Programming in Java Dr. M. T. Wanjari

**Unit – IV**

**Servlets**

Servlets are small programs that execute on the server side of a Web connection. Just as applets dynamically extend the functionality of a **Web browser**, servlets dynamically extend the functionality of a **Web server**. We focus on the core concepts, interfaces, and classes, and several applications.

**Background**

In order to understand the advantages of servlets, must have a basic understanding of how Web browsers and servers cooperate to provide content to a user. Consider a request for a static Web page. A user enters a Uniform Resource Locator (URL) into a browser. The browser generates an HTTP request to the appropriate Web server. The Web server maps this request to a specific file. That file is returned in an HTTP response to the browser. The HTTP header in the response indicates the type of the content.

The **Multipurpose Internet Mail Extensions (MIME)** are used for this purpose. For example, ordinary ASCII text has a MIME type of text/plain. The Hypertext Markup Language (HTML) source code of a Web page has a MIME type of text/html. Now consider dynamic content. Assume that an online store uses a database to store information about its business. This would include items for sale, prices, availability, orders, and so forth. It wishes to make this information accessible to customers via Web pages. The contents of those Web pages must be dynamically generated in order to reflect the latest information in the database.

In the early days of the Web, a server could dynamically construct a page by creating a separate process to handle each client request. The process would open connections to one or more databases in order to obtain the necessary information. It communicated with the Web server via an interface known as the **Common Gateway Interface (CGI)**.

CGI allowed the separate process to read data from the HTTP request and write data to the HTTP response. A variety of different languages were used to build CGI programs. These included C, C++, and Perl. However, CGI suffered serious performance problems. It was expensive in terms of processor and memory resources to create a separate process for each client request. It was also expensive to open and close database connections for each client request.

M.Sc. Semester – I (Computer Science) 1

Programming in Java Dr. M. T. Wanjari

In addition, the CGI programs were not platform-independent. Therefore, other techniques were introduced. Among these are 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. A number of Web servers from different vendors offer the Servlet API. Programs developed for this API can be moved to any of these environments without recompilation.

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 that have seen already.

**The Life Cycle of a Servlet**

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. It is possible for the servlet to read data that has been provided in the HTTP request. It may also formulate an HTTP response for the client. 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.

M.Sc. Semester – I (Computer Science) 2

Programming in Java Dr. M. T. Wanjari

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.

**Using Tomcat For Servlet Development**

To create servlets, need to download a servlet development environment. The one currently recommended by Sun is Tomcat 4.0, which supports the latest servlet specification, which is 2.3. (The complete servlet specification is available for download through java.sun.com.) Tomcat replaces the old JSDK (Java Servlet Development Kit) that was previously provided by Sun. Tomcat is an open-source product maintained by the Jakarta Project of the Apache Software Foundation. It contains the class libraries, documentation, and run-time support that need to create and test servlets. Can download Tomcat through the Sun Microsystems Web site at java.sun.com. The current version is 4.0. Follow the instructions to install this toolkit on machine. The examples assume a Windows environment. The default location for Tomcat 4.0 is

C:\Program Files\Apache Tomcat 4.0\

If load Tomcat in a different location, will need to make appropriate changes to the examples. May need to set the environmental variable JAVA\_HOME to the top-level directory in which the Java Software Development Kit is installed. For Java 2, version 1.4, the default directory is C:\j2sdk1.4.0, but will need to confirm this for environment. To start Tomcat, select Start Tomcat in the Start | Programs menu, or run startup.bat from the directory. C:\Program Files\Apache Tomcat 4.0\bin\

When done testing servlets, can stop Tomcat by selecting Stop Tomcat in the Start | Programs menu, or run shutdown.bat. The directory contains servlet.jar.

C:\Program Files\Apache Tomcat 4.0\common\lib\

This JAR file contains the classes and interfaces that are needed to build servlets. To make this file accessible, update your CLASSPATH environment variable so that it includes C:\Program Files\Apache Tomcat 4.0\common\lib\servlet.jar.

