.NET
  - Web Forms
  - Web Services

- ADO.NET, evolution of ADO
- Visual Studio.NET
- CLR works like a virtual machine in executing all languages.
- All .NET languages must obey the rules and standards imposed by CLR. Examples:
    - Object declaration, creation and use
    - Data types,language libraries
    - Error and exception handling
    - Interactive Development Environment (IDE)
- Development
    - Mixed language applications
- Common Language Specification (CLS)
- Common Type System (CTS)
- Standard class framework
- Automatic memory management
    - Consistent error handling and safer execution
    - Potentially multi-platform
- Deployment
    - Removal of registration dependency
    - Safety – fewer versioning problems
- CTS is a rich type system built into the CLR
    - Implements various types (int, double, etc)
    - And operations on those types
- CLS is a set of specifications that language and library designers need to follow
    - ill ensure interoperability between languages
- .NET languages are not compiled to machine code. They are compiled to an Intermediate Language (IL).
- CLR accepts the IL code and recompiles it to machine code. The recompilation is just-in-time (JIT) meaning it is done as soon as a function or subroutine is called.
- The JIT code stays in memory for subsequent calls. In cases where there is not enough memory it is discarded thus making JIT process interpretive.

UNIT – V
PART – B

1. Explain the following

i. ASP.net page life cycle

ASP.NET life cycle specifies, how:

- ASP.NET processes pages to produce dynamic output
- The application and its pages are instantiated and processed
- ASP.NET compiles the pages dynamically

The ASP.NET life cycle could be divided into two groups:

- Application Life Cycle
- Page Life Cycle

ASP.NET Application Life Cycle

- The application life cycle has the following stages:
- User makes a request for accessing application resource, a page. Browser sends this request to the web server.
- A unified pipeline receives the first request and the following events take place:
- An object of the class ApplicationManager is created.
- An object of the class HostingEnvironment is created to provide information regarding the resources.

- Top level items in the application are compiled.
- Response objects are created. The application objects such as HttpContext, HttpRequest and HttpResponse are created and initialized.
- An instance of the HttpApplication object is created and assigned to the request.
- The request is processed by the HttpApplication class. Different events are raised by this class for processing the request.
- ASP.NET Page Life Cycle
- When a page is requested, it is loaded into the server memory, processed, and sent to the browser. Then it is unloaded from the memory. At each of these steps, methods and events are available, which could be overridden according to the need of the application. In other words, you can write your own code to override the default code.
- The Page class creates a hierarchical tree of all the controls on the page. All the components on the page, except the directives, are part of this control tree. You can see the control tree by adding trace= "true" to the page directive. We will cover page directives and tracing under 'directives' and 'event handling'.

The page life cycle phases are:

- Initialization
- Instantiation of the controls on the page
- Restoration and maintenance of the state
- Execution of the event handler codes
- Page rendering

Understanding the page cycle helps in writing codes for making some specific thing happen at any stage of the page life cycle. It also helps in writing custom controls and initializing them at right time, populate their properties with view-state data and run control behavior code.

Following are the different stages of an ASP.NET page:

- **Page request -** When ASP.NET gets a page request, it decides whether to parse and compile the page, or there would be a cached version of the page; accordingly the response is sent.
- **Starting of page life cycle -** At this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.
- **Page initialization -** At this stage, the controls on the page are assigned unique ID by setting the UniqueID property and the themes are applied. For a new request, postback data is loaded and the control properties are restored to the view-state values.
- **Page load -** At this stage, control properties are set using the view state and control state values.
- **Validation -** Validate method of the validation control is called and on its successful execution, the IsValid property of the page is set to true.
- **Postback event handling -** If the request is a postback (old request), the related event handler is invoked.
- **Page rendering -** At this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Response property of page.
- **Unload -** The rendered page is sent to the client and page properties, such as Response and Request, are unloaded and all cleanup done.

ii. ASP Vs ASP.NET

ASP:

- It has limited oops support and not having built in support for xml.
- Very less development and debugging tool available. Meaning that difficult to debug the code.
- ASP you can only do scripting using visual basic scripting and java scripting.
- Error handling is very poor.
- It has no high level programming structure. Mixed of html and server side scripting.
- You must be entering first line as -

- <%LANGUAGE="VBSCRIPT" CODEPAGE="960"%>
- It has no in built validation control. Meaning that validating page is difficult for developers.
- In the classic ASP if you need to update code on the existing page then it is mandatory to restart the server to get reflect.

ASP.NET

- ASP.NET is full featured object oriented programming.
- It has full support of xml. Which helps easy data exchange.
- Various tools and compiler available. Microsoft Visual studio makes your debugging job easier.
- ASP.NET we can use either C# or VB.NET as server side programming language.
- ASP.NET gives you three tire architecture. It allow you to keep your business logic, views everything separate. Meaning that easy to enhance applications.
- Error handling is very good.
- You are required to make language directive with page as below.
- <%@Page Language="VB" CodePage="960"%>
- <%@QutputCache Duration="60" VaryByParam="none" %>
- It has state management support.
- In built validation controls. It has rich validation set - custom validator, range validator, regular expression, compare and require field validation control which makes your job easier.

2. Explain about the validation controls with example

ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

ASP.NET provides the following validation controls:

- BaseValidator
- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

BaseValidator Class

The validation control classes are inherited from the BaseValidator class hence they inherit its properties and methods. Therefore, it would help to take a look at the properties and the methods of this base class, which are common for all the validation controls:

Members	**Description**
ControlToValidate	Indicates the input control to validate.
Display	Indicates how the error message is shown.
EnableClientScript	Indicates whether client side validation will take.
Enabled	Enables or disables the validator.
ErrorMessage	Indicates error string.
Text	Error text to be shown if validation fails.
IsValid	Indicates whether the value of the control is valid.

SetFocusOnError	It indicates whether in case of an invalid control, the focus should switch to the related input control.
ValidationGroup	The logical group of multiple validators, where this control belongs.
Validate()	This method revalidates the control and updates the IsValid property.

RequiredFieldValidator Control

The RequiredFieldValidator control ensures that the required field is not empty. It is generally tied to a text box to force input into the text box.

The syntax of the control is as given:

```
<asp:RequiredFieldValidator ID="rfvcandidate"
runat="server" ControlToValidate ="ddlcandidate"
ErrorMessage="Please choose a candidate"
InitialValue="Please choose a candidate">
</asp:RequiredFieldValidator>
```

RangeValidator Control

The RangeValidator control verifies that the input value falls within a predetermined range.
It has three specific properties:

Properties	Description
Type	It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String.
MinimumValue	It specifies the minimum value of the range.
MaximumValue	It specifies the maximum value of the range.

The syntax of the control is as given:

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"
ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
MinimumValue="6" Type="Integer">
</asp:RangeValidator>
```

CompareValidator Control

The CompareValidator control compares a value in one control with a fixed value or a value in another control.

It has the following specific properties:

Properties	**Description**
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.

The basic syntax of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
ErrorMessage="CompareValidator">
</asp:CompareValidator>
```

RegularExpressionValidator

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.

The following table summarizes the commonly used syntax constructs for regular expressions:

Character Escapes	**Description**
\b	Matches a backspace.
\t	Matches a tab.
\r	Matches a carriage return.
\v	Matches a vertical tab.
\f	Matches a form feed.
\n	Matches a new line.
\	Escape character.

Apart from single character match, a class of characters could be specified that can be matched, called the metacharacters.

Metacharacters	**Description**
.	Matches any character except \n.
[abcd]	Matches any character in the set.
[^abcd]	Excludes any character in the set.
[2-7a-mA-M]	Matches any character specified in the range.
\w	Matches any alphanumeric character and underscore.
\W	Matches any non-word character.
\s	Matches whitespace characters like, space, tab, new line etc.
\S	Matches any non-whitespace character.
\d	Matches any decimal character.
\D	Matches any non-decimal character.

Quantifiers could be added to specify number of times a character could appear.

Quantifier	**Description**
*	Zero or more matches.
+	One or more matches.
?	Zero or one matches.
{N}	N matches.
{N,}	N or more matches.
{N,M}	Between N and M matches.

The syntax of the control is as given:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
  ValidationExpression="string" ValidationGroup="string">
  </asp:RegularExpressionValidator>
```

CustomValidator

The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.

The client side validation is accomplished through the ClientValidationFunction property. The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.

The server side validation routine must be called from the control's ServerValidate event handler. The server side validation routine should be written in any .Net language, like C# or VB.Net.

The basic syntax for the control is as given:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
   ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">

</asp:CustomValidator>
```

ValidationSummary
The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.
The following two mutually inclusive properties list out the error message:

- **ShowSummary** : shows the error messages in specified format.
- **ShowMessageBox** : shows the error messages in a separate window.

The syntax for the control is as given:

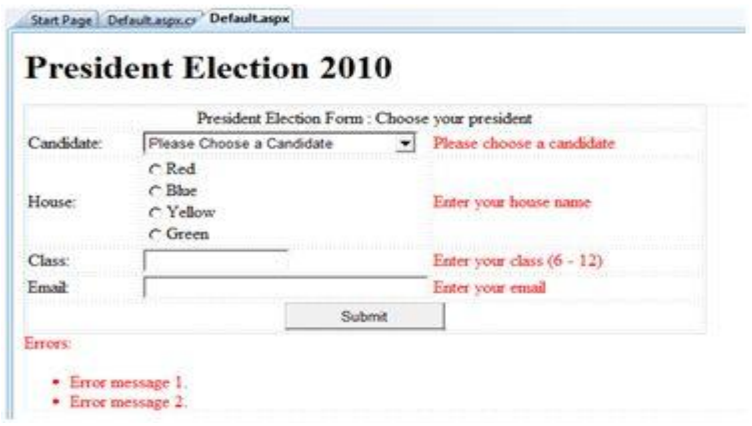
```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
   DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

Validation Groups
Complex pages have different groups of information provided in different panels. In such situation, a need might arise for performing validation separately for separate group. This kind of situation is handled using validation groups.

To create a validation group, you should put the input controls and the validation controls into the same logical group by setting their *ValidationGroup* property.

Example
The following example describes a form to be filled up by all the students of a school, divided into four houses, for electing the school president. Here, we use the validation controls to validate the user input.

This is the form in design view:



President Election 2010

President Election Form : Choose your president

Candidate:	Please Choose a Candidate	Please choose a candidate
House:	Red Blue Yellow Green	Enter your house name
Class:		Enter your class (6 - 12)
Email:		Enter your email

Errors:

- Error message 1.
- Error message 2.

The content file code is as given:

```
<form id="form1" runat="server">
```

```
<table style="width: 66%;">

  <tr>
    <td class="style1" colspan="3" align="center">
    <asp:Label ID="lblmsg"
      Text="President Election Form : Choose your president"
      runat="server" />
    </td>
  </tr>

  <tr>
    <td class="style3">
      Candidate:
    </td>

    <td class="style2">
      <asp:DropDownList ID="ddlcandidate" runat="server"  style="width:239px">
        <asp:ListItem>Please Choose a Candidate</asp:ListItem>
        <asp:ListItem>M H Kabir</asp:ListItem>
        <asp:ListItem>Steve Taylor</asp:ListItem>
        <asp:ListItem>John Abraham</asp:ListItem>
        <asp:ListItem>Venus Williams</asp:ListItem>
      </asp:DropDownList>
    </td>

