UNIT-4

SERVER SIDE PROGRAMMING

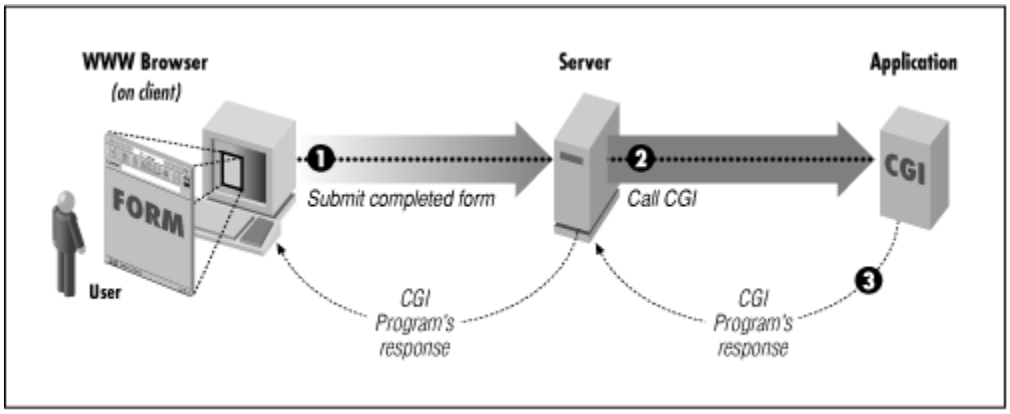
INTERNET PROGRAMMING PARADIGM

It can be classified into two categories:

- Client side programming Eg: Applets
- Server side programming

Server side programming

Common Gateway Interface (CGI) is one of important server-side programming techniques.



Architecture of CGI

Languages for CGI

- c/c++
- Perl
- Tcl
- Python
- Unix/Linux Shell

Environment variables

variable Name	Value
DOCUMENT_ROOT	The root directory of your server
HTTP_COOKIE	The visitor's cookie, if one is set
HTTP_HOST	The hostname of your server
HTTP_REFERER	The URL of the page that called your script
HTTP_USER_AGENT	The browser type of the visitor
HTTPS	"on" if the script is being called through a secure server
PATH	The system path your server is running under

QUERY_STRING	The query string (see GET, below)
REMOTE_ADDR	The IP address of the visitor
REMOTE_HOST	The hostname of the visitor (if your server has reverse-name-lookups on; otherwise this is the IP address again)
REMOTE_PORT	The port the visitor is connected to on the web server
REMOTE_USER	The visitor's username (for .htaccess-protected pages)
REQUEST_METHOD	GET or POST
REQUEST_URI	The interpreted pathname of the requested document or CGI (relative to the document root)
SCRIPT_FILENAME	The full pathname of the current CGI
SCRIPT_NAME	The interpreted pathname of the current CGI (relative to the document root)
SERVER_ADMIN	The email address for your server's webmaster
SERVER_NAME	Your server's fully qualified domain name (e.g. www.cgi101.com)
SERVER_PORT	The port number your server is listening on
SERVER_SOFTWARE	The server software you're using (such as Apache 1.3)

CGI building blocks

- Reading parameters passed to the script
- Processing the parameters
- Writing HTML response to standard output

Writing CGI program

```
#!/usr/bin/perl

print "Content-type:text/html\r\n\r\n";
print '<html>';
print '<head>';
print '<title>Hello Word - First CGI Program</title>';
print '</head>';
print '<body>';
print '<h2>Hello Word! This is my first CGI program</h2>';
print '</body>';
print '</html>';
```

This hello.cgi script is a simple PERL script which is writing its output on STDOUT file ie. screen. There is one important and extra feature available which is first line to be printed **Content-type:text/html\r\n\r\n**. This line is sent back to the browser and specify the content type to be displayed on the browser screen. Now you must have understood basic concept of CGI and

you can write many complicated CGI programs using PERL. This script can interact with any other external system also to exchange information such as RDBMS.

Alternatives and Enhancement to CGI

ASP

Active Server Pages, or ASP, was created by Microsoft for its web server, but it is now available for many servers. The ASP engine is integrated into the web server so it does not require an additional process. It allows programmers to mix code within HTML pages instead of writing separate programs.

PHP

PHP is a programming language that is similar to Perl, and its interpreter is embedded within the web server. PHP supports embedded code within HTML pages. PHP is supported by the Apache web server.

ColdFusion

Allaire's ColdFusion creates more of a distinction than PHP between code pages and HTML pages. HTML pages can include additional tags that call ColdFusion functions. A number of standard functions are available with ColdFusion, and developers can create their own controls as extensions. ColdFusion was originally written for Windows, but versions for various Unix platforms are now available as well. The ColdFusion interpreter is integrated into the web server.

Java servlets

Java servlets were created by Sun. Servlets are similar to CGI scripts in that they are code that creates documents. However, servlets, because they use Java, must be compiled as classes before they are run, and servlets are dynamically loaded as classes by the web server when they are run. The interface is quite different than CGI. JavaServer Pages, or JSP, is another technology that allows developers to embed Java in web pages, much like ASP.

FastCGI

FastCGI maintains one or more instances of perl that it runs continuously along with an interface that allows dynamic requests to be passed from the web server to these instances. It avoids the biggest drawback to CGI, which is creating a new process for each request, while still remaining largely compatible with CGI. FastCGI is available for a variety of web servers.

SERVLETS :

Servlets offer several advantages in comparison with CGI. First, performance is significantly better. Servlets execute within the address space of a web server. It is not necessary to create a separate process to handle each client request. Second, servlets are platform-independent because they are written in Java. Third, the Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. Finally, the full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases,or other software via the sockets and RMI mechanisms.

Server-side Java

Server-side Java (SSJ), sometimes called servlets or server-side applets, is a powerful hybrid of the Common Gateway Interface (CGI) and lower-level server API programming -- such as NSAPI from Netscape and ISAPI from Microsoft. SSJ is loaded dynamically into the server like NSAPI/ISAPI. This eliminates the start-up delays we've come to expect from CGI. It also allows the SSJ to maintain some of its state between executions, such as keeping an open connection to a database.

Advantages over Applets

Applets bring life to the traditional HTML pages. Servlets create dynamic web pages.