Alternatively, can specify this class file when compile the servlets. For example, the following command compiles the first servlet example:

M.Sc. Semester – I (Computer Science) 3

Programming in Java Dr. M. T. Wanjari

javac HelloServlet.java -classpath "C:\Program Files\Apache Tomcat 4.0\common\lib\servlet.jar" Once have compiled a servlet, must copy the class file into the directory that Tomcat uses for example servlet class files, must put the servlet files into the following directory: C:\Program Files\Apache Tomcat 4.0\webapps\examples\WEB-INF\classes **A Simple Servlet**

To become familiar with the key servlet concepts, we will begin by building and testing a simple servlet. The basic steps are the following:

1. Create and compile the servlet source code.

2. Start Tomcat.

3. Start a Web browser and request the servlet.

Let us examine each of these steps in detail.

**Create and Compile the Servlet Source Code**

To begin, create a file named HelloServlet.java that contains the following program: import java.io.\*;

import javax.servlet.\*;

public class HelloServlet extends GenericServlet {

public void service(ServletRequest request,

ServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter pw = response.getWriter();

pw.println("<B>Hello!");

pw.close();

}

}

Above program. First, note that it imports the **javax.servlet package**. This package contains the classes and interfaces required to build servlets. Next, the program defines HelloServlet as a subclass of GenericServlet. The GenericServlet class provides functionality that makes it easy to handle requests and responses. Inside HelloServet, the service( ) method (which is inherited from GenericServlet) is overridden. This method handles requests from a

M.Sc. Semester – I (Computer Science) 4

Programming in Java Dr. M. T. Wanjari

client. Notice that the first argument is a ServletRequest object. This enables the servlet to read data that is provided via the client request.

The second argument is a ServletResponse object. This enables the servlet to formulate a response for the client. The call to setContentType( ) establishes the MIME type of the HTTP response. In this program, the MIME type is text/html. This indicates that the browser should interpret the content as HTML source code. Next, the getWriter( ) method obtains a PrintWriter. Anything written to this stream is sent to the client as part of the HTTP response. Then println( ) is used to write some simple HTML source code as the HTTP response. Compile this source code and place the HelloServlet.class file in the Tomcat class files directory as described. **Start Tomcat**

As explained, to start Tomcat, select Start Tomcat in the Start | Programs menu, or run startup.bat from the directory.

C:\Program Files\Apache Tomcat 4.0\bin\

**Start a Web Browser and Request the Servlet**

Start a Web browser and enter the URL shown here:

http://localhost:8080/examples/servlet/HelloServlet

Alternatively, you may enter the URL shown here:

http://127.0.0.1:8080/examples/servlet/HelloServlet

This can be done because 127.0.0.1 is defined as the IP address of the local machine. Observe the output of the servlet in the browser display area. It will contain the string Hello! in bold type.

**The Servlet API**

Two packages contain the classes and interfaces that are required to build servlets. These are **javax.servlet** and **javax.servlet.http**. They constitute the Servlet API. Keep in mind that these packages are not part of the Java core packages. Instead, they are standard extensions. Therefore, they are not included in the Java Software Development Kit. Must download Tomcat to obtain their functionality. The Servlet API has been in a process of ongoing development and enhancement. The current servlet specification is version is 2.3 and that is the one used. However, because changes happen fast in the world of Java, want to check for any additions or alterations. The Servlet API is supported by most Web servers, such as those from Sun,

Microsoft, and others.

M.Sc. Semester – I (Computer Science) 5

Programming in Java Dr. M. T. Wanjari

**The javax.servlet Package**

The javax.servlet package contains a number of interfaces and classes that establish the framework in which servlets operate. The following table summarizes the core interfaces that are provided in this package. The most significant of these is Servlet. All servlets must implement this interface or extend a class that implements the interface. The ServletRequest and ServletResponse interfaces are also very important.

|  |  |
| --- | --- |
| **Interface** | **Description** |
| Servlet | Declares life cycle methods for a servlet. |
| ServletConfig | Allows servlets to get initialization parameters. |
| ServletContext | Enables servlets to log events and access information  about their environment. |
| ServletRequest | Used to read data from a client request. |
| ServletResponse | Used to write data to a client response. |
| SingleThreadModel | Indicates that the servlet is thread safe. |

The following table summarizes the core classes that are provided in the javax.servlet package.

|  |  |
| --- | --- |
| **Class** | **Description** |
| GenericServlet | Implements the Servlet and ServletConfig interfaces. |
| ServletInputStream | Provides an input stream for reading requests from a client. |
| ServletOutputStream | Provides an output stream for writing responses to a client. |
| ServletException | Indicates a servlet error occurred. |
| UnavailableException | Indicates a servlet is unavailable. |