    <td>
      <asp:RequiredFieldValidator ID="rfvcandidate"
        runat="server" ControlToValidate ="ddlcandidate"
        ErrorMessage="Please choose a candidate"
        InitialValue="Please choose a candidate">
      </asp:RequiredFieldValidator>
    </td>
  </tr>

  <tr>
    <td class="style3">
      House:
    </td>

    <td class="style2">
      <asp:RadioButtonList ID="rblhouse" runat="server" RepeatLayout="Flow">
        <asp:ListItem>Red</asp:ListItem>
        <asp:ListItem>Blue</asp:ListItem>
        <asp:ListItem>Yellow</asp:ListItem>
        <asp:ListItem>Green</asp:ListItem>
      </asp:RadioButtonList>
    </td>

    <td>
      <asp:RequiredFieldValidator ID="rfvhouse" runat="server"
        ControlToValidate="rblhouse" ErrorMessage="Enter your house name" >
      </asp:RequiredFieldValidator>
```

```
            <br />
          </td>
        </tr>

        <tr>
          <td class="style3">
            Class:
          </td>

          <td class="style2">
            <asp:TextBox ID="txtclass" runat="server"></asp:TextBox>
          </td>

          <td>
            <asp:RangeValidator ID="rvclass"
              runat="server" ControlToValidate="txtclass"
              ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
              MinimumValue="6" Type="Integer">
            </asp:RangeValidator>
          </td>
        </tr>

        <tr>
          <td class="style3">
            Email:
          </td>

          <td class="style2">
            <asp:TextBox ID="txtemail" runat="server" style="width:250px">
            </asp:TextBox>
          </td>

          <td>
            <asp:RegularExpressionValidator ID="remail" runat="server"
              ControlToValidate="txtemail" ErrorMessage="Enter your email"
              ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
            </asp:RegularExpressionValidator>
          </td>
        </tr>

        <tr>
          <td class="style3" align="center" colspan="3">
            <asp:Button ID="btnsubmit" runat="server" onclick="btnsubmit_Click"
              style="text-align: center" Text="Submit" style="width:140px" />
          </td>
        </tr>
      </table>
      <asp:ValidationSummary ID="ValidationSummary1" runat="server"
        DisplayMode ="BulletList" ShowSummary ="true" HeaderText="Errors:" />
</form>
```
The code behind the submit button:

```
protected void btnsubmit_Click(object sender, EventArgs e)
```

```
{
  if (Page.IsValid)
  {
    lblmsg.Text = "Thank You";
  }
  else
  {
    lblmsg.Text = "Fill up all the fields";
  }
}
```

3. Explain about the session management in ASP. NET

State management is the process by which you maintain state and page information over multiple requests for the same or different pages.

Types of State Management

1. Client – Side State Management This stores information on the client's computer by embedding the information into a Web page, a uniform resource locator(url), or a cookie. The techniques available to store the state information at the client end are listed down below:

- **View State** – Asp.Net uses View State to track the values in the Controls. You can add custom values to the view state. It is used by the Asp.net page framework to automatically save the values of the page and of each control just prior to rendering to the page. When the page is posted, one of the first tasks performed by page processing is to restore view state.
- **Control State** – If you create a custom control that requires view state to work properly, you should use control state to ensure other developers don't break your control by disabling view state.

  Hidden fields – Like view state, hidden fields store data in an HTML form without displaying it in the user's browser. The data is available only when the form is processed.
- **Cookies** – Cookies store a value in the user's browser that the browser sends with every page request to the same server. Cookies are the best way to store state data that must be available for multiple Web pages on a web site.
- **Query Strings** - Query strings store values in the URL that are visible to the user. Use query strings when you want a user to be able to e-mail or instant message state data with a URL.

2. Server – Side State Management

- **Application State -** Application State information is available to all pages, regardless of which user requests a page.
- **Session State** – Session State information is available to all pages opened by a user during a single visit.

Both application state and session state information is lost when the application restarts. To persist user data between application restarts, you can store it using profile properties. Implementation Procedure

Client – Side State Management:

View State:

The ViewState property provides a dictionary object for retaining values between multiple requests for the same page. When an ASP.NET page is processed, the current state of the page and controls is hashed into a string and saved in the page as a hidden field. If the data is too long for a single field, then ASP.NET performs view state chunking (new in ASP.NET 2.0) to split it across multiple hidden fields. The following code sample demonstrates how view state adds data as a hidden form within a Web page's HTML:

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE”
value="/wEPDwUKMTIxNDIyOTM0Mg9kFgICAw9kFgICAQ8PFgIeBFRleHQFEzQvNS8yMDA2IDE
6Mzc6MTEgUE1kZGROWHn/rt75XF/pMGnqjqHlH66cdw==" />
```

Encrypting of the View State: You can enable view state encryption to make it more difficult for attackers and malicious users to directly read view state information. Though this adds processing overhead to the Web server, it supports in storing confidential information in view state. To configure view state encryption for an application does the following:

```
<Configuration>
        <system.web>
   <pages viewStateEncryptionMode="Always"/>
        </system.web>
</configuration>
```

Alternatively, you can enable view state encryption for a specific page by setting the value in the page directive, as the following sample demonstrates:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default"
ViewStateEncryptionMode="Always"%>
```

View State is enabled by default, but if you can disable it by setting the EnableViewState property for each web control to false. This reduces the server processing time and decreases page size. Reading and Writing Custom View State Data:
If you have a value that you'd like to keep track of while the user is visiting a single ASP.NET Web page, adding a custom value to ViewState is the most efficient and secure way to do that. However, ViewState is lost if the user visits a different Web page, so it is useful only for temporarily storing values. Example: Determine the time of last visit to the page

```
// Check if View State object exists, and display it if it does
If (ViewState ["lastVisit"]!= null)
Label1.Text = (string)ViewState["lastVisit"]; else
Label1.Text = "lastVisit ViewState not defined.";
// Define the ViewState object for the next page view ViewState.Add("lastVisit",
DateTime.Now.ToString());
```

4. Explain about data binding and caching in ASP.NET

The ASP.NET web form control inherits the DataBind method from its parent Control class, which gives it an inherent capability to bind data to at least one of its properties. This is known as **simple data binding** or **inline data binding**.
Simple data binding involves attaching any collection (item collection) which implements the IEnumerable interface, or the DataSet and DataTable classes to the DataSource property of the control.
On the other hand, some controls can bind records, lists, or columns of data into their structure through a DataSource control. These controls derive from the BaseDataBoundControl class. This is called **declarative data binding**.

The data source controls help the data-bound controls implement functionalities such as, sorting, paging, and editing data collections.
The BaseDataBoundControl is an abstract class, which is inherited by two more abstract classes:
DataBoundControl
HierarchicalDataBoundControl
The abstract class DataBoundControl is again inherited by two more abstract classes:
ListControl
CompositeDataBoundControl
The controls capable of simple data binding are derived from the ListControl abstract class and these controls are:
BulletedList
CheckBoxList
DropDownList
ListBox
RadioButtonList

The controls capable of declarative data binding (a more complex data binding) are derived from the abstract class CompositeDataBoundControl. These controls are:
DetailsView
FormView
GridView
RecordList

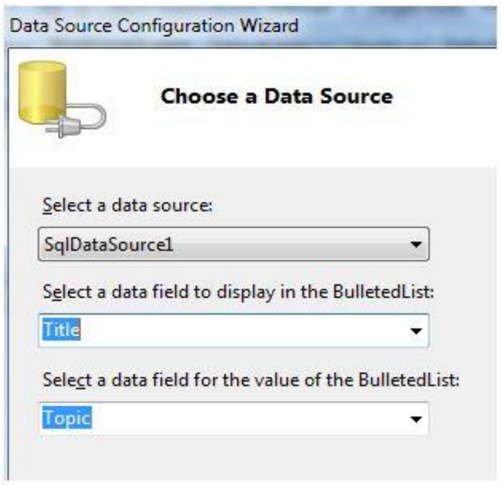
Simple Data Binding

Simple data binding involves the read-only selection lists. These controls can bind to an array list or fields from a database. Selection lists takes two values from the database or the data source; one value is displayed by the list and the other is considered as the value corresponding to the display.

Let us take up a small example to understand the concept. Create a web site with a bulleted list and a SqlDataSource control on it. Configure the data source control to retrieve two values from your database (we use the same DotNetReferences table as in the previous chapter).

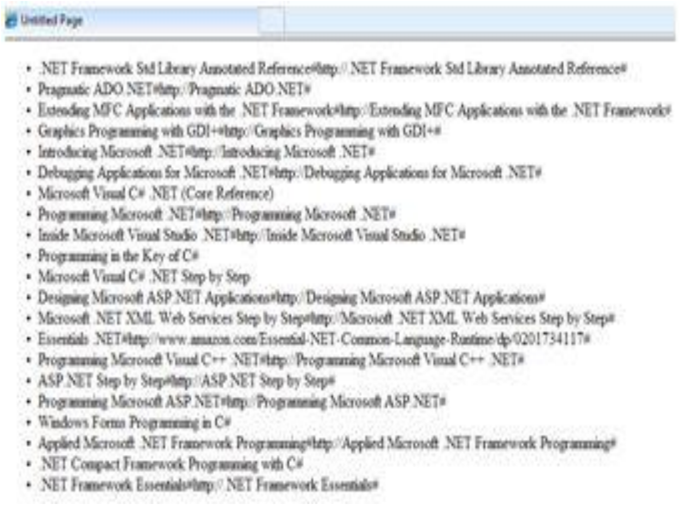
Choosing a data source for the bulleted list control involves:

- Selecting the data source control
- Selecting a field to display, which is called the data field
- Selecting a field for the value



Select a data source: SqlDataSource1
Select a data field to display in the BulletedList: Title
Select a data field for the value of the BulletedList: Topic

When the application is executed, check that the entire title column is bound to the bulleted list and displayed.



- .NET Framework Std Library Annotated Reference#http://.NET Framework Std Library Annotated Reference#
- Pragmatic ADO.NET#http://Pragmatic ADO.NET#
- Extending MFC Applications with the .NET Framework#http://Extending MFC Applications with the .NET Framework#
- Graphics Programming with GDI+#http://Graphics Programming with GDI+#
- Introducing Microsoft .NET#http://Introducing Microsoft .NET#
- Debugging Applications for Microsoft .NET#http://Debugging Applications for Microsoft .NET#
- Microsoft Visual C# .NET (Core Reference)
- Programming Microsoft .NET#http://Programming Microsoft .NET#
- Inside Microsoft Visual Studio .NET#http://Inside Microsoft Visual Studio .NET#
- Programming in the Key of C#
- Microsoft Visual C# .NET Step by Step
- Designing Microsoft ASP.NET Applications#http://Designing Microsoft ASP.NET Applications#
- Microsoft .NET XML Web Services Step by Step#http://Microsoft .NET XML Web Services Step by Step#
- Essentials .NET#http://www.amazon.com/Essential-NET-Common-Language-Runtime/dp/0201734117#
- Programming Microsoft Visual C++ .NET#http://Programming Microsoft Visual C++ .NET#
- ASP.NET Step by Step#http://ASP.NET Step by Step#
- Programming Microsoft ASP.NET#http://Programming Microsoft ASP.NET#
- Windows Forms Programming in C#
- Applied Microsoft .NET Framework Programming#http://Applied Microsoft .NET Framework Programming#
- .NET Compact Framework Programming with C#
- .NET Framework Essentials#http://.NET Framework Essentials#

CACHING:

Caching is a technique of storing frequently used data/information in memory, so that, when the same data/information is needed next time, it could be directly retrieved from the memory instead of being generated by the application.

Caching is extremely important for performance boosting in ASP.NET, as the pages and controls are dynamically generated here. It is especially important for data related transactions, as these are expensive in terms of response time.

Caching places frequently used data in quickly accessed media such as the random access memory of the computer. The ASP.NET runtime includes a key-value map of CLR objects called cache. This resides with the application and is available via the HttpContext and System.Web.UI.Page.

In some respect, caching is similar to storing the state objects. However, the storing information in state objects is deterministic, i.e., you can count on the data being stored there, and caching of data is nondeterministic.

The data will not be available in the following cases:

- If its lifetime expires,
- If the application releases its memory,
- If caching does not take place for some reason.

You can access items in the cache using an indexer and may control the lifetime of objects in the cache and set up links between the cached objects and their physical sources.

Caching in ASP.Net

ASP.NET provides the following different types of caching:

- **Output Caching**: Output cache stores a copy of the finally rendered HTML pages or part of pages sent to the client. When the next client requests for this page, instead of regenerating the page, a cached copy of the page is sent, thus saving time.
- **Data Caching**: Data caching means caching data from a data source. As long as the cache is not expired, a request for the data will be fulfilled from the cache. When the cache is expired, fresh data is obtained by the data source and the cache is refilled.
- **Object Caching**: Object caching is caching the objects on a page, such as data-bound controls. The cached data is stored in server memory.
- **Class Caching**: Web pages or web services are compiled into a page class in the assembly, when run for the first time. Then the assembly is cached in the server. Next time when a request is made for the page or service, the cached assembly is referred to. When the source code is changed, the CLR recompiles the assembly.
- **Configuration Caching**: Application wide configuration information is stored in a configuration file. Configuration caching stores the configuration information in the server memory.

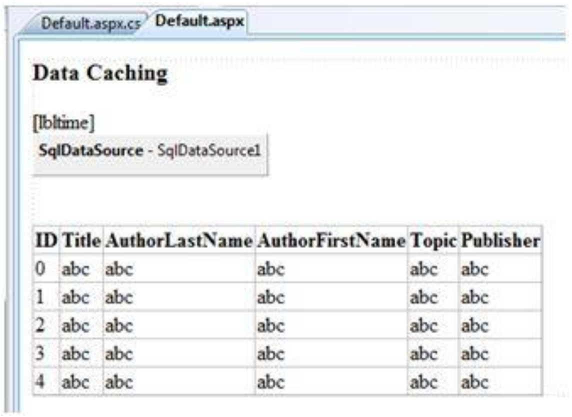
Output Caching

Rendering a page may involve some complex processes such as, database access, rendering complex controls etc. Output caching allows bypassing the round trips to server by caching data in memory. Even the whole page could be cached.

The OutputCache directive is responsible of output caching. It enables output caching and provides certain control over its behaviour.

Syntax for OutputCache directive:

```
<%@ OutputCache Duration="15" VaryByParam="None" %>
```



ID	Title	AuthorLastName	AuthorFirstName	Topic	Publisher
0	abc	abc	abc	abc	abc
1	abc	abc	abc	abc	abc
2	abc	abc	abc	abc	abc
3	abc	abc	abc	abc	abc
4	abc	abc	abc	abc	abc

Put this directive under the page directive. This tells the environment to cache the page for 15 seconds. The following event handler for page load would help in testing that the page was really cached.

```
protected void Page_Load(object sender, EventArgs e)
{
  Thread.Sleep(10000);
  Response.Write("This page was generated and cache at:" +
  DateTime.Now.ToString());
}
```

The **Thread.Sleep()** method stops the process thread for the specified time. In this example, the thread is stopped for 10 seconds, so when the page is loaded for first time, it takes 10 seconds. However, next time you refresh the page it does not take any time, as the page is retrieved from the cache without being loaded.

Data Caching
The main aspect of data caching is caching the data source controls. We have already discussed that the data source controls represent data in a data source, like a database or an XML file. These controls derive from the abstract class DataSourceControl and have the following inherited properties for implementing caching:

- **CacheDuration -** It sets the number of seconds for which the data source will cache data.
- **CacheExpirationPolicy -** It defines the cache behavior when the data in cache has expired.
- **CacheKeyDependency -** It identifies a key for the controls that auto-expires the content of its cache when removed.
- **EnableCaching -** It specifies whether or not to cache the data.

Example
To demonstrate data caching, create a new website and add a new web form on it. Add a SqlDataSource control with the database connection already used in the data access tutorials.

For this example, add a label to the page, which would show the response time for the page.

```
<asp:Label ID="lbltime" runat="server"></asp:Label>
```

Apart from the label, the content page is same as in the data access tutorial. Add an event handler for the page load event:

```
protected void Page_Load(object sender, EventArgs e)
{
  lbltime.Text = String.Format("Page posted at: {0}", DateTime.Now.ToLongTimeString());
}
```

The designed page should look as shown:

When you execute the page for the first time, nothing different happens, the label shows that, each time you refresh the page, the page is reloaded and the time shown on the label changes.
Next, set the EnableCaching attribute of the data source control to be 'true' and set the Cacheduration attribute to '60'. It will implement caching and the cache will expire every 60 seconds.

The timestamp changes with every refresh, but if you change the data in the table within these 60 seconds, it is not shown before the cache expires.

```
<asp:SqlDataSource ID = "SqlDataSource1" runat = "server"
  ConnectionString = "<%$ ConnectionStrings: ASPDotNetStepByStepConnectionString %>"
  ProviderName = "<%$ ConnectionStrings: ASPDotNetStepByStepConnectionString.ProviderName
%>"
  SelectCommand = "SELECT * FROM [DotNetReferences]"
  EnableCaching = "true" CacheDuration = "60">
</asp:SqlDataSource>
```

5. Write the steps how the silver light application executed in a web browser
Steps for rendering Silverlight on a web page
The Plug-in acts in the following sequence when a user navigates to a Page with Silverlight, as shown in the figure 11.3:

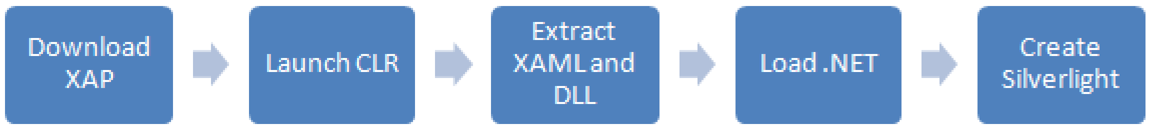


Figure: CLR is launched which then extract XAML and other assemblies.

1. User opens a web page with Silverlight
2. The page has embedded code which refers to a XAML Application (.XAP) file
3. The Silverlight Plug-in is installed in the browser, and then downloads the .XAP file
4. Launch the embedded Common Language Runtime (CLR), the environment which ex- ecutes and manages the Silverlight application
5. Extract the XAML Files and Assemblies from the XAML Application (.XAP) file
6. Load the .NET Assemblies
7. Create instances of the Silverlight Control that includes:
    1. Creating a User Interface Element Tree
    2. Managing the Layout of the User Interface
    3. Drawing the User Interface

Silverlight application is hosted on a web page using an HTML Object tag. We discuss how this works in the next section “Silverlight Host”.

Silverlight Host
Microsoft Visual Studio and Expression Blend both allow you to create Silverlight applications. These applications essentially create a XAML Application file (.XAP) file, referenced by a link in an object tag.

There are three ways you can host a Silverlight Application File in a web page

1. Hosting the Silverlight in any HTML Page using <object> tag
2. Hosting in an ASPX Page using a Silverlight Control
3. Hosting in an ASPX Page using a MediaPlayer Control

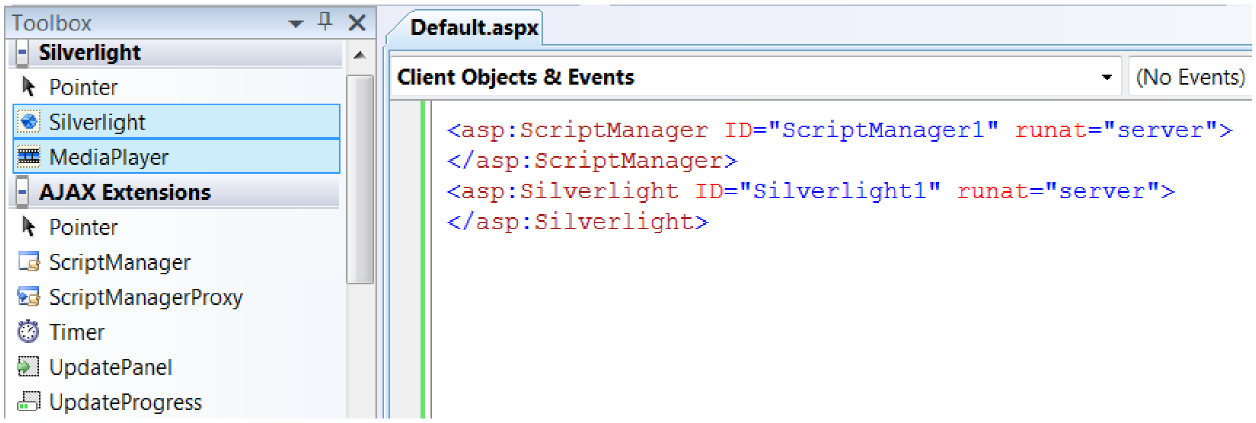
Using an <object> Tag is similar to embedding any ActiveX control like a Media Player or a Flash file. Here is a sample code which embeds a Silverlight Object that link to a HelloWorld.xap.

Hide Copy Code

```
<object data="data:application/x-silverlight-2," type="application/x-silverlight-2">
<param name="source" value="ClientBin/HelloWorld.xap" />
<param name="onError" value="onSilverlightError" />
<param name="background" value="white" />
<param name="minRuntimeVersion" value="4.0.50826.0" />
<param name="autoUpgrade" value="true" />
</object>
```

Visual Studio allows creating web projects that uses Silverlight User Controls in a webform. This user control is eventually converted into an object tag as shown in the earlier example. Silverlight also allows using Media control inside a web form (ASPX).

Silverlight Controls can also be embedded into an existing or new Web application by dragging the user control from the Tool Box to the web page. To use these controls, you will also need a Script- Manager control, the control which manages script files. See figure 11.4.