Difference between Applets and Servlets are

- Applet is client side program and Servlet is Server side.
- Applets can run under any web server their execution is dependent on Client as they require JRE Whereas Servlets do not require anything specific at client side, as they require java enabled web/application ServerApplet extends the Functionality of the Browser whereas Servlet Extends the Functionality of the Server.
- Applet runs at client side where as servlets run at server side. Unlike applets, however, servlets have no graphical user interface.
- Servlet doesn't have GUI , while applet have GUI. Applets are very heavy to handle as compared to servlet. Servlets are for server side and applet are for client view.
- An applet is a small program that is intended not to be run on its own, but rather to be embedded inside another application. Applets usually have some form of user interface or perform a particular piece of the overall user interface in a web page. This distinguishes them from a program written in a scripting programming language (such as JavaScript) that also runs in the context of a larger, client program, but which would not be considered an

applet. Whereas The Java Servlets API allows a software developer to add dynamic content to a web server using the Java platform. A servlet is an object that receives requests and generates a response based on the request. The API defines HTTP subclasses of the generic servlet requests and responses as well as an HTTP session object that tracks multiple requests and responses between the web server and a client. Servlets may be packaged as a Web application

Servlet alternatives

Java Server Pages

JavaServer Pages (JSP) technology is an extension of Java Servlet technology. It is the Java equivalent of MSP technology allowing Java code to be embedded in HTML pages. Since web*Mathematica* has a JSP implementation, it is very strongly integrated with JSPs.

PHP

PHP is a server-side, cross-platform, HTML-embedded scripting language. There is a PHP extension that allows interaction of PHP and servlets.

Servlet strengths

The advantages of Servlets are:

Portability

Portable across operating systems and across web servers

Power

Harness the full power of the core Java APIs: networking and URL access, multithreading, image manipulation, data compression, JDBC, object serialization, internationalization

Efficiency & Endurance

Memory resident, so invocation highly efficient—no process to spawn or interpreter to invoke

Safety

Support safe programming since inherit Java's strong type safety, exception-handling mechanism

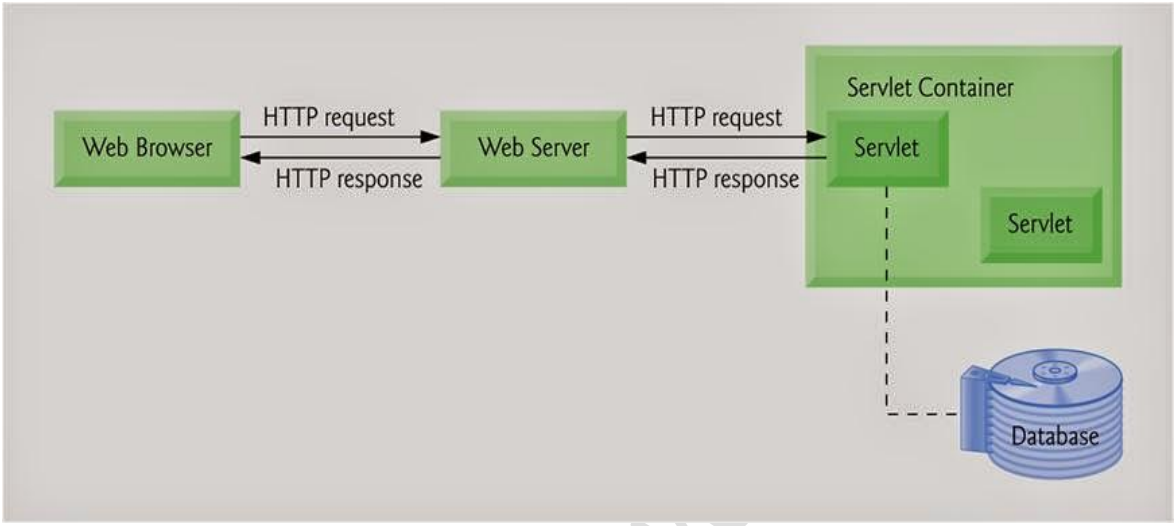
Elegance

Code is clean, object-oriented, modular, and simple (i.e.. Session tracking, cookie)

Integration

Tightly integrated with the server—translate file paths, perform logging, check authorization, and MIME type mapping

Servlet architecture



- Servlets read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
- Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.
- Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Servlet life cycle

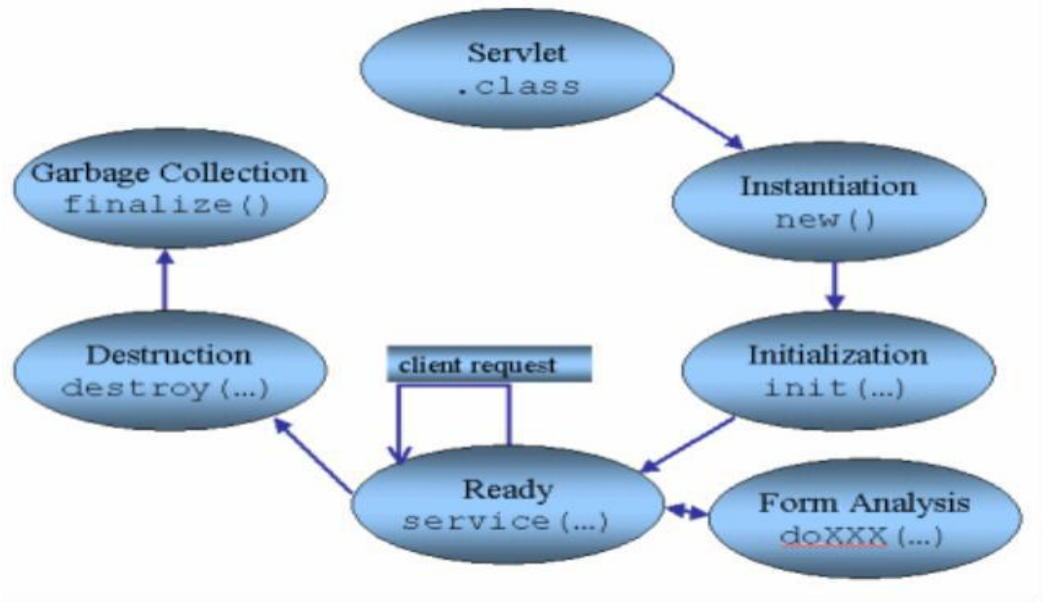
Three methods are central to the life cycle of a servlet. These are init(), service( ), and destroy( ). They are implemented by every servlet and are invoked at specific times by the server. Let us consider a typical user scenario to understand when these methods are called. First, assume that a user enters a Uniform Resource Locator (URL) to a web browser.

The browser then generates an HTTP request for this URL. This request is then sent to the appropriate server.