**The Servlet Interface**

All servlets must implement the Servlet interface. It declares the init( ), service( ), and destroy( ) methods that are called by the server during the life cycle of a servlet. A method is also provided that allows a servlet to obtain any initialization parameters. The methods defined by Servlet are shown in Table 1.

|  |  |
| --- | --- |
| **Method** | **Description** |
| void destroy( ) | Called when the servlet is unloaded |
| ServletConfig getServletConfig( ) | Returns a ServletConfig object that contains any initialization parameters. |
| String getServletInfo( ) | Returns a string describing the servlet. |
| void init(ServletConfig sc)  throws ServletException | Called when the servlet is initialized. Initialization parameters for the servlet can be obtained from sc. An UnavailableException should be thrown if the servlet cannot be initialized. |

M.Sc. Semester – I (Computer Science) 6

Programming in Java Dr. M. T. Wanjari

|  |  |
| --- | --- |
| void service(ServletRequest req, ServletResponse res)  throws ServletException,  IOException | Called to process a request from a client. The request from the client can be read from req. The response to the client can be written to res. An exception is generated if a servlet or IO problem occurs. |
| **Table 1:- The Methods defined by Servlet** | |

The init( ), service( ), and destroy( ) methods are the life cycle methods of the servlet. These are invoked by the server. The getServletConfig( ) method is called by the servlet to obtain initialization parameters. A servlet developer overrides the getServletInfo( ) method to provide a string with useful information (for example, author, version, date, copyright). This method is also invoked by the server.

**The ServletConfig Interface**

The ServletConfig interface is implemented by the server. It allows a servlet to obtain configuration data when it is loaded. The methods declared by this interface are summarized here:

|  |  |
| --- | --- |
| **Method** | **Description** |
| ServletContext getServletContext( ) | Returns the context for this servlet. |
| String getInitParameter(String param) | Returns the value of the initialization parameter named param. |
| Enumeration getInitParameterNames( ) | Returns an enumeration of all  initialization parameter names. |
| String getServletName( ) | Returns the name of the invoking servlet. |

**The ServletContext Interface**

The ServletContext interface is implemented by the server. It enables servlets to obtain information about their environment. Several of its methods are summarized in Table 2.

|  |  |
| --- | --- |
| **Method** | **Description** |
| Object getAttribute(String attr) | Returns the value of the server attribute named attr. |
| String getMimeType(String file) | Returns the MIME type of file. |
| String getRealPath(String vpath) | Returns the real path that corresponds to the virtual path vpath. |
| String getServerInfo( ) | Returns information about the server. |
| void log(String s) | Writes s to the servlet log. |
| void log(String s, Throwable e) | Write s and the stack trace for e to the servlet log. |
| void setAttribute(String attr, Object val) | Sets the attribute specified by attr to the value passed in val. |
| **Table: 2 Various Methods Defined by ServletContext** | |

M.Sc. Semester – I (Computer Science) 7

Programming in Java Dr. M. T. Wanjari

**The ServletRequest Interface**

The ServletRequest interface is implemented by the server. It enables a servlet to obtain information about a client request. Several of its methods are summarized in Table 3

|  |  |
| --- | --- |
| **Method** | **Description** |
| Object getAttribute(String attr) | Returns the value of the attribute named attr. |
| String getCharacterEncoding( ) | Returns the character encoding of the request. |
| int getContentLength( ) | Returns the size of the request. The value –1 is returned if the size is unavailable. |
| String getContentType( ) | Returns the type of the request. A null value is returned if the type cannot be determined. |
| ServletInputStream getInputStream( ) throws IOException | Returns a ServletInputStream that can be used to read binary data from the request. An IllegalStateException is thrown if getReader( ) has already been invoked for this request. |
| String getParameter(String pname) | Returns the value of the parameter named pname. |
| Enumeration getParameterNames( ) | Returns an enumeration of the parameter names for this request. |
| String[] getParameterValues(String name ) | Returns an array containing values associated with the parameter specified by name. |
| String getProtocol( ) | Returns a description of the protocol. |
| BufferedReader getReader( )  throws IOException | Returns a buffered reader that can be used to read text from the request. An IllegalStateException is thrown if getInputStream( ) has already been invoked for this request. |
| String getRemoteAddr( ) | Returns the string equivalent of the client IP address. |
| String getRemoteHost( ) | Returns the string equivalent of the client host name. |
| String getScheme( ) | Returns the transmission scheme of the URL used for the request (for example, “http”, “ftp”). |
| String getServerName( ) | Returns the name of the server. |
| int getServerPort( ) | Returns the port number. |
| **Table 3: Various Methods Defined by ServletRequest** | |

.