```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:Silverlight ID="Silverlight1" runat="server">
</asp:Silverlight>
```

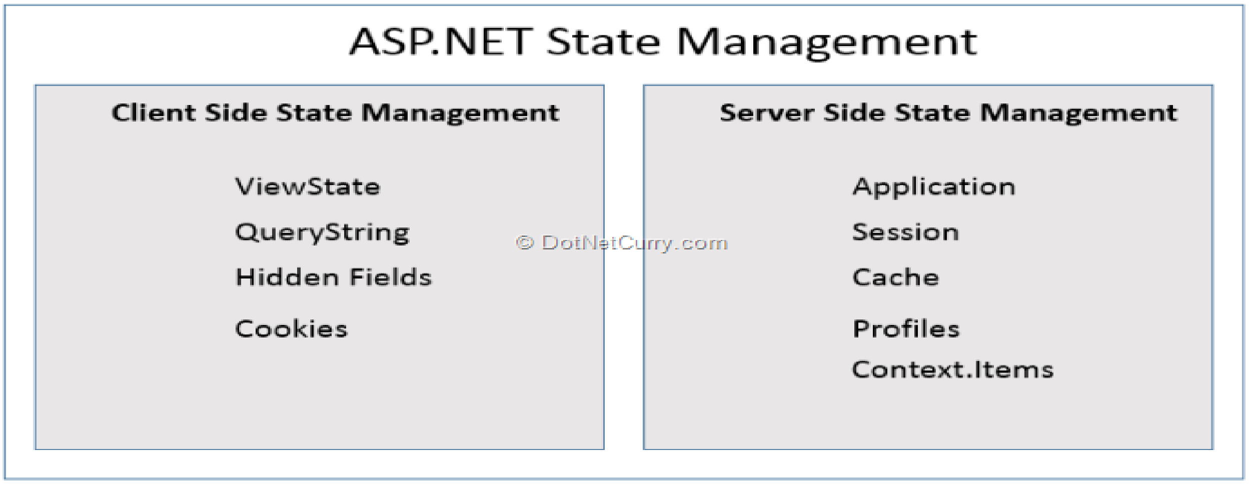
Figure: Silverlight and MediaPlayer control are available in the Toolbox.

Here is an example using a Silverlight Control in a Web Form (ASPX)
<asp:Silverlight ID=”Xaml1” runat=""”server”" Source=”HelloWorld.xap” Minimum-

Version=”4.0.50826.0” Width=”100%” Height=”100%” />

Using a MediaPlayer Control in a Web Form (ASPX)
<asp:MediaPlayer ID=”MediaPlayer1” runat=""”server”" Height=”240px” Width=”320px”>

6. What is state management? Explain Application state and Viewstate with suitable example.



Client Side State Management
Client-side management techniques don't utilize server resources. They are less secure but fast performing. Let's explore some client-side techniques to save state:

ViewState - ViewState is ASP.NET's own integrated state serialization mechanism. ViewState objects are used to store page related data which are preserved between the round trips from client to server and server to client. View state is maintained as a hidden field in the page. Here's how viewstate for an ASP.NET textbox would look like if you view the page source:

```
<input type="hidden" ID="__VIEWSTATE" name="__VIEWSTATE"
value="dDw3GTg2NTI5MDEDC0y3=" />
```

Let's see ViewState in action with an example. Design a simple web form and add a button on the web form. Write the following code in the code behind of the web form -

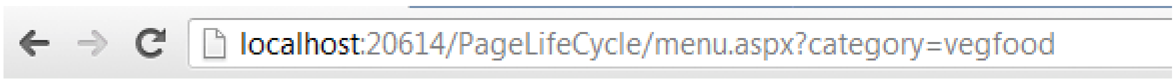
```
protected void Page_Load(object sender, EventArgs e)
{
if (!IsPostBack)
{
ViewState["PageHitCounter"] = 0;
}
}
protected void btnPageHitCounter_Click(object sender, EventArgs e)
{
ViewState["PageHitCounter"] = Convert.ToInt32(ViewState["PageHitCounter"]) + 1;
Response.Write("The number of postbacks are - " + ViewState["PageHitCounter"].ToString());
}
```

This code shows how many times the page has been posted back to the server. Now normally with every rountrip the *PageHitCounter* would be reset to 0, but since we are using ViewState, the counter gets preserved between round trips.

Query String - Query String is passed in the URL. They store the value in the form of Key-Value pair. You will have to maintain the Query Strings when you pass it from one page to second page and second page to the third page.

For example, when redirecting a request from one page to another page, you can pass the Query String as shown here -

Response.Redirect("menu.aspx?category=vegfood");



localhost:20614/PageLifeCycle/menu.aspx?category=vegfood

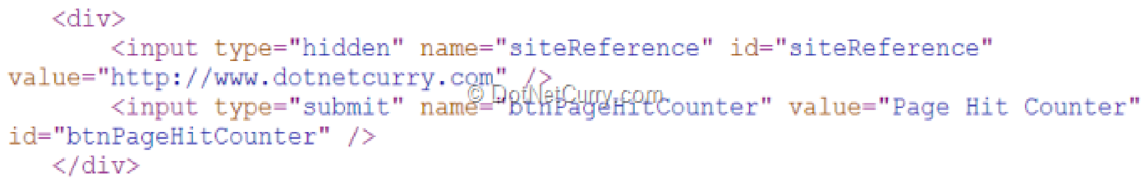
Query string is limited to simple string information, and they're easily accessible and readable.

Hidden Fields - Hidden Fields are similar to a text box but does not get displayed on the UI. However you can see the value of hidden fields when you view page source in the browser. Using Hidden Fields, you can pass information from one page to another page without the end user's knowledge, primarily for internal processing.

Drag and Drop a hiddenfield control on the web page from the Visual Studio toolbox and set its value in the Page Load event as shown below -

```
protected void Page_Load(object sender, EventArgs e)
{
if (!IsPostBack)
{
siteReference.Value = "http://www.dotnetcurry.com";
}
}
```

Run the page and check the source code. You will see the Hidden Field value as shown below -



```
    <div>
        <input type="hidden" name="siteReference" id="siteReference" 
value="http://www.dotnetcurry.com" />
        <input type="submit" name="btnPageHitCounter" value="Page Hit Counter" 
id="btnPageHitCounter" />
    </div>
```

Cookies - Cookies are small files that are created in the web browser's memory or on the client's hard disk. The maximum size of a cookie file is 4 KB. You can create two types of cookies. Transient Cookie (memory) and Persistent Cookie (hard disk). Transient Cookies are accessible till the time the browser is running. Persistent Cookies are cookies which have an expiry time. When you don't set expiry time for the cookie, the cookies are treated as transient cookies.

Cookie contains a Key/Value pair. Cookie can also contain the collection of key/value pairs. Let's see an example of a Cookie -

```
Response.Cookies["CityName"].Value = "London";
HttpCookie dncCookie = new HttpCookie("DotNetCurry");
dncCookie.Values["ArticleName"] = "SharePoint 2013 - Business Connectivity Services";
dncCookie.Values["Author"] = "SharePoint Admin";
dncCookie.Values["PublishDate"] = DateTime.Now.AddDays(10).ToLongDateString();
dncCookie.Values["ArticleName"] = "SharePoint 2013 - Business Connectivity Services";
dncCookie.Expires = DateTime.Now.AddDays(20);
Response.AppendCookie(dncCookie);
```

Just like query strings, cookies are limited to storing simple strings. Users also disable cookies or manually delete them which renders them ineffective.

Server Side State Management

Unlike Client-side state management techniques, Server-side options are more secure but can become slow to respond, depending on the information that is stored. Since these techniques directly use the resources of the web server, scalability issues is something to be considered when storing large amount of data.

ASP.NET provides some options to implement server-side state management:

Application State - The Application object is an instance of the System.Web.HttpApplicationState class. You can create Application Level object like a name/value pair and share it across all users. You can declare Application level variable when the web application starts. Usually developers declare Application Level variables in Global.asax file. The Global.asax file contains events (Application_Start Event, Application_End Event and Application_Error Event) which get associated with Application object.

You can declare Application level objects in the Application_Start event. Let's see an example -

Add a Global.asax file in your web application and write the following code in the Application_Start event -

```
void Application_Start(object sender, EventArgs e)
{
Application["CopyRightNote"] = "DotNetCurry CopyRight 2013-2014";
Application["HitCounter"] = 0;
}
void Session_Start(object sender, EventArgs e)
{
Application.Lock();
Application["HitCounter"] =Convert.ToInt32(Application["HitCounter"]) + 1;
Application.Lock();
}
```

Now you can access both application variables on any page.

Session State – Session objects are stored on the server side and are an instance of the System.Web.SessionState.HttpSessionState class. It can be used to store any type of data in the memory. They are primarily used for storing user specific information like shopping cart, preferences etc and is never transferred to the client. You can declare session level variables in the following manner -

```
Session["AccountNo"] = 1998773887324556;
```

By default sessions are stored in server memory called as In-Process. You can also store session as out-of-process. You can store the session information in Microsoft SQL Server or State Server. Session state must be used thoughtfully as it leads to scalability issues if misused.

Cache - The Cache object is an instance of the System.Web.Caching.Cache class. Cache is stored on the server side and is more scalable in nature, as ASP.NET removes objects if the memory becomes scarce. This also makes it unreliable in some cases. Cache objects can have expiration polices set on them and is shared across users. You can implement Page Caching and Data Caching. Let's see a small example of both.

Page Caching -

Data Caching -

```
Cache.Insert("CacheDateTime", DateTime.Now);
```

You can also cache client-side by using the Location=”Client” option.

Profile - Profile data is stored in the SQL Server database by default. This database structure is preset, so if you want any custom user details to be stored, you will need to create a custom database and write a custom provider for it. Profiles are designed to store information permanently. Here’s a simple example of using profiles:

```
<profile>
<properties>
<add name="FirstName" type="System.String"/>
<add name="LastName" type="System.String"/>
<add name="Address" type="System.String"/>
<add name="City" type="System.String"/>
<add name="Age" type="System.Int32"/>
</properties>
</profile>
```

Using the profile properties in your application -

```
Profile.FirstName = "John";
Profile.LastName = "Mark";
Profile.City = "London";
Profile.Age = 46;
```

The Profile object provides you with a persistent form of session state that is strongly typed.

Context.Items - The HttpContext object is provided by the Page.Context property. The HttpContext.Items collection can be used to *temporarily* store data across postback. View state and session state can be used for a similar effect, but they assume longer-term storage. Context can be used for storing data for one request only. We can store it as a key/value pair as shown below -