Second, this HTTP request is received by the web server. The server maps this request to a particular servlet. The servlet is dynamically retrieved and loaded into the address space of the server.

Third, the server invokes the init( ) method of the servlet. This method is invoked only when the servlet is first loaded into memory. It is possible to pass initialization parameters to the servlet so it may configure itself.

Fourth, the server invokes the service( ) method of the servlet. This method is called to process the HTTP request.



The servlet remains in the server's address space and is available to process any other HTTP requests received from clients. The service( ) method is called for each HTTP request. Finally, the server may decide to unload the servlet from its memory. The algorithms by which this determination is made are specific to each server. The server calls the destroy( ) method to relinquish any resources such as file handles that are allocated for the servlet. Important data may be saved to a persistent store. The memory allocated for the servlet and its objects can then be garbage collected.

Generic and HTTPServlet

GenericServlet class

It implements **Servlet**, **ServletConfig** and **Serializable**interfaces. It provides the implementaion of all the methods of these interfaces except the service method. It can handle any type of request so it is protocol-independent.

We may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

Methods of GenericServlet class

There are many methods in GenericServlet class. They are as follows:

1. **public void init(ServletConfig config)** is used to initialize the servlet.
2. **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
3. **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4. **public ServletConfig getServletConfig()** returns the object of ServletConfig.
5. **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
6. **public void init()** it is a convenient method for the servlet programmers, now there is no need to call super.init(config)
7. **public ServletContext getServletContext()** returns the object of ServletContext.
8. **public String getInitParameter(String name)** returns the parameter value for the given parameter name.
9. **public Enumeration getInitParameterNames()** returns all the parameters defined in the web.xml file.
10. **public String getServletName()** returns the name of the servlet object.
11. **public void log(String msg)** writes the given message in the servlet log file.
12. **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

The **HttpServlet** class extends the GenericServlet class and implements Serializable interface. It provides http specific methods such as doGet, doPost, doHead, doTrace etc.

Methods of HttpServlet class

There are many methods in HttpServlet class. They are as follows:

1. **public void service(ServletRequest req,ServletResponse res)** dispatches the request to the protected service method by converting the request and response object into http type.

2. **protected void service(HttpServletRequest req, HttpServletResponse res)** receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
3. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.
4. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.
5. **protected void doHead(HttpServletRequest req, HttpServletResponse res)** handles the HEAD request. It is invoked by the web container.
6. **protected void doOptions(HttpServletRequest req, HttpServletResponse res)** handles the OPTIONS request. It is invoked by the web container.
7. **protected void doPut(HttpServletRequest req, HttpServletResponse res)** handles the PUT request. It is invoked by the web container.
8. **protected void doTrace(HttpServletRequest req, HttpServletResponse res)** handles the TRACE request. It is invoked by the web container.
9. **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** handles the DELETE request. It is invoked by the web container.
10. **protected long getLastModified(HttpServletRequest req)** returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

First Servlet

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class DemoServlet extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
res.setContentType("text/html");//setting the content type
PrintWriter pw=res.getWriter();//get the stream to write the data

//writing html in the stream
pw.println("<html><body>");
pw.println("Welcome to servlet");
pw.println("</body></html>");
pw.close();//closing the stream
```

}}

Passing & Retrieving parameters in Servlets

The ServletRequest interface includes methods that allow you to read the names and values of parameters that are included in a client request. We will develop a servlet that illustrates their use. The example contains two files. A web page is defined in PostParameters.htm, and a servlet is defined in PostParametersServlet.java. The HTMLsource code for PostParameters.htm is shown in the following listing. It defines a table that contains two labels and two text fields. One of the labels is Employee and the other is Phone. There is also a submit button. Notice that the action parameter of the form tag specifies a URL. The URL identifies the servlet to process the HTTP POST request.

```
 <html>

<body> <center>

<form name="Form1" method="post"
action="http://localhost:8080/servlets-examples/
servlet/PostParametersServlet">

<table>

 <tr><td><B>Employee</td>

<td><input type=textbox name="e" size="25" value=""></td></tr>

<tr><td><B>Phone</td><td><input type=textbox name="p" size="25"
value=""></td> </tr>

 </table>

<input type=submit value="Submit"> </body> </html>
```

The source code for PostParametersServlet.java is shown in the following listing. The service( ) method is overridden to process client requests. The getParameterNames( ) method returns an enumeration of the parameter names. These are processed in a loop. The parameter value is obtained via the getParameter( ) method.

```
 import java.io.*;

 import java.util.*;

 import javax.servlet.*;

public class PostParametersServlet extends GenericServlet {

 public void service(ServletRequest request,ServletResponse response)
throws ServletException, IOException {
```

```
// Get print writer.
PrintWriter pw = response.getWriter();
// Get enumeration of parameter names.
Enumeration e = request.getParameterNames();
// Display parameter names and values.
while(e.hasMoreElements()) {
String pname = (String)e.nextElement();
pw.print(pname + " = ");
String pvalue = request.getParameter(pname);
pw.println(pvalue);
} pw.close();
} }
```

Server-side include

SSI (Server Side Includes) are directives that are placed in HTML pages, and evaluated on the server while the pages are being served. They let you add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program, or other dynamic technology.

For example, you might place a directive into an existing HTML page, such as:

```
<!--#echo var="DATE_LOCAL" -->
```

And, when the page is served, this fragment will be evaluated and replaced with its value:

Tuesday, 15-Jan-2013 19:28:54 EST

The decision of when to use SSI, and when to have your page entirely generated by some program, is usually a matter of how much of the page is static, and how much needs to be recalculated every time the page is served. SSI is a great way to add small pieces of information, such as the current time - shown above. But if a majority of your page is being generated at the time that it is served, you need to look for some other solution.

Cookies

Cookies are usually small text files, given ID tags that are stored on your computer's browser directory or program data subfolders. Cookies are

created when you use your browser to visit a website that uses cookies to keep track of your movements within the site, help you resume where you left off, remember your registered login, theme selection, preferences, and other customization functions. The website stores a corresponding file(with same ID tag)to the one they set in your browser and in this file they can track and keep information on your movements within the site and any information you may have voluntarily given while visiting the website, such as email address.

Cookies are often indispensable for websites that have huge databases, need logins, have customizable themes, other advanced features.

Cookies usually don't contain much information except for the url of the website that created the cookie, the duration of the cookie's abilities and effects, and a random number. Due to the little amount of information a cookie contains, it usually cannot be used to reveal your identity or personally identifying information. However, marketing is becoming increasingly sophisticated and cookies in some cases can be aggressively used to create a profile of your surfing habits.

There are two types of cookies: session cookies and persistent cookies. Session cookies are created temporarily in your browser's subfolder while you are visiting a website. Once you leave the site, the session cookie is deleted. On the other hand, persistent cookie files remain in your browser's subfolder and are activated again once you visit the website that created that particular cookie. A persistent cookie remains in the browser's subfolder for the duration period set within the cookie's file.

Filters

Servlet Filters are Java classes that can be used in Servlet Programming for the following purposes:

- To intercept requests from a client before they access a resource at back end.
- To manipulate responses from server before they are sent back to the client.

There are are various types of filters suggested by the specifications:

- Authentication Filters.
- Data compression Filters.
- Encryption Filters.
- Filters that trigger resource access events.
- Image Conversion Filters.

- Logging and Auditing Filters.
- MIME-TYPE Chain Filters.
- Tokenizing Filters .
- XSL/T Filters That Transform XML Content.

Filters are deployed in the deployment descriptor file **web.xml** and then map to either servlet names or URL patterns in your application's deployment descriptor.

When the web container starts up your web application, it creates an instance of each filter that you have declared in the deployment descriptor. The filters execute in the order that they are declared in the deployment descriptor.

Problems with Servlet

- In many Java servlet-based applications, processing the request and generating the response are both handled by a single servlet class.
- Thorough Java programming knowledge is needed to develop and maintain all aspects of the application, since the processing code and the HTML elements are lumped together. Z
- Changing the look and feel of the application, or adding support for a new type of client (such as a WML client), requires the servlet code to be updated and recompiled.
- It's hard to take advantage of web-page development tools when designing the application interface. If such tools are used to develop the web page layout, the generated HTML must then be manually embedded into the servlet code, a process which is time consuming, error prone, and extremely boring.

Server-side Security Issues

- Interception of Session State Information
- Forgery of Session State Information
- Session Timeout
- Buffer Overflow
- Data Validation, Page Sequencing
- Information Reporting
- Browser Residue
- User Authentication

• Logging of Sensitive Information

JSP engines

- Tomcat
- Java Web Server
- Web Logic
- Websphere

How JSP works

A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.

The following are the paths followed by a JSP

- Compilation
- Initialization
- Execution
- Cleanup

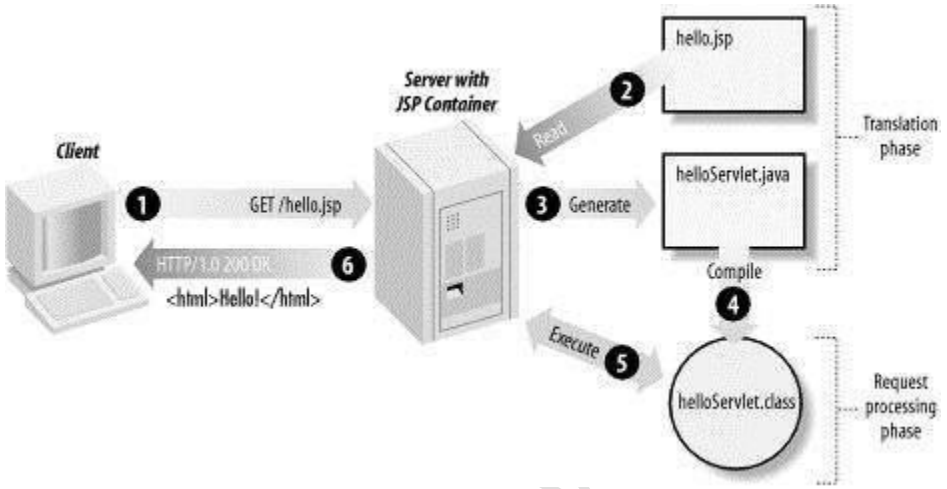
JSP and Servlet

The following steps explain how the web server creates the web page using JSP:

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of .html.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.

- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

All the above mentioned steps can be shown below in the following diagram:



Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster.

JSP components

Four different elements are used in constructing JSPs

- Directives
- Actions
- Declarations
- Scriptlets
- Expressions

Directives:

The jsp specification defines three directives

- Page: provder information about page, such as scripting language that is used, content type, or buffer size

- Include – used to include the content of external files
- Taglib – used to import custom actions defined in tag libraries

Actions:

- Standard actions should be supported by J2EE compliant web servers
- Custom actions can be created using tag libraries
- The different actions are
  - Include action
  - Forward action
  - Param action
  - useBean action
  - getProperty action
  - setProperty action
  - plugIn action

Declarations :

Declarations are used to define methods & instance variables

- Do not produce any output that is sent to client
- Embedded in <%! and %> delimiters

Example:

```
<%!
 Public void jspDestroy() {
        System.out.println("JSP Destroyed");
}
Public void jspInit() {
        System.out.println("JSP Loaded");
}
int myVar = 123;
%>
```

Scriptlets:

Used to embed java code in JSP pages.

- Contents of JSP go into _JSPpageservice() method
- Code should comply with syntactical and semantic constuct of java
- Embedded in <% and %> delimiters

Example:

```
<%
int x = 5;
int y = 7;
int  z = x + y;
%>
```

Expressions:

Used to write dynamic content back to the browser.

- If the output of expression is Java primitive the value is printed back to the browser
- If the output is an object then the result of calling toString on the object is output to the browser
- Embedded in <%= and %> delimiters

Example:

- <%="Fred"+ " " + "Flintstone %>

prints "Fred Flintstone" to the browser

- <%=Math.sqrt(100)%>

prints 10 to the browser

SHORT ANSWER QUESTIONS:

1. Define the GET() and POST() method

GET()

The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

POST()

The POST method is used to request that the destination server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the RequestLine

2. What is a Servlet?

Servlets are modules of Java code that run in a server application (hence the name "Servlets", similar to "Applets" on the client side) to answer client requests. Servlets are not tied to a specific client-server protocol but they are most commonly used with HTTP and the word "Servlet" is often used in the meaning of "HTTP Servlet

3. Compare Servlet with CGI Servlets have several advantages over CGI:

A Servlet does not run in a separate process. This removes the overhead of creating a new process for each request. A Servlet stays in memory between requests. A CGI program (and probably also an extensive runtime system or interpreter) needs to be loaded and started for each CGI request. There is only a single instance which answers all requests concurrently. This saves memory and allows a Servlet to easily manage persistent data

4. How the Sessions can be maintained in Session tracking?

1. By using Cookies. A Cookie is a string (in this case that string is the session ID) which is sent to a client to start a session. If the client wants to continue the session it sends back the Cookie with subsequent requests. This is the most common way to implement session tracking.

2. By rewriting URLs. All links and redirections which are created by a Servlet have to be encoded to include the session ID. This is a less elegant solution (both, for Servlet implementors and users) because the session cannot be maintained by requesting a wellknown URL oder selecting a URL which was created in a different (or no) session.

5.Explain the life cycle methods of a Servlet.

The javax.servlet.Servlet interface defines the three methods known as life-cycle method. public void init(ServletConfig config) throws ServletException public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException public void destroy() First the servlet is constructed, then initialized wih the init() method. Any request from client are handled initially by the service() method before delegating to the doXxx() methods in the case of HttpServlet. The servlet is removed from service, destroyed with the destroy() methid, then garbaged collected and finalized

6. What is the difference between the getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface and javax.servlet.ServletContext interface?

The getRequestDispatcher(String path) method of javax.servlet.ServletRequest interface accepts parameter the path to the resource to be included or forwarded to, which can be relative to the request of the calling servlet. If the path begins with a "/" it is interpreted as relative to the current context root. The getRequestDispatcher(String path) method of javax.servlet.ServletContext interface cannot accepts relative paths. All path must sart with a "/" and are interpreted as relative to curent context root.

7. Explain the directory structure of a web application.

The directory structure of a web application consists of two parts. A private directory called WEB-INF. A public resource directory which contains public resource folder. WEB-INF folder consists of

1. web.xml
2. classes directory
3. lib directory

8. What are the common mechanisms used for session tracking?

- Cookies
- SSLsessions
- URL- rewriting

9. Explain ServletContext?

ServletContext interface is a window for a servlet to view it's environment. A servlet can use this interface to get information such as initialization parameters for the web applicationor servlet container's version. Every web application has one and only one ServletContext and is accessible to all active resource of that application.

10.What is the difference between ServletContext and ServletConfig?

ServletContext: Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.The ServletContext object is contained within the ServletConfig object, which the Web server provides the servlet when the servlet is initialized

ServletConfig: The object created after a servlet is instantiated and its default constructor is read. It is created to pass.

BITS

1. The doGet() method in the example extracts values of the parameter's type and number by using _____ [A]
a) request.getParameter() b) request.setParameter()

c) responce.getParameter() d) responce.getAttribute()

2. A JSP is transformed into a(n): [B]

a) Java applet b) Java servlet c) Either 1 or 2 above d) Neither 1 nor 2 above

3. The method getWriter returns an object of type PrintWriter. This class has println methods to generate output. Which of these classes define thegetWriter method? **[C]**
(A) HttpServletRequest (B) ServletConfig

(C) HttpServletResponse (D) ServletContext

4.In which file do we define a servlet mapping? [C]

(A) servlet.mappings (B) servlet.xml (C) web.xml (D) Simple.java

5. For a given ServletResponse response, which retrieve an object for writing text data? [B]

(A) response.getOutputWriter() (B) response.getWriter()

(C) response.getWriter().getOutputStream()
(D)response.getWriter(Writer.OUTPUT_TEXT)

6. Name the method defined in the HttpServletResponse class that may be used to set the content type. [A]

(A) setContent (B) setType (C) setContentType (D) setResponseContentType

7. If a jsp is to generate a pdf page, what attribute of page directive it should use? [A]

A) contentType B)generatePdf C) typePDF D) contentPDF

8.Which of the following attributes are mandatory in <jsp:getProperty /> tag? [A]

A) name, property B) type, id C) name, type D) id, property

9. Which of the following attributes are used in <jsp:include /> tag?

[B]

A) id, type B) page, flush C) type, class D) type,page

10. out is instance of which class? [A]

A) javax.servlet.jsp.JspWriter B - javax.servlet.jsp.PringWriter

C) javax.servlet.Writer D) javax.servlet.jsp.Writer

Unit-5

SEVER SIDE PROGRAMMING

Beans:

In a web application, server may be responding to several clients at a time so session tracking is a way by which a server can identify the client. As we know HTTP protocol is stateless which means client needs to open a separate connection every time it interacts with server and server treats each request as a new request.

Now to identify the client , server needs to maintain the state and to do so , there are several session tracking techniques.

Session Tracking Techniques

There are four techniques which can be used to identify a user session.

a) Cookies

b) Hidden Fields

c) URL Rewriting

d) Session Object

Cookie

Cookie is a key value pair of information, sent by the server to the browser and then browser sends back this identifier to the server with every request there on.

There are two types of cookies:

· **Session cookies** - are temporary cookies and are deleted as soon as user closes the browser. The next time user visits the same website, server will treat it as a new client as cookies are already deleted.

· **Persistent cookies** - remains on hard drive until we delete them or they expire.

Hidden Field

Hidden fields are similar to other input fields with the only difference is that these fields are not displayed on the page but its value is sent as other input fields. For example

<input type="hidden" name="sessionId" value="unique value"/>

is a hidden form field which will not displayed to the user but its value will be send to the server and can be retrieved using request.getParameter("sessionId") .

URL Rewriting

URL Rewriting is the approach in which a session (unique) identifier gets appended with each request URL so server can identify the user session. For example if we apply URL rewriting on **http://localhost:8080/jsp-tutorial/home.jsp** , it will become something like

?jSessionId=XYZ where jSessionId=XYZ is the attached session identifier and value XYZ will be used by server to identify the user session.

There are several advantages of URL rewriting over above discussed approaches like it is browser independent and even if user's browser does not support cookie or in case user has disabled cookies, this approach will work.

Another advantage is , we need not to submit extra hidden parameter.

Session Object

Session object is representation of a user session. User Session starts when a user opens a browser and sends the first request to server. Session object is available in all the request (in entire user session) so attributes stored in Http session in will be available in any servlet or in a jsp.

When session is created, server generates a unique ID and attach that ID with the session. Server sends back this Id to the client and there on , browser sends back this ID with every request of that user to server with which server identifies the user.

Users passing control and data between pages

Separating presentation pages from request more than one page used to◊processing/business logic

- process client request need to be able to pass control from one page to another

--- e.g. in the example, infovalidate.jsp need to be able to forward to either userinput.jsp or confirmed.jsp depending on validation result Can use the standard action tag

For example: A validation page (infovalidate.jsp) forward control to a page, userinput.jsp, in order to display an error message. Need to include error message in the the forwarding instruction.

<jsp:forward page="home.jsp"/>
<jsp:param name="msg" value=target page/>

The target page in this example is assumed to be in the same directory on the web server as the current JSP page The target page in this example is assumed to be in the /somedir/ directory as a subset of the main application directory (../webapps/myapp/) Note: the tag is similar to the tag, but also allows redirection to a different URL.

Sharing session and Application data

HTTP is a stateless, request-response protocol. This means that the browser sends a request for a web resource, and the web server processes the request and returns a response. The server then forgets this transaction ever happened.

• So when the same browser sends a new request, the web server has no idea that this request is related to the previous one.

•This is fine if you're dealing with static files, but it's a problem in an interactive web application. • In a travel agency application, for instance, it's important to remember the dates and destination entered to book the flight so the customer doesn't have to enter the same information again when it's time to make hotel and rental car reservations.

•The way to solve this problem is to let the server send a piece of information to the browser that the browser then includes in all subsequent requests.

•This piece of information, called a session ID, is used by the server to recognize a set of requests from the same browser as related: in other words, as part of the same session.

•A session starts when the browser makes the first request for a JSP page in a particular application. The session can be ended explicitly by the application, or the JSP container can end it after a period of user inactivity (the default value is typically 30 minutes after the last request).

There are a number of different techniques available to web applications to enable session tracking, including cookies (next course). In JSP, can be done by simply using the 'scope' attribute of whatever¬ needs to be tracked. Set the 'scope' to session, and the relevant attribute will be available¬ throughout the entire session of the client.

Database connectivity

Database programming has traditionally been a technological Tower Java is supposed to bring us the ability to "write once, compile once, and run anywhere," so it should bring it to us with database programming as well. Java's JDBC API gives us a shared language through which our applications can talk to database engines. Following in the tradition of its other multi-platform APIs such as the AWT, JDBC provides us with a set of interfaces that create a common point at which database applications and database engines can meet..

JDBC

JavaSoft developed a single API for database access--JDBC. As part of this process, they kept three main goals in mind:

JDBC should be an SQL-level API.
JDBC should capitalize on the experience of existing database APIs.
JDBC should be simple.

An SQL-level API means that JDBC allows us to construct SQL statements and embed them inside Java API calls. In short, you are basically using SQL. But JDBC lets you smoothly translate between the world of the database and the world of the Java application.

JDBC Drivers

Driver categories :

type 1

These drivers use a bridging technology to access a database. The JDBC-ODBC bridge that comes with the JDK 1.1 is a good example of this kind of driver. It provides a gateway to the ODBC API. Implementations of that API in turn do the actual database access. Bridge solutions generally require software to be installed on client systems, meaning that they are not good solutions for applications that do not allow you to install software on the client.

type 2

The type 2 drivers are native API drivers. This means that the driver contains Java code that calls native C or C++ methods provided by the individual database vendors that perform the database access. Again, this solution requires software on the client system.

type 3

Type 3 drivers provide a client with a generic network API that is then translated into database specific access at the server level. In other words, the JDBC driver on the client uses sockets to call a middleware application on the server that translates the client requests into an API specific to the desired driver. As it turns out, this kind of driver is extremely flexible since it requires no code installed on the client and a single driver can actually provide access to multiple databases.

type 4

Using network protocols built into the database engine, type 4 drivers talk directly to the database using Java sockets. This is the most direct pure Java solution. In nearly every case, this type of driver will come only from the database vendor.

Basic steps

- Loading a driver
- Making a connection
- Executing an SQL statement

Loading a driver