**The ServletResponse Interface**

The ServletResponse interface is implemented by the server. It enables a servlet to formulate a response for a client. Several of its methods are summarized in Table 4.

|  |  |
| --- | --- |
| **Method** | **Description** |
| String getCharacterEncoding( ) | Returns the character encoding for the response. |
| ServletOutputStream  getOutputStream( )  throws IOException | Returns a ServletOutputStream that can be used to write binary data to the response. An IllegalStateException is thrown if getWriter( ) has already been invoked for this request. |
| PrintWriter getWriter( ) | Returns a PrintWriter that can be used to write character data |

M.Sc. Semester – I (Computer Science) 8

Programming in Java Dr. M. T. Wanjari

|  |  |
| --- | --- |
| throws IOException | to the response. An IllegalStateException is thrown if getOutputStream( ) has already been invoked for this request. |
| void setContentLength(int size) | Sets the content length for the response to size. |
| Void setContentType(String type) | Sets the content type for the response to type. |
| **Table 4: Various Methods Defined by ServletResponse** | |

**The SingleThreadModel Interface**

This interface is used to indicate that only a single thread will execute the service( ) method of a servlet at a given time. It defines no constants and declares no methods. If a servlet implements this interface, the server has two options. First, it can create several instances of the servlet. When a client request arrives, it is sent to an available instance of the servlet. Second, it can synchronize access to the servlet.

**The GenericServlet Class**

The GenericServlet class provides implementations of the basic life cycle methods for a servlet and is typically subclassed by servlet developers. GenericServlet implements the Servlet and ServletConfig interfaces. In addition, a method to append a string to the server log file is available. The syntax of the method is given below:

void log(String s)

void log(String s, Throwable e)

Here, s is the string to be appended to the log, and e is an exception that occurred. **The ServletInputStream Class**

The ServletInputStream class extends InputStream. It is implemented by the server and provides an input stream that a servlet developer can use to read the data from a client request. It defines the default constructor. In addition, a method is provided to read bytes from the stream. Its syntax is given below:

int readLine(byte[ ] buffer, int offset, int size) throws IOException

Here, buffer is the array into which size bytes are placed starting at offset. The method returns the actual number of bytes read or –1 if an end-of-stream condition is encountered. **The ServletOutputStream Class**

The ServletOutputStream class extends OutputStream. It is implemented by the server and provides an output stream that a servlet developer can use to write data to a client response.

M.Sc. Semester – I (Computer Science) 9

Programming in Java Dr. M. T. Wanjari

A default constructor is defined. It also defines the print( ) and println( ) methods, which output data to the stream.

**The Servlet Exception Classes**

javax.servlet defines two exceptions. The first is ServletException, which indicates that a servlet problem has occurred. The second is UnavailableException, which extends ServletException. It indicates that a servlet is unavailable.

**Reading Servlet Parameters**

The ServletRequest class includes methods that allow 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 HTML source code for PostParameters.htm is shown in the following code. 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/examples/servlet/PostParametersServlet">

<table>

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

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

</tr>

<tr> <td> <B> Phone </B> </td>

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

</tr>

</table>

<input type=submit value="Submit">

</body>

</html>

M.Sc. Semester – I (Computer Science) 10

Programming in Java Dr. M. T. Wanjari

The source code for PostParametersServlet.java is shown in the following code. 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. We can see the parameter name and value are output to the client. 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 out = response.getWriter();

// Get enumeration of parameter names.

Enumeration e = request.getParameterNames();

// Display parameter names and values.

while(e.hasMoreElements()) {

String pname = (String)e.nextElement();

out.print(pname + " = ");

String pvalue = request.getParameter(pname);

out.println(pvalue);

}

out.close();

}

}

Compile the servlet and perform these steps to test this example:

1. Start Tomcat (if it is not already running).

2. Display the Web page in a browser.

3. Enter an employee name and phone number in the text fields.

4. Submit the Web page.

M.Sc. Semester – I (Computer Science) 11

Programming in Java Dr. M. T. Wanjari

After following these steps, the browser will display a response that is dynamically generated by the servlet. The javax.servlet.http Package The javax.servlet.http package contains a number of interfaces and classes that are commonly used by servlet developers. Its functionality makes it easy to build servlets that work with HTTP requests and responses. The following table summarizes the core interfaces that are provided in this package:

|  |  |
| --- | --- |
| **Interface** | **Description** |
| HttpServletRequest | Enables servlets to read data from an HTTP request. |
| HttpServletResponse | Enables servlets to write data to an HTTP response. |
| HttpSession | Allows session data to be read and written. |
| HttpSessionBindingListener | Informs an object that it is bound to or unbound from a session. |

The following table summarizes the core classes that are provided in this package. The most important of these is HttpServlet. Servlet developers typically extend this class in order to process HTTP requests.

|  |  |
| --- | --- |
| **Class** | **Description** |
| Cookie | Allows state information to be stored on a client machine. |
| HttpServlet | Provides methods to handle HTTP requests and responses. |
| HttpSessionEvent | Encapsulates a session-changed event. |
| HttpSessionBindingEvent | Indicates when a listener is bound to or unbound from a session value, or that a session attribute changed. |

**The HttpServletRequest Interface**

The HttpServletRequest interface is implemented by the server. It enables a servlet to obtain information about a client request. Several of its methods are shown in Table 5.

|  |  |
| --- | --- |
| **Method** | **Description** |
| String getAuthType( ) | Returns authentication scheme. |
| Cookie[ ] getCookies( ) | Returns an array of the cookies in this request. |
| long getDateHeader(String field) | Returns the value of the date header field named field |
| String getHeader(String field) | Returns the value of the header field named field. |
| Enumeration getHeaderNames( ) | Returns an enumeration of the header names. |

M.Sc. Semester – I (Computer Science) 12

Programming in Java Dr. M. T. Wanjari

|  |  |
| --- | --- |
| int getIntHeader(String field) | Returns the int equivalent of the header field named field. |
| String getMethod( ) | Returns the HTTP method for this request. |
| String getPathInfo( ) | Returns any path information that is located after the servlet path and before a query string of the URL. |
| String getPathTranslated( ) | Returns any path information that is located after the servlet path and before a query string of the URL after translating it to a real path. |
| String getQueryString( ) | Returns any query string in the URL. |
| String getRemoteUser( ) | Returns the name of the user who issued this request. |
| String getRequestedSessionId( ) | Returns the ID of the session. |
| String getRequestURI( ) | Returns the URI. |
| StringBuffer getRequestURL( ) | Returns the URL. |
| String getServletPath( ) | Returns that part of the URL that identifies the servlet. |
| HttpSession getSession( ) | Returns the session for this request. If a session does not exist, one is created and then returned. |
| HttpSession getSession(boolean new) | If new is true and no session exists, creates and returns a session for this request. Otherwise, returns the existing session for this request. |
| boolean  isRequestedSessionIdFromCookie( ) | Returns true if a cookie contains the session ID. Otherwise, returns false. |
| boolean  isRequestedSessionIdFromURL( ) | Returns true if the URL contains the session ID. Otherwise, returns false. |
| boolean  isRequestedSessionIdValid( ) | Returns true if the requested session ID is valid in the current session context. |
| **Table 5: Various Methods Defined by HttpServletRequest** | |

**The HttpServletResponse Interface**

The HttpServletResponse interface is implemented by the server. It enables a servlet to formulate an HTTP response to a client. Several constants are defined. These correspond to the

M.Sc. Semester – I (Computer Science) 13

Programming in Java Dr. M. T. Wanjari

different status codes that can be assigned to an HTTP response. For example, SC\_OK indicates that the HTTP request succeeded and SC\_NOT\_FOUND indicates that the requested resource is not available. Several methods of this interface are summarized in Table 6.

|  |  |
| --- | --- |
| **Method** | **Description** |
| void addCookie(Cookie cookie) | Adds cookie to the HTTP response. |
| boolean containsHeader(String field) | Returns true if the HTTP response header contains a field named field. |
| String encodeURL(String url) | Determines if the session ID must be encoded in the URL identified as url. If so, returns the modified version of url. Otherwise, returns url. All URLs generated by a servlet should be processed by this method. |
| String encodeRedirectURL(String url) | Determines if the session ID must be encoded in the URL identified as url. If so, returns the modified version of url. Otherwise, returns url. All URLs passed to sendRedirect( ) should be processed by this method. |
| void sendError(int c) throws IOException | Sends the error code c to the client. |
| void sendRedirect(String url) throws IOException | Redirects the client to url. |
| void setDateHeader(String field, long msec) | Adds field to the header with date value equal to msec (milliseconds since midnight, January 1, 1970, GMT). |
| void setHeader(String field, String value) | Adds field to the header with value equal to value. |
| void setIntHeader(String field, int value) | Adds field to the header with value equal to value. |
| void setStatus(int code) | Sets the status code for this response to code. |
| **Table 6: Various Methods Defined by HttpServletResponse** | |