```
Context.Items["SocialPinNo"] = 3666736;
Response.Write(Context.Items["SocialPinNo"].ToString());
```

7. Write the short note on WCF, Silverlight and WF.

WCF:

WCF stands for Windows Communication Foundation. The elementary feature of WCF is interoperability. It is one of the latest technologies of Microsoft that is used to build service-oriented applications. Based on the concept of message-based communication, in which an HTTP request is represented uniformly, WCF makes it possible to have a unified API irrespective of diverse transport mechanisms.

WCF was released for the first time in 2006 as a part of the .NET framework with Windows Vista, and then got updated several times. WCF 4.5 is the most recent version that is now widely used.

A WCF application consists of three components:

- WCF service,
- WCF service host, and
- WCF service client.

WCF platform is also known as the Service Model.

Fundamental Concepts of WCF

Message

This is a communication unit that comprises of several parts apart from the body. Message instances are sent as well as received for all types of communication between the client and the service.

Endpoint

It defines the address where a message is to be sent or received. It also specifies the communication mechanism to describe how the messages will be sent along with defining the set of messages. A structure of an endpoint comprises of the following parts:

- **Address -** Address specifies the exact location to receive the messages and is specified as a Uniform Resource Identifier (URI). It is expressed as scheme://domain[:port]/[path]. Take a look at the address mentioned below:

net.tcp://localhost:9000/ServiceA

Here, 'net.tcp' is the scheme for the TCP protocol. The domain is 'localhost' which can be the name of a machine or a web domain, and the path is 'ServiceA'.

- **Binding -** It defines the way an endpoint communicates. It comprises of some binding elements that make the infrastructure for communication. For example, a binding states the protocols used for transport like TCP, HTTP, etc., the format of message encoding, and the protocols related to security as well as reliability.

- **Contracts -** It is a collection of operations that specifies what functionality the endpoint exposes to the clinet. It generally consists of an interface name.

Hosting

Hosting from the viewpoint of WCF refers to the WCF service hosting which can be done through many available options like self-hosting, IIS hosting, and WAS hosting.

Metadata

This is a significant concept of WCF, as it facilitates easy interaction between a client application and a WCF service. Normally, metadata for a WCF service is generated automatically when enabled, and this is done by inspection of service and its endpoints.

WCF Client

A client application that gets created for exposing the service operations in the form of methods is known as a WCF client. This can be hosted by any application, even the one that does service hosting.

Channel
Channel is a medium through which a client communicates with a service. Different types of channels get stacked and are known as Channel Stacks.
SOAP
Although termed as 'Simple Object Access Protocol', SOAP is not a transport protocol; instead it is an XML document comprising of a header and body section.
Advantages of WCF

- It is interoperable with respect to other services. This is in sharp contrast to .NET Remoting in which both the client and the service must have .Net.
- WCF services offer enhanced reliability as well as security in comparison to ASMX (Active Server Methods) web services.
- Implementing the security model and binding change in WCF do not require a major change in coding. Just a few configuration changes is required to meet the constraints.
- WCF has built-in logging mechanism whereas in other technologies, it is essential to do the requisite coding.
- WCF has integrated AJAX and support for JSON (JavaScript object notation).
- It offers scalability and support for up-coming web service standards.
- It has a default security mechanism which is extremely robust.

Silverlight
Silverlight is a powerful cross-browser & cross-platform technology for building the next generation web experience & rich internet applications for the web. You can run Silverlight in most of all the popular browsers like Internet Explorer, Firefox, Chrome, Safari etc. Silverlight can run in various devices and operating systems like Windows, Apple Mac OS-X and Windows Phone 7. Using Silverlight you can create rich, visually stunning web applications like flash. Also you can create smooth animations using Storyboards; you can stream media over the net etc.
Silverlight web browser plug-in comes as free to install (approximately 4-5 MB in size). You can download the required plug-in from Microsoft Silverlight Site.

8. Explain in detail about various ASP.NET webform controls
All server controls must appear within a <form> tag, and the <form> tag must contain the runat="server" attribute. The runat="server" attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts:

```
<form runat="server">
...HTML + server controls
</form>
```

If you select view source in an .aspx page containing a form with no name, method, action, or id attribute specified, you will see that ASP.NET has added these attributes to the form. It looks something like this:

```
<form name="_ctl0" method="post" action="page.aspx" id="_ctl0">
...some code
</form>
```

Submitting a Form

A form is most often submitted by clicking on a button. The Button server control in ASP.NET has the following format:

```
<asp:Button id="id" text="label" OnClick="sub" runat="server" />
```

The id attribute defines a unique name for the button and the text attribute assigns a label to the button. The onClick event handler specifies a named subroutine to execute.

Example:

```
<script  runat="server">
Sub submit(Source As Object, e As EventArgs)
   button1.Text="You clicked me!"
End Sub
</script>
<!DOCTYPE html>
<html>
<body>

<form runat="server">
<asp:Button id="button1" Text="Click me!" runat="server" OnClick="submit" />
</form>
</body>
</html>
```

9. What is Master Page? Give ASP.Net code to demonstrate the usage of nested master page.
What do you mean by data bound controls? Explain DataList control with code to display list of students with enrolmentno and branch.
Master pages allow you to create a page layout — a template page — and then create separate pages containing content that is merged with the master page at run time.

Code for nested Master Page

```
<%@ Master Language="VB" CodeFile="Site_Main.master.vb" Inherits="MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head id="Head1" runat="server">
    <style type="text/css">
        html
        {
            background-color:Silver;
            font:14px Georgia,Serif;
        }
        .content
        {
            width:700px;
            margin:auto;
            border-style:solid;
            background-color:white;
            padding:10px;
        }

        .tabstrip
        {
            padding:3px;
            border-top:solid 1px black;
            border-bottom:solid 1px black;
        }

        .tabstrip a
        {
            font:14px Arial;
            color:Fuchsia;
            text-decoration:none;
        }
        .column
        {
            float:left;
            padding:10px;
            border-right:solid 1px black;
        }
        .rightColumn
        {
            float:left;
            padding:10px;
        }
        .footer
        {
          background-color:Lime;
            border:3px dotted red;
            text-align:center;
        }
       .clear
        {
            clear:both;
        }
    </style>
    <title>Website Main Master Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div class="content">
        <asp:Image
            id="imgLogo"
            ImageUrl="~/adds/CSSiteLogo.gif"
            AlternateText="Logo"
            Runat="server" />
        <div class="tabstrip">
        <asp:HyperLink
            id="lnkProducts"
            Text="HOME"
            NavigateUrl="~/Column1_Con_Home.aspx"
            Runat="server" />
        &nbsp;|&nbsp;
        <asp:HyperLink
            id="lnkServices"
            Text="FORUM"
            NavigateUrl="~/Column2_Con_About.aspx"
            Runat="server" />
        </div>
        <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
        </asp:contentplaceholder>
        <br class="clear" />
        <br />
        <br />
        <div class="footer">
        Copyright &copy; 2010 by CompanyName

        </div>
    </div>
    </form>
</body>
</html>
```

There are two Master Pages Column1.master and Column2.master given below which are nested Master Pages. Both Master Pages include a MasterPageFile attribute that points to the Site.master Master Page.

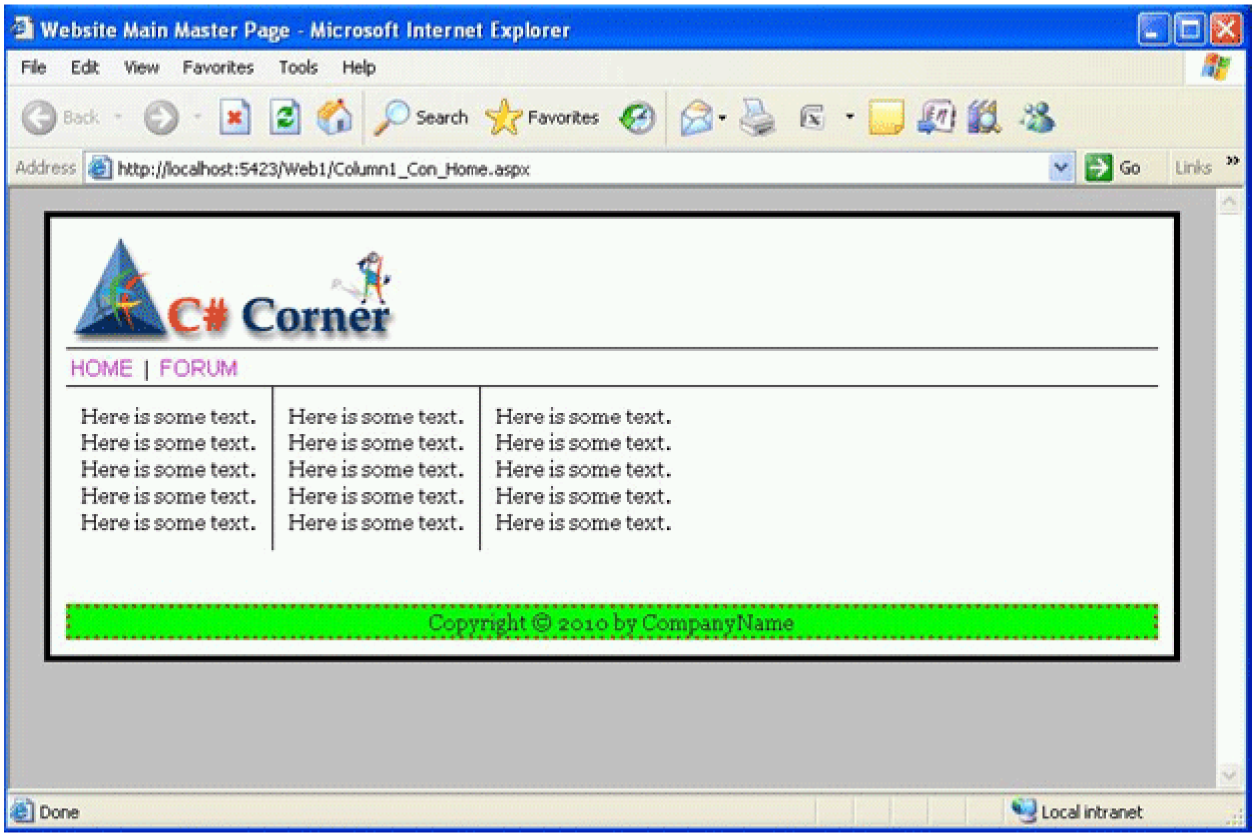
Column1.Master

```
<%@ Master Language="VB" MasterPageFile="~/Site_Main.master" AutoEventWireup="false"CodeFil
e="Column1.master.vb" Inherits="Column1" %>
<asp:Content
    id="Content1"
    ContentPlaceHolderID="ContentPlaceHolder1"
    Runat="server">
    <div class="column">
        <asp:ContentPlaceHolder
            id="ContentPlaceHolder1"
            Runat="server" />
    </div>
    <div class="column">
        <asp:ContentPlaceHolder
            id="ContentPlaceHolder2"
            Runat="server" />
    </div>
    <div class="rightColumn">
        <asp:ContentPlaceHolder
            id="ContentPlaceHolder3"
            Runat="server" />
    </div>
</asp:Content>
```

Column2.Master

```
<%@ Master Language="VB" MasterPageFile="~/Site_Main.master" AutoEventWireup="false"CodeFil
e="Column2.master.vb" Inherits="Column2" %>
<asp:Content
    id="Content1"
    ContentPlaceHolderID="ContentPlaceHolder1"
    Runat="server">
    <div class="column">
        <asp:ContentPlaceHolder
            id="ContentPlaceHolder1"
            Runat="server" />
    </div>
    <div class="rightColumn">
        <asp:ContentPlaceHolder
            id="ContentPlaceHolder2"
            Runat="server" />
    </div>
</asp:Content>
```

The Master Pages given above create three-column and two-column pages layouts. The Column1_Con_Home.aspx page given below uses the Column1.master Master Page. When we request a HOME page, the contents of Site_Main.master, Column1.master and Column1_Con_Home.aspx are combined to generate the rendered output.


Website Main Master Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:5423/Web1/Column1_Con_Home.aspx
Go Links
C# Corner
HOME | FORUM
Done
Local intranet

Here is some text. Here is some text. Here is some text.
Here is some text. Here is some text. Here is some text.
Here is some text. Here is some text. Here is some text.
Here is some text. Here is some text. Here is some text.
Here is some text. Here is some text. Here is some text.

Copyright © 2010 by CompanyName

Column1_Con_Home.aspx Page