```
String driver = "org.gjt.mm.mysql.Driver";
 Class.forName(driver).newInstance();
```

Connection to database:

```
String driver =
"org.gjt.mm.mysql.Driver";Class.forName(driver).newInstance();
String
url="jdbc:mysql://localhost/books?user=<userName>&password=<passwor
d>";
con=DriverManager.getConnection(url);
```

Executing Query or Accessing data from database:

```
stmt=con.createStatement(); //create a Statement object
rst=stmt.executeQuery("select * from books_details");
```

stmt is the Statement type variable name and **rst** is the RecordSet type variable. A query is always executed on a Statement object.A Statement object is created by calling createStatement() method on connection object **con.** The two most important methods of this Statement interface are executeQuery() and executeUpdate(). The executeQuery() method executes an SQL statement that returns a single ResultSet object. The executeUpdate() method executes an insert, update, and delete SQL statement. The method returns the number of records affected by the SQL statement execution.After creating a Statement ,a method executeQuery() or executeUpdate() is called on Statement object **stmt** and a SQL query string is passed in method executeQuery() or executeUpdate(). This will return a ResultSet **rst** related to the query string.

Reading values from a ResultSet

```
while(rst.next()){
  %>
<tr><td><%=no%></td><td><%=rst.getString("book_name")%></td>
<td><%=rst.getString("author")%></
td></tr> <%
}
```

The ResultSet represents a table-like database result set. A ResultSet object maintains a cursor pointing to its current row of data. Initially, the cursor is positioned before the first row. Therefore, to access the first row in the ResultSet, you use the next() method. This method moves the cursor to the next record and returns true if the next row is valid, and false if there are no more records in the ResultSet object. Other important methods are getXXX() methods, where XXX is the data type returned by the method at the specified index, including String, long, and int. The indexing used is 1-based. For example, to obtain the second column of type String, you use the following code:

```
resultSet.getString(2);
```

Introduction to JavaBeans

A Java Beans is software component that has been designed to be reusable in a variety of different environments. There is no restriction on the capability of a Bean. It mayperform simple function, such as checking the spelling of a document, or complex function, such as forecasting the performance of a stock portfolio. A bean may be visible to an end user. One example of this is a button on a graphical user interface. A bean may be designed to work autonomously on a user's workstation or to work in cooperation with a set of other distributed components.

Bean builder

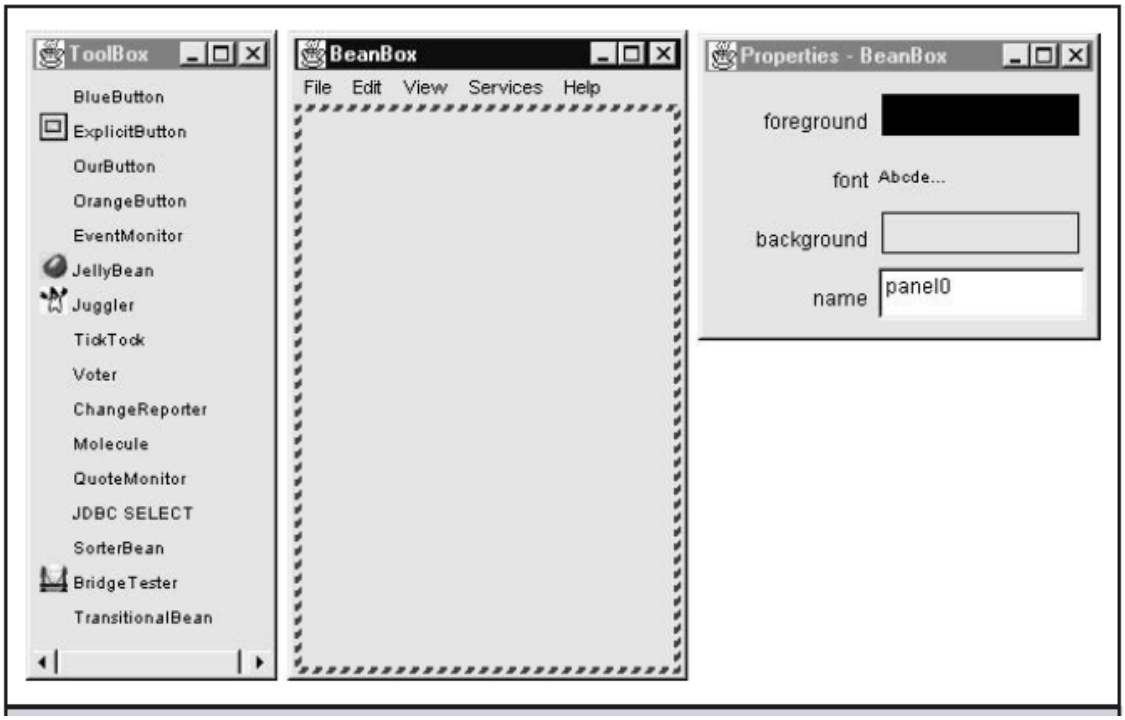
The Bean Developer Kit (BDK), available from the JavaSoft site, is a simple example of a tool that enables you to create, configure, and connect a set of Beans. There is also a set of sample Beans with their source code.

Thissection provides step-by-step instructions for installing and using this tool.

Starting the BDK

To start the BDK, follow these steps:

1. Change to the directory **c:\bdk\beanbox**.
2. Execute the batch file called **run.bat**. This causes the BDK to display the three windows shown in Figure . ToolBox lists all of the different Beans that have been included with the BDK. BeanBox provides an area to lay out and connect the Beans selected from the ToolBox. Properties provides the ability to configure a selected Bean. You may also see a window called Method Tracer,.



Advantages of Java Beans

- A bean obtains all the benefits of Java's "write once, run-anywhere" paradigm.
- The properties, events and methods of a bean that are exposed to an application builder tool can be controlled.
- A bean may be designed to operate correctly in different locales, which makes it useful in global markets.
- Auxiliary software can be provided to help a person configure a bean.
- The configuration settings of a bean can be saved in persistent storage and restored at a later time.
- A bean may register to receive events from other objects and can generate events that are sent to other objects.

BDK introspection

Introspection is the process of analyzing a bean to determine its capabilities. This is a very important feature of Java Bean API, because it allows an application builder tool to present information about a component to a software designer. Without introspection, the java beans technology could not operate. One way exposed the properties, events and methods of bean to application builder tool is using simple naming conventions.

Properties

Design pattern for properties

Property is a subset of a bean's state. The values that are assigned to the properties determine the behavior and appearance of that component.

Simple properties:

A simple property has a single value. It can be identified by the following design patterns, where N is the name of the property and T is its type.

Public T getN( );
Public void setN( );

Boolean properties:

A Boolean property has a value of true or false. It can be identified by the following design patterns, where N is name of the property.

Public Boolean isN ( );
Public Boolean getN( );
Public void setN(Boolean value);

indexed properties

An indexed property consists of multiple values. It can be identified by the following design patterns, where N is the name of the property and T is its type.

Public T getN(int index);
Public void setN(int index, T value);
Public T[ ] getN( );
Public void setN(T values[ ]);

BeanInfo interface

This interface defines several methods, including these:

PropertyDescription[ ] getPropertyDescriptors( )
EventSetDescriptor[ ] getEventSetDescriptors( )
MethodDescriptor[ ] getMethodDescriptors( )

The above methods will return array of objects that provide information about the properties, events, and methods of bean. SimpleBeanInfo is a class that provides default implementations of the BeanInfo interface, including the three methods . this class and override on or more of them.

Constrained Properties

A bean that has a constrained property generates an event when an attempt is made to change its value. The event is of type PropertyChangeEvent. It is sent to objects that previously registered an

interest in receiving such notifications. This capability allows a Bean to operate differently according to its run-time environment.

Persistence

Persistence is the ability to save a Bean to nonvolatile storage and retrieve it at a later time. The information that is particularly important are the configuration settings.

Customizers

A bean developer can provide a customizer that helps another developer configure this software. A customizer can
provide a step-by-step guide through the process that must be followed to use the component in a specific context.

Java Beans API

Interface Description

AppletInitializer Methods present in this interface are used to initialize Beans that are also applets BeanInfo This interface allows a designer to specify information about the properties, events and methods of a Bean. Customizer This interface allows a designer to provide a graphical user interface through which a Bean may be configured. Design Mode Methods in this interface determine if a Bean is executing in design mode.
PropertyChangeListener A method in this interface is invoked when a bound property is changed.
Visibility Methods in this interface allow a bean to execute in environments where graphical user interface is not available.

Class Description

BeanDescriptor This class provides information about a Bean. Beans This class is used to obtain information about a Bean IntrospectionException An exception of this type is generated if a problem occurs when analyzing a bean. PropertyChangeEvent This event is generated when bound or constrained properties are changed. PropertyDescriptor Instances of this class describe a property of a Bean.

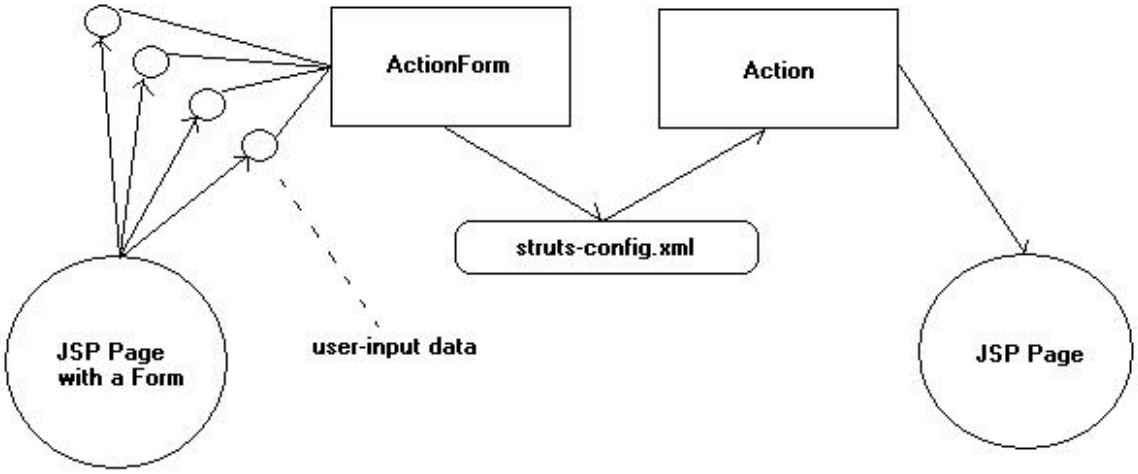
EJB

Enterprise JavaBeans (EJB) is a comprehensive technology that provides the infrastructure for building enterpriselevel server-side distributed Java components. The EJB technology provides a distributed component architectuthat integrates several enterprise-level requirements such as distribution, transactions, security, messaging, persistence, and connectivity to mainframes and Enterprise Resource Planning (ERP) systems. When compared with other distributed component technologies such as Java RMI and CORBA, the EJB architecture hides most the underlying system-level semantics that are typical of distributed component applications, such as instance management, object pooling, multiple threading, and connection pooling. Secondly, unlike other component models, EJB technology provides us with different types of components for business logic, persistence, and enterprise messages.

Introduction to Struts Framework

Struts is a framework that promotes the use of the Model-View-Controller architecture for designing large scale applications. The framework

includes a set of custom tag libraries and their associated Java classes, along with various utility classes. The most powerful aspect of the Struts framework is its support for creating and processing web-based forms.
The following diagram roughly depicts the use of Struts for using forms.



SHORT ANSWER QUESTIONS:

1. What is JDBC?

JDBC may stand for Java Database Connectivity. It is also a trade mark. JDBC is a layer of abstraction that allows users to choose between databases. It allows you to change to a different database engine and to write to a single API. JDBC allows you to write database applications in Java without having to concern yourself with the underlying details of a particular database.

2. **What are the common tasks of JDBC?**

- Create an instance of a JDBC driver or load JDBC drivers through jdbc.drivers
- Register a driver
- Specify a database
- Open a database connection
- Submit a query
- Receive results

3. There are three basic types of SQL statements, what are they?

i)Statement

ii)CallableStatement

iii)PreparedStatement

4. How can you load the drivers?

Loading the driver or drivers you want to use is very simple and involves just one line of code. If, for example, you want to use the JDBC-ODBC Bridge driver, the following code will load it:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

5. How to make a query?

Create a Statement object and call the Statement.executeQuery method to select data from the database. The results of the query are returned in a ResultSet object.