**The HttpSession Interface**

The HttpSession interface is implemented by the server. It enables a servlet to read and write the state information that is associated with an HTTP session. Several of its methods are

M.Sc. Semester – I (Computer Science) 14

Programming in Java Dr. M. T. Wanjari

summarized in Table 7. All of these methods throw an IllegalStateException if the session has already been invalidated.

|  |  |
| --- | --- |
| **Method** | **Description** |
| Object getAttribute(String attr) | Returns the value associated with the name passed in attr. Returns null if attr is not found. |
| Enumeration getAttributeNames( ) | Returns an enumeration of the attribute names associated with the session. |
| long getCreationTime( ) | Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when this session was created. |
| String getId( ) | Returns the session ID. |
| long getLastAccessedTime( ) | Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the client last made a request for this session. |
| void invalidate( ) | Invalidates this session and removes it from the context. |
| boolean isNew( ) | Returns true if the server created the session and it has not yet been accessed by the client. |
| void removeAttribute(String attr) | Removes the attribute specified by attr from the session. |
| void setAttribute(String attr, Object val) | Associates the value passed in val with the attribute name passed in attr. |
| **Table 7: The Methods Defined by HttpSession** | |

**The HttpSessionBindingListener Interface**

The HttpSessionBindingListener interface is implemented by objects that need to be notified when they are bound to or unbound from an HTTP session. The methods that are invoked when an object is bound or unbound are

void valueBound(HttpSessionBindingEvent e)

void valueUnbound(HttpSessionBindingEvent e)

Here, e is the event object that describes the binding.

**The Cookie Class**

The Cookie class encapsulates a cookie. A cookie is stored on a client and contains state information. Cookies are valuable for tracking user activities. For example, assume that a user

M.Sc. Semester – I (Computer Science) 15

Programming in Java Dr. M. T. Wanjari

visits an online store. A cookie can save the user’s name, address, and other information. The user does not need to enter this data each time he or she visits the store. A servlet can write a cookie to a user’s machine via the addCookie( ) method of the HttpServletResponse interface. The data for that cookie is then included in the header of the HTTP response that is sent to the browser.

The names and values of cookies are stored on the user’s machine. Some of the information that is saved for each cookie includes the following:

■ The name of the cookie

■ The value of the cookie

■ The expiration date of the cookie

■ The domain and path of the cookie

The expiration date determines when this cookie is deleted from the user’s machine. If an expiration date is not explicitly assigned to a cookie, it is deleted when the current browser session ends. Otherwise, the cookie is saved in a file on the user’s machine. The domain and path of the cookie determine when it is included in the header of an HTTP request. If the user enters a URL whose domain and path match these values, the cookie is then supplied to the Web server. Otherwise, it is not. There is one constructor for Cookie. It has the signature shown here:

Cookie(String name, String value)

Here, the name and value of the cookie are supplied as arguments to the constructor. The methods of the Cookie class are summarized in Table 8.

|  |  |
| --- | --- |
| **Method** | **Description** |
| Object clone( ) | Returns a copy of this object. |
| String getComment( ) | Returns the comment. |
| String getDomain( ) | Returns the domain. |
| int getMaxAge( ) | Returns the age (in seconds). |
| String getName( ) | Returns the name. |
| String getPath( ) | Returns the path. |
| boolean getSecure( ) | Returns true if the cookie must be sent using only a secure protocol. Otherwise, returns false. |
| String getValue( ) | Returns the value. |
| int getVersion( ) | Returns the cookie protocol version. (Will be 0 or 1.) |