```
<%@ Page Title="" Language="VB" MasterPageFile="~/Column1.master" AutoEventWireup="false"CodeFile="Column1_Con_Home.aspx.vb" Inherits="Column1_Con_Home" %>
<asp:Content
    ID="Content1"
    ContentPlaceHolderID="ContentPlaceHolder1"
    Runat="Server">
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
</asp:Content>
<asp:Content
    ID="Content2"
    ContentPlaceHolderID="ContentPlaceHolder2"
    Runat="Server">
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
</asp:Content>
<asp:Content
    ID="Content3"
    ContentPlaceHolderID="ContentPlaceHolder3"
    Runat="Server">
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
    <br />
    Here is some text.
</asp:Content>
```

Column2_Con_About.aspx Page

```
<%@ Page Title="" Language="VB" MasterPageFile="~/Column2.master" AutoEventWireup="false"Co
deFile="Column2_Con_About.aspx.vb" Inherits="Column2_Con_About" %>
<asp:Content
    ID="Content1"
    ContentPlaceHolderID="ContentPlaceHolder1"
    Runat="Server">
    Here is list of forum questions.
    <br />
    Here is list of forum questions.
    <br />
    Here is list of forum questions.
    <br />
    Here is list of forum questions.
    <br />
    Here is list of forum questions.
```

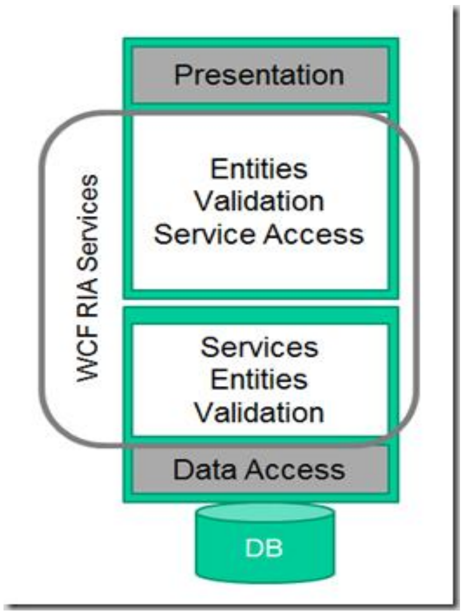
```
</asp:Content>
<asp:Content
   ID="Content2"
   ContentPlaceHolderID="ContentPlaceHolder2"
   Runat="Server">
   Here is list of forum questions.
   <br />
   Here is list of forum questions.
   <br />
   Here is list of forum questions.
   <br />
   Here is list of forum questions.