```
Statement stmt = con.createStatement();
```

ResultSet results = stmt.executeQuery("SELECT data FROM aDatabase ");

6. How to set a scroll type?

Both Statements and PreparedStatements have an additional constructor that accepts a scroll type and an update type parameter. The scroll type value can be one of the following values:

ResultSet.TYPE_FORWARD_ONLY

Default behavior in JDBC 1.0, application can only call next() on the result set.

ResultSet.SCROLL_SENSITIVE

ResultSet is fully navigable and updates are reflected in the result set as they occur.

ResultSet.SCROLL_INSENSITIVE

7. How to set update type parameter?

In the constructors of Statements and PreparedStatements, you may use

- ResultSet.CONCUR_READ_ONLY

The result set is read only.

- ResultSet.CONCUR_UPDATABLE

The result set can be updated.

8. What Class.forName will do while loading driver?

It is used to create an instance of a driver and register it with the DriverManager. When you have loaded a driver, it is available for making a connection with DBMS.

9. What do you mean by fastest type of JDBC driver?

JDBC driver performance or fastness depends on a number of issues Quality of the driver code, size of the driver code, database server and its load, Network topology, Number of times your request is translated to a different API.

10. What are JDBC driver types?

There are four types of JDBC drivers

- JDBC-ODBC Bridge plus ODBC driver – also called Type 1 calls native code of the locally available ODBC driver.
- Native-API, partly Java driver – also called Type 2 calls database vendor native library on a client side. This code then talks to database over network.
- JDBC-Net, pure Java driver – also called Type 3 the pure-java driver that talks with the server-side middleware that then talks to database.
- Native-protocol, pure Java driver – also called Type 4 the pure-java driver that uses database native protocol.

BITS

1. A Java program cannot directly communicate with an ODBC driver because……. [A]

A) ODBC written in C language

B) ODBC written in C# language

C) ODBC written in C++ language

D) ODBC written in Basic language

2. The JDBC-ODBC Bridge driver translates the JDBC API to the ODBC API and used with……. [B]

A) JDBC drivers

B) ODBC drivers

C) Both A and B

D) None of the above

3. The…… package contains classes that help in connecting to a database, sending SQL statements to the database, and processing the query results. [D]

A) connection.sql

B) db.sql

C) pkg.sql

D) java.sql

4. The..... method executes a simple query and returns a single Result Set object. [B]

A) executeUpdate()

B) executeQuery()

C) execute()

D) noexecute()

5. The ..... method executes an SQL statement that may return multiple results. [C]

A) executeUpdate()

B) executeQuery()

C) execute()

D) noexecute()

6. The ..... object allows you to execute parameterized queries. [C]

A) ResultSet

B) Parametrized

C) PreparedStatement

D) Condition

7. The ..... object provides you with methods to access data from the table. [A]

A) ResultSet

B) Parametrized

C) TableStatement

D) Condition

8. The parameters of the PreparedStatement object are …. when the user clicks on the Query button. [A]

A) initialized

B) started

C) paused

D) stopped

9. The …… method sets the query parameters of the PreparedStatement Object. [C]

A) putString()

B) insertString()

C) setString()

D) setToString()

10. Connection object can be initialized using the…. method of the Driver Manager class. [D]

A) putConnection()

B) setConnection()

C) Connection()

D) getConnetion()