M.Sc. Semester – I (Computer Science) 16

Programming in Java Dr. M. T. Wanjari

|  |  |
| --- | --- |
| void setComment(String c) | Sets the comment to c. |
| void setDomain(String d) | Sets the domain to d. |
| void setMaxAge(int secs) | Sets the maximum age of the cookie to secs. This is the number of seconds after which the cookie is deleted. Passing –1 causes the cookie to be removed when the browser is terminated. |
| void setPath(String p) | Sets the path to p. |
| void setSecure(boolean secure) | Sets the security flag to secure, which means that cookies will be sent only when a secure protocol is being used. |
| void setValue(String v) | Sets the value to v. |
| void setVersion(int v) | Sets the cookie protocol version to v, which will be 0 or 1. |
| **Table 8: The Methods Defined by Cookie** | |

**The HttpServlet Class**

The HttpServlet class extends GenericServlet. It is commonly used when developing servlets that receive and process HTTP requests. The methods of the HttpServlet class are summarized in Table 9.

|  |  |
| --- | --- |
| **Method** | **Description** |
| void doDelete(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException | Performs an HTTP DELETE. |
| void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException | Performs an HTTP GET. |
| void doHead(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException | Performs an HTTP HEAD. |
| void doOptions(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException | Performs an HTTP OPTIONS. |
| void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException | Performs an HTTP POST. |
| void doPut(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException | Performs an HTTP PUT. |
| void doTrace(HttpServletRequest req, HttpServletResponse res) | Performs an HTTP TRACE. |

M.Sc. Semester – I (Computer Science) 17

Programming in Java Dr. M. T. Wanjari

|  |  |
| --- | --- |
| throws IOException, ServletException |  |
| long getLastModified(HttpServletRequest req) | Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the requested resource was last modified. |
| void service(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException | Called by the server when an HTTP request arrives for this servlet. The arguments provide access to the HTTP request and  response, respectively. |
| **Table 9: The Methods Defined by HttpServlet** | |

**The HttpSessionEvent Class**

HttpSessionEvent encapsulates session events. It extents EventObject and is generated when a change occurs to the session. It defines this constructor:

HttpSessionEvent(HttpSession session)

Here, session is the source of the event.

HttpSessionEvent defines one method, getSession( ), which is shown here: HttpSession getSession( )

It returns the session in which the event occurred.

**The HttpSessionBindingEvent Class**

The HttpSessionBindingEvent class extends HttpSessionEvent. It is generated when a listener is bound to or unbound from a value in an HttpSession object. It is also generated when an attribute is bound or unbound. Here are its constructors:

HttpSessionBindingEvent(HttpSession session, String name)

HttpSessionBindingEvent(HttpSession session, String name, Object val)

Here, session is the source of the event and name is the name associated with the object that is being bound or unbound. If an attribute is being bound or unbound, its value is passed in val. The getName( ) method obtains the name that is being bound or unbound. It is shown here:

M.Sc. Semester – I (Computer Science) 18

Programming in Java Dr. M. T. Wanjari

String getName( )

The getSession( ) method, shown next, obtains the session to which the listener is being bound or unbound:

HttpSession getSession( )

The getValue( ) method obtains the value of the attribute that is being bound or unbound. It is shown here:

Object getValue( )

**Handling HTTP Requests and Responses**

The HttpServlet class provides specialized methods that handle the various types of HTTP requests. A servlet developer typically overrides one of these methods. These methods are doDelete( ), doGet( ), doHead( ), doOptions( ), doPost( ), doPut( ), and doTrace( ). However, the GET and POST requests are commonly used when handling form input.

**Handling HTTP GET Requests**

Here we will develop a servlet that handles an HTTP GET request. The servlet is invoked when a form on a Web page is submitted. The example contains two files. A Web page is defined in ColorGet.htm and a servlet is defined in ColorGetServlet.java. The HTML source code for ColorGet.htm is shown in the following code. It defines a form that contains a select element and a submit button. Notice that the action parameter of the form tag specifies a URL. The URL identifies a servlet to process the HTTP GET request.

<html>

<body>

<center>

<form name="Form1" action="http://localhost:8080/examples/servlet/ColorGetServlet"> <B>Color:</B>

<select name="color" size="1">

<option value="Red">Red</option>

<option value="Green">Green</option>

<option value="Blue">Blue</option>

</select>

<br><br>

<input type=submit value="Submit">

M.Sc. Semester – I (Computer Science) 19

Programming in Java Dr. M. T. Wanjari

</form>

</body>

</html>

The source code for ColorGetServlet.java is shown in the following code. The doGet( ) method is overridden to process any HTTP GET requests that are sent to this servlet. It uses the getParameter( ) method of HttpServletRequest to obtain the selection that was made by the user. A response is then formulated.