   <br />
   Here is list of forum questions.
</asp:Content>
```

10. Explain how to use WCF RIA services with silver light

In order to build serious business application in Silverlight (and other client technologies), you have to work with a lot of data. And that data is usually not resident on the client machine, it is usually distributed amongst many clients and is stored and operated on by back-end services. If you try to write this kind of application architecture yourself, you have to tackle a lot of technologies and write a lot of plumbing. In the end, most of what you are doing is pushing and pulling data from the client to the back end and invoking operations on the server from the Silverlight client application. What would be great is if most of that plumbing and push-pull logic could be automated for you, allowing you to just focus on what data you need, the rules that surround the manipulation of that data, and how to present it in the client application.

This is exactly what WCF RIA Services does for you. WCF RIA Services is a new part of the .NET 4 and Silverlight 4 frameworks that lets you quickly build N-Tier Silverlight client applications without needing to focus on the service plumbing to get data into and out of your client application from back-end services, logic and data access. This article and the subsequent articles in this series will get you up and running with all the capabilities of WCF RIA Services.

RIA Services helps you to write one set of server code, but have appropriate parts of that service code available on the client without having to duplicate it or write client side code to access it. On the server side, WCF RIA Services helps you define your services, domain entities, and supporting logic. On the client side, WCF RIA Services code generates corresponding classes that let you easily call those services, have the same entities available and populated on the client side, along with supporting validation logic and other kinds of code that you can share between the client and the service side. The diagram below shows what part of your architecture WCF RIA Services focuses on.



key concepts in WCF RIA Services:

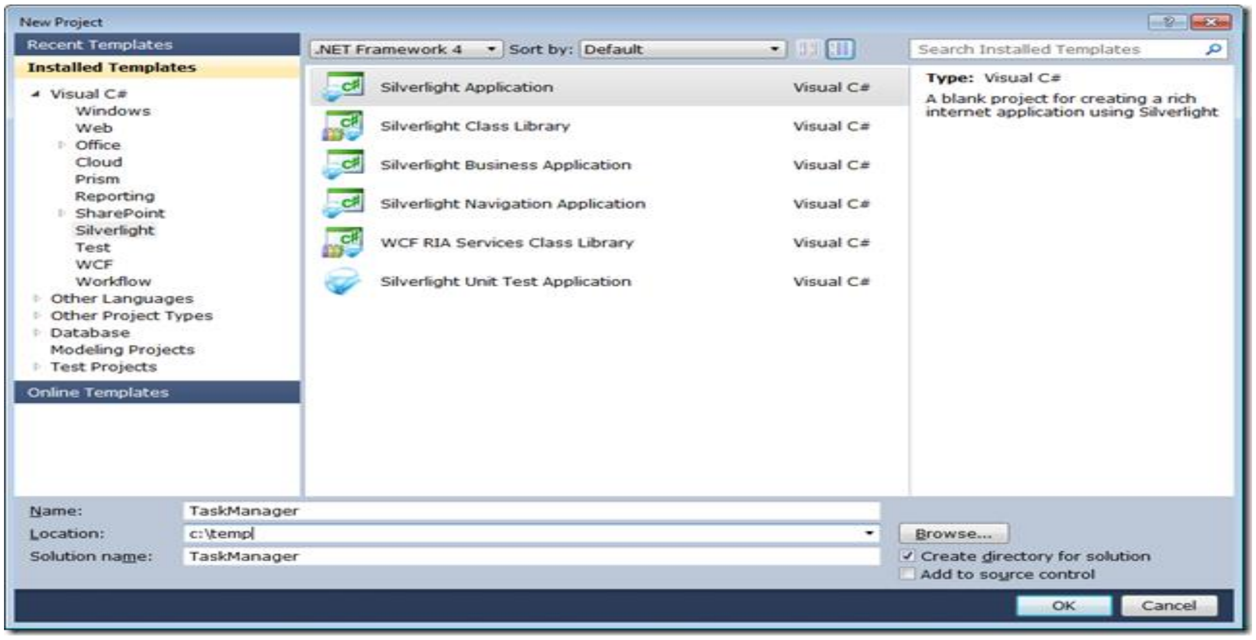
- Project Links – A link between a Silverlight client project and a server web application or class library. The server project defines or references the domain services, entities, metadata, and shared code defined on the server side. As a result of the link, appropriate code is generated in the client project at compile time.
- Domain Services – This is the core construct of WCF RIA Services. A domain service defines the operations supported on the server side. These are usually mostly focused on CRUD operations against entity types, but can also be arbitrary operations to be invoked on the server from the client. These operations are exposed automatically via WCF and can be called from the client generated code without needing to know much at all about WCF.
- Entities – you define entitiy types on the server, and a client-side definition of the same entity type is generated for use on the client side. You can add validation and other kinds of metadata to the the entity definition which will affect the code generated on the client side, allowing you to just maintain a single server-side definition. The entity types on the client and server side are used to serialize and deserialize the data across the WCF service. The entity types can be created with Entity Framework, LINQ to SQL, or can be Plain Old CLR Objects (POCOs) that you define yourself.
- Domain Context – This is the client side counterpart to the domain service. It is generated code on the client side that gives you easy access to the functionality that resides on the server side. Internally it contains a WCF proxy that makes the service calls, and it also manages creating the queries that get executed on the server side, the change tracking for the entities being manipulated on the client and more.
- DomainDataSource – This is a data source object that can be used to execute queries and submit changes to the server side. It gets associated with the domain context and makes calls through that domain context on your behalf to query and update entities. It facilitate direct data binding from XAML or can be used behind the scenes as well.

Step 1: Create the Silverlight Application Project and Link to the Server Project

The application I'll build in this series is based on a simple database that stores task and related entity information. To follow along with the steps, you will need to run the TaskManager.sql script in the download sample to create the database schema and populate it with a few sample entities.

Once you have the database in place, you are ready to start creating the application. You'll also need to have the Silverlight 4 Tools installed for Visual Studio 2010, which installs WCF RIA Services support as well.

Create a new Silverlight Application project named TasksManager. You can use WCF RIA Services with any of the Silverlight project types, including the Silverlight Business Application template that is added by WCF RIA Services itself. But to start simple, I'll go with just the Silverlight Application template.

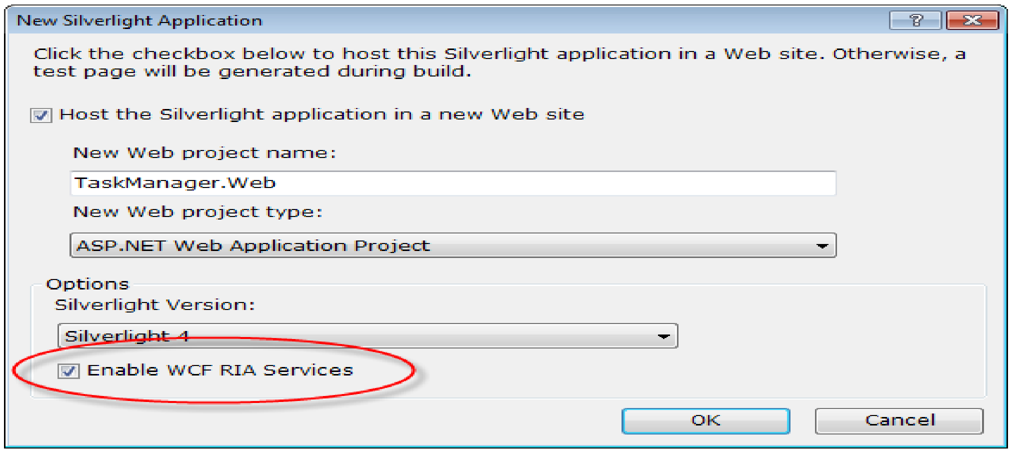

New Project

Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
Silverlight Business Application	Visual C#
Silverlight Navigation Application	Visual C#
WCF RIA Services Class Library	Visual C#
Silverlight Unit Test Application	Visual C#

Type: Visual C#
A blank project for creating a rich internet application using Silverlight

Name: TaskManager
Location: c:\temp
Solution name: TaskManager
✓ Create directory for solution
Add to source control

After you click OK in the new project window, you will be presented with the normal dialog to create a hosting web application project. What you will see that is new is a checkbox at the bottom to Enable WCF RIA Services. Check that box to create the link between the client project and the server project so that WCF RIA Services can code generate the appropriate client code based on the domain services and entities you will create.


New Silverlight Application

Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

✓ Host the Silverlight application in a new Web site

New Web project name: TaskManager.Web

New Web project type: ASP.NET Web Application Project

Options

Silverlight Version: Silverlight 4

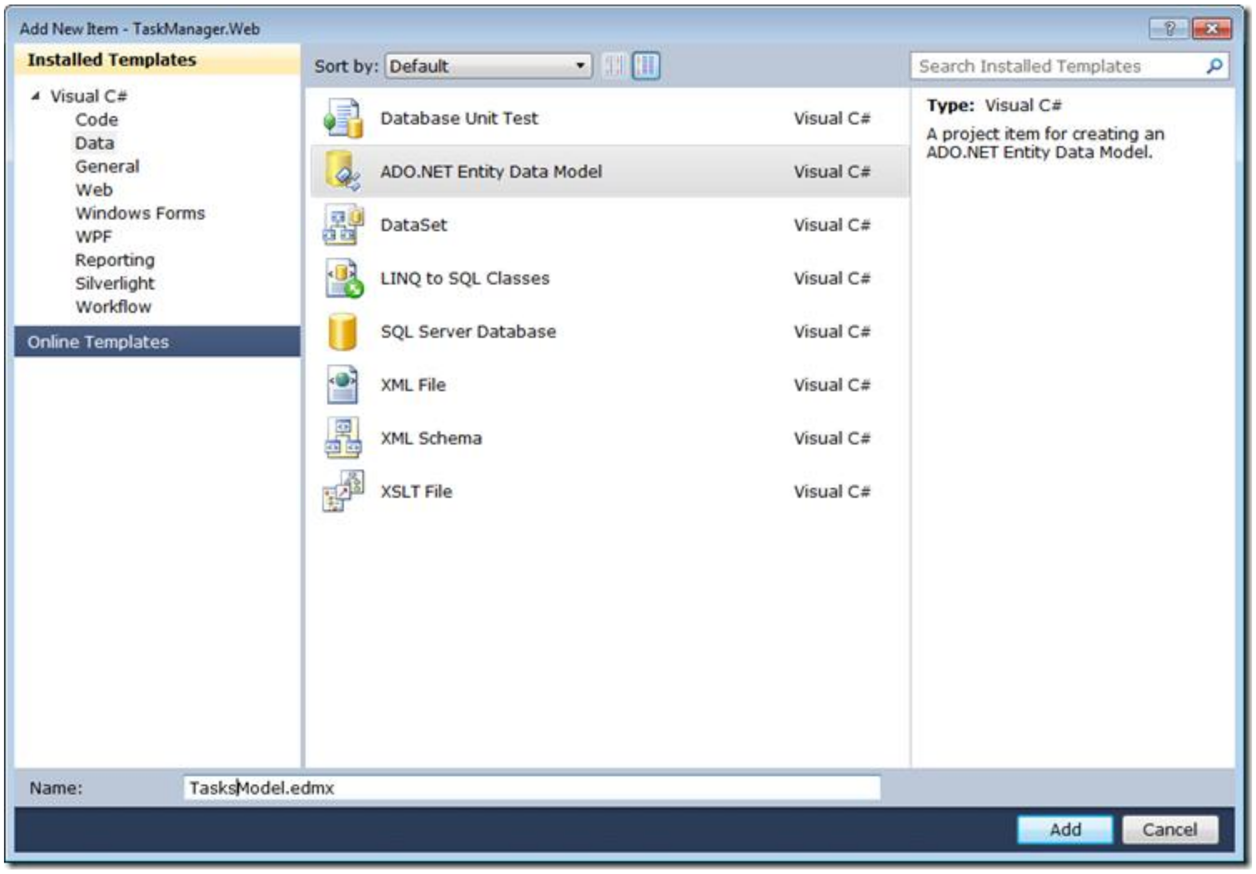
✓ Enable WCF RIA Services

Click OK and the projects will be created.

Step 2: Create the Domain Model Entities

WCF RIA Services is mostly about moving data back and forth between the client and server for you. So you need some data to work with. Out of the box, Entity Framework is supported best. With the WCF RIA Services Toolkit, there is also support for LINQ to SQL. And you can create your own domain services if you want to work with POCOs. For this article, you will use Entity Framework.

Right click on the TaskManager.Web server project in Solution Explorer and select Add > New Item, pick the Data category on the left, and select ADO.NET Entity Data Model, and name it TasksModel.edmx.



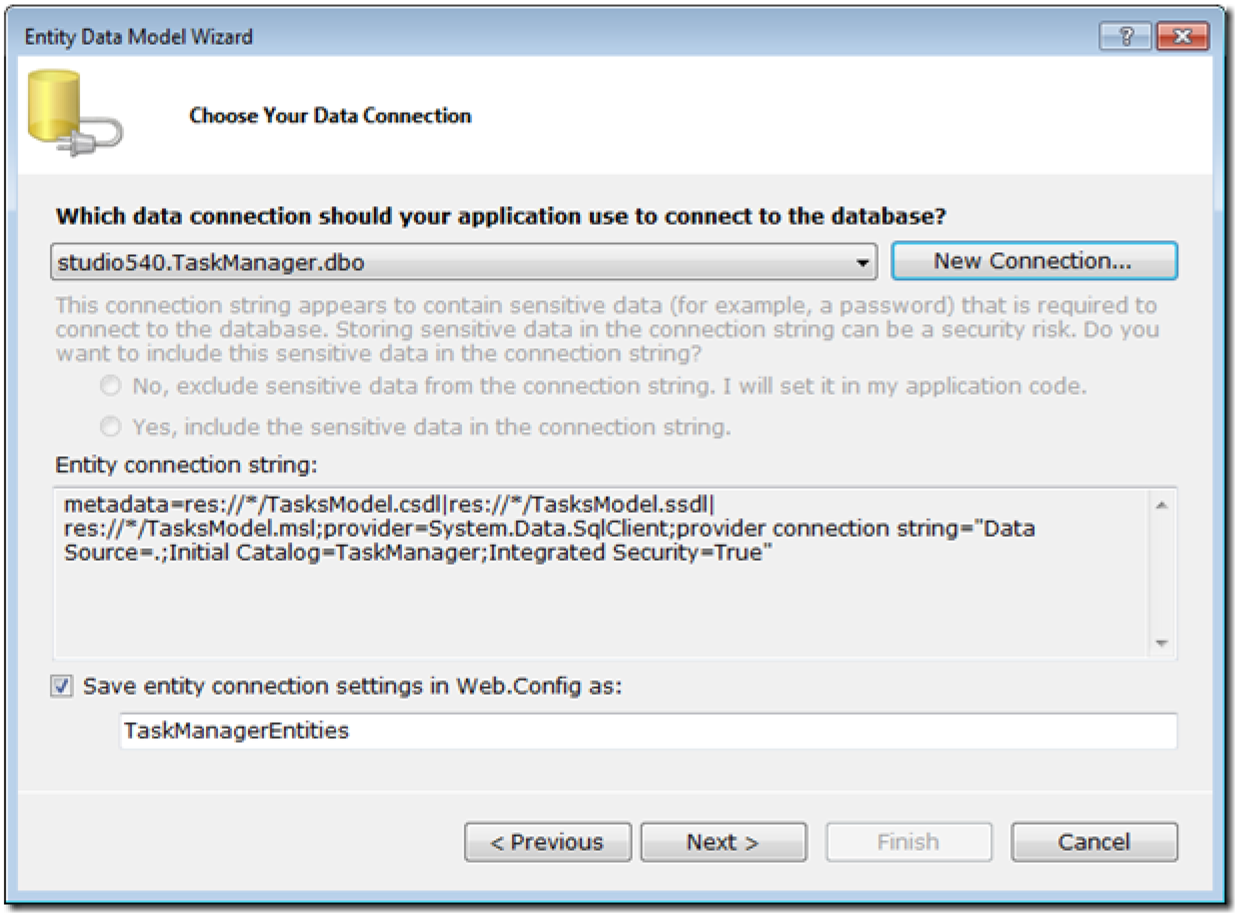
Database Unit Test	Visual C#
ADO.NET Entity Data Model	Visual C#
DataSet	Visual C#
LINQ to SQL Classes	Visual C#
SQL Server Database	Visual C#
XML File	Visual C#
XML Schema	Visual C#
XSLT File	Visual C#

Type: Visual C#

A project item for creating an ADO.NET Entity Data Model.

Name: TasksModel.edmx

Click Add and you will see the Entity Data Model Wizard. In the first step, leave the option to Generate from database selected and click Next. Create a New Connection in the next step to the TaskManager database you created with the sample code SQL script at the beginning.



Which data connection should your application use to connect to the database?

studio540.TaskManager.dbo

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

No, exclude sensitive data from the connection string. I will set it in my application code.

Yes, include the sensitive data in the connection string.

Entity connection string:

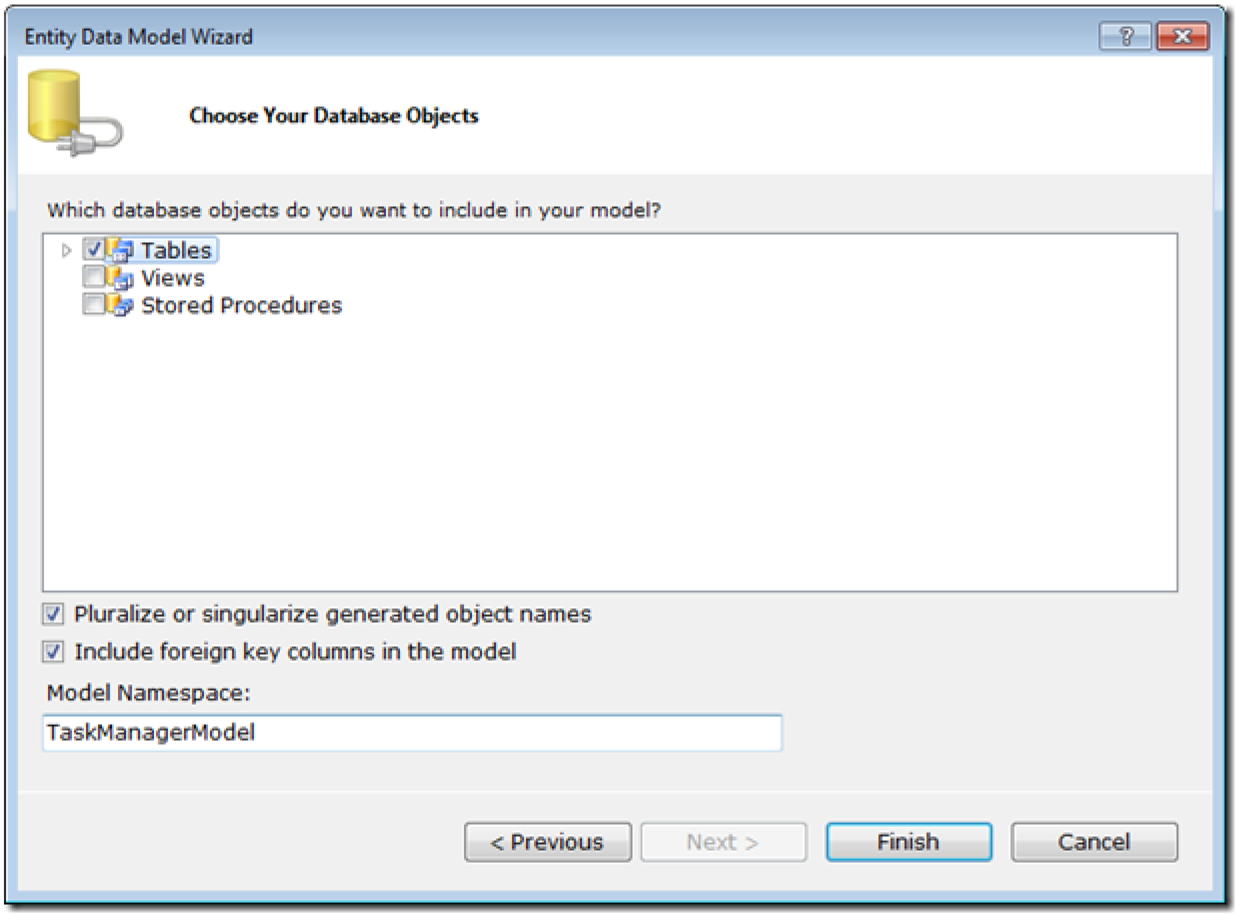
```
metadata=res://*/TasksModel.csdl|res://*/TasksModel.ssdl|
res://*/TasksModel.msl;provider=System.Data.SqlClient;provider connection string="Data
Source=.;Initial Catalog=TaskManager;Integrated Security=True"
```

☑ Save entity connection settings in Web.Config as:

TaskManagerEntities

The connection name should default to TaskManagerEntities, which is fine. Click Next.

In the next step, Choose Your Database Objects, check the box for Tables to select all the tables and click Finish.


Entity Data Model Wizard

Choose Your Database Objects

Which database objects do you want to include in your model?

- [✓] Tables
- [ ] Views
- [ ] Stored Procedures

[✓] Pluralize or singularize generated object names

[✓] Include foreign key columns in the model

Model Namespace: TaskManagerModel

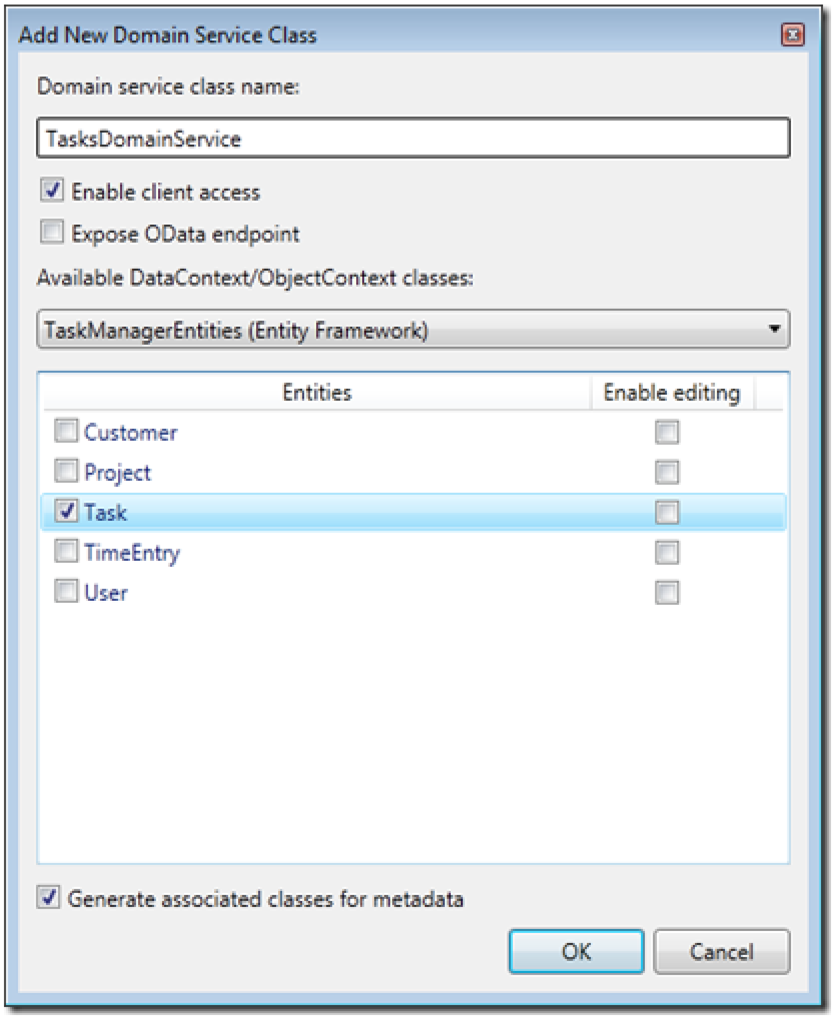
You now have an Entity Framework data model for all the tables in the database. In this article, you will only be working with the Task table, but in future articles in the series you will work with one or two other types.

Build the solution to make sure all is well. If you forget to build after adding your entity data model, it will not show up when creating the domain service in the next step.

Step 3: Define a Domain Service

So far, the only thing specific to WCF RIA Services you have done is check the box to create the link between the client and server project. Now you will do the core activity around RIA Services: define a domain service. Right click on the TaskManager.Web project and select Add > New Item. Select the Web category, and select Domain Service Class. Name the class TasksDomainService.

Next you will be presented with a dialog that lets you select a number of options for your service, including the most important option of what entities it will expose.


Add New Domain Service Class
Domain service class name:
Available DataContext/ObjectContext classes:
Entities
Enable editing
OK
Cancel

Domain service class name: TasksDomainService

[✓] Enable client access

[ ] Expose OData endpoint

Available DataContext/ObjectContext classes: TaskManagerEntities (Entity Framework)

Entities	Enable editing
[ ] Customer	[ ]
[ ] Project	[ ]
[✓] Task	[ ]
[ ] TimeEntry	[ ]
[ ] User	[ ]

[✓] Generate associated classes for metadata

you will just be working with Tasks, so select that entity type and leave the rest of the defaults selected. I'll be getting into more detail on some of the other options and what they do in other articles. Click OK and your domain service will be created.

Minus a bunch of comments they insert, the resulting class looks like this:

```
[EnableClientAccess()]
public class TasksDomainService : LinqToEntitiesDomainService<TaskManagerEntities>
{
    public IQueryable<Task> GetTasks()
    {
        return this.ObjectContext.Tasks;
    }
}
```

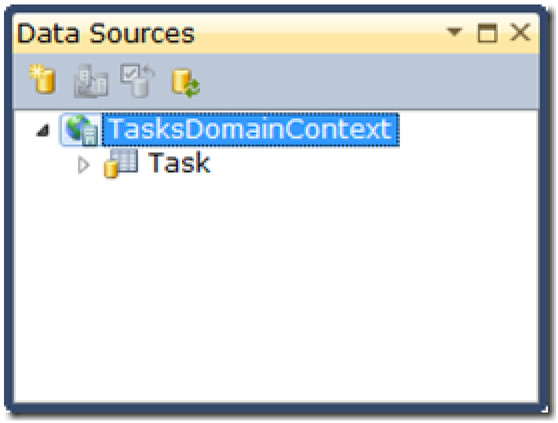
The LinqToEntitiesDomainService<T> class provides the glue between an entity data model and the services that will be exposed by WCF RIA Services. The EnableClientAccess attribute is what causes the client side code to be generated at compile time in the linked client project. From there, you add methods to your domain service to perform CRUD operations on the entities and possibly just expose operations that can be invoked from the client as well.

In this case, because you did not select the Enable Editing option in the previous dialog, the initial code only contains a single query method. I'll get into more detail on the options and variations you have for query methods in the next article. But the one generated here uses convention to indicate that it is a query method based on the return type of IQueryable<T> and the method name. It just delegates to an instance of the entity data model ObjectContext that is created as a member by the base class and uses it to return all the Tasks. If you want to modify that query to order, filter, or page the results, as long as you return an IQueryable<T>, WCF RIA Services will be able to expose that as a service and generate the client code to consume it.

Rebuild the solution to make sure all is well and to generate the client side code. If you Show All Files in the TaskManager project, you will see a Generated_Code folder with a TaskManager.Web.g.cs file in it. This contains the generated code. I'll dig into what is there in more detail in the next article.

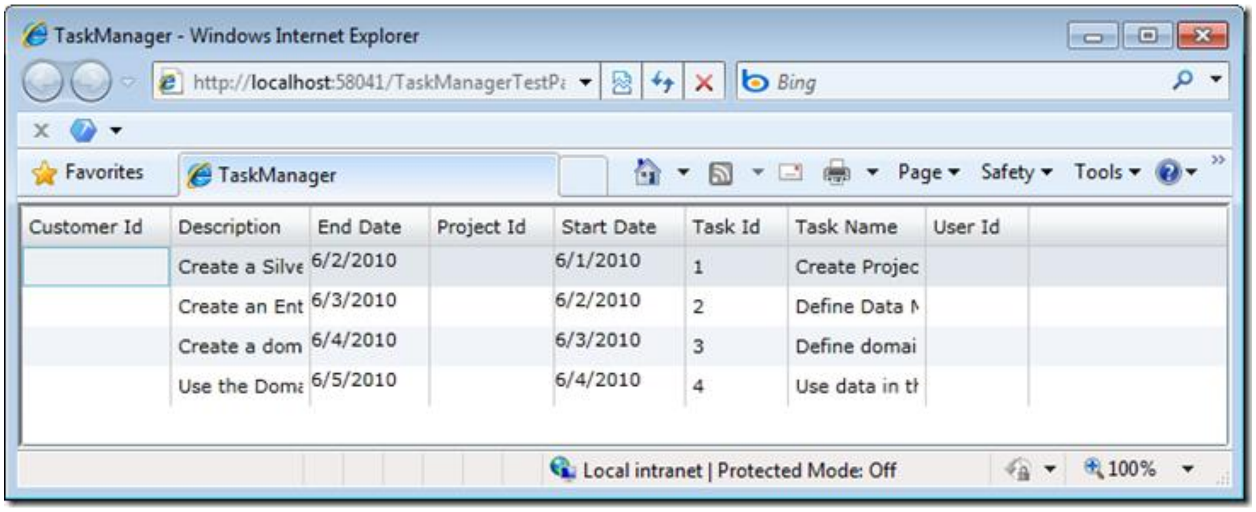
Step 4: Retrieve Data into the UI with DomainDataSource
For this step, you will leverage the drag and drop capabilities of WCF RIA Services using hte DomainDataSource directly in the UI. In a later article I'll discuss the MVVM pattern, why this is not the best architecture for complex applications, and how you can still leverage the DomainDataSource but do so behind the scenes in a local service.

For this article though, open MainPage.xaml in the TaskManager project. The designer will open. Open the Data Sources window (Data menu > Show Data Sources). It will take a moment the first time you open it after the client code generation, so be patient. But you should see entities for the types returned by the domain service listed in the Data Sources window.



Drag and drop the Task entity type onto the MainPage.xaml in the designer. A DataGrid will be generated with appropriate columns for each of the properties on the entity type, and a DomainDataSource will also be declared and hooked up to the GetTasksQuery method on the TasksDomainContext class. You can drop down in the XAML and remove all the sizing and positioning properties from the DataGrid (Height, Width, HorizontalAlignment,VerticalAlignment, and Margin) so that the DataGrid fills the containing Grid.

Build and run the application. You should see something like the following.


TaskManager - Windows Internet Explorer
http://localhost:58041/TaskManagerTestPa
Bing
Favorites
TaskManager
Page
Safety
Tools
Local intranet | Protected Mode: Off
100%

Customer Id	Description	End Date	Project Id	Start Date	Task Id	Task Name	User Id
	Create a Silve	6/2/2010		6/1/2010	1	Create Projec	
	Create an Ent	6/3/2010		6/2/2010	2	Define Data M	
	Create a dom	6/4/2010		6/3/2010	3	Define domai	
	Use the Doma	6/5/2010		6/4/2010	4	Use data in th	