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class ColorGetServlet extends HttpServlet {

public void doGet(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

String color = request.getParameter("color");

response.setContentType("text/html");

PrintWriter pw = response.getWriter();

pw.println("<B>The selected color is: ");

pw.println(color);

pw.close();

}

}

Compile the servlet and perform these steps to test this example:

1. Start Tomcat, if it is not already running.

2. Display the Web page in a browser.

3. Select a color.

4. Submit the Web page.

After completing these steps, the browser will display the response that is dynamically generated by the servlet. One other point: Parameters for an HTTP GET request are included as part of the URL that is sent to the Web server. Assume that the user selects the red option and submits the form. The URL sent from the browser to the server is

M.Sc. Semester – I (Computer Science) 20

Programming in Java Dr. M. T. Wanjari

http://localhost:8080/examples/servlet/ColorGetServlet?color=Red

The characters to the right of the question mark are known as the query string. **Handling HTTP POST Requests**

Here we will develop a servlet that handles an HTTP POST request. The servlet is invoked when a form on a Web page is submitted. The example contains two files. A Web page is defined in ColorPost.htm and a servlet is defined in ColorPostServlet.java. The HTML source code for ColorPost.htm is shown in the following code. It is identical to ColorGet.htm except that the method parameter for the form tag explicitly specifies that the POST method should be used, and the action parameter for the form tag specifies a different servlet. <html>

<body>

<center>

<form name="Form1" method="post"

action="http://localhost:8080/examples/servlet/ColorPostServlet">

<B>Color:</B>

<select name="color" size="1">

<option value="Red">Red</option>

<option value="Green">Green</option>

<option value="Blue">Blue</option>

</select>

<br><br>

<input type=submit value="Submit">

</form>

</body>

</html>

The source code for ColorPostServlet.java is shown in the following code. The doPost( ) method is overridden to process any HTTP POST requests that are sent to this servlet. It uses the getParameter( ) method of HttpServletRequest to obtain the selection that was made by the user. A response is then formulated.

import java.io.\*;

import javax.servlet.\*;

M.Sc. Semester – I (Computer Science) 21

Programming in Java Dr. M. T. Wanjari

import javax.servlet.http.\*;

public class ColorPostServlet extends HttpServlet {

public void doPost(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

String color = request.getParameter("color");

response.setContentType("text/html");

PrintWriter pw = response.getWriter();

pw.println("<B>The selected color is: ");

pw.println(color);

pw.close();

}

}

Compile the servlet and perform the same steps as above.

Note: Parameters for an HTTP POST request are not included as part of the URL that is sent to the Web server. In this example, the URL sent from the browser to the server is: http://localhost:8080/examples/servlet/ColorGetServlet

The parameter names and values are sent in the body of the HTTP request.

**Using Cookies**

Now, let’s develop a servlet that illustrates how to use cookies. The servlet is invoked when a form on a Web page is submitted. The example contains three files as summarized here:

|  |  |
| --- | --- |
| **File** | **Description** |
| AddCookie.htm | Allows a user to specify a value for the cookie named MyCookie. |
| AddCookieServlet.java | Processes the submission of AddCookie.htm. |
| GetCookiesServlet.java | Displays cookie values. |

The HTML source code for AddCookie.htm is shown in the following code. This page contains a text field in which a value can be entered. There is also a submit button on the page. When this button is pressed, the value in the text field is sent to AddCookieServlet via an HTTP POST request.

<html>

M.Sc. Semester – I (Computer Science) 22

Programming in Java Dr. M. T. Wanjari

<body>

<center>

<form name="Form1" method="post"

action="http://localhost:8080/examples/servlet/AddCookieServlet">

<B>Enter a value for MyCookie:</B>

<input type=textbox name="data" size=25 value="">

<input type=submit value="Submit">

</form>

</body>

</html>

The source code for AddCookieServlet.java is shown in the following code. It gets the value of the parameter named “data”. It then creates a Cookie object that has the name “MyCookie” and contains the value of the “data” parameter. The cookie is then added to the header of the HTTP response via the addCookie( ) method. A feedback message is then written to the browser.

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class AddCookieServlet extends HttpServlet {

public void doPost(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

// Get parameter from HTTP request.

String data = request.getParameter("data");

// Create cookie.

Cookie cookie = new Cookie("MyCookie", data);

// Add cookie to HTTP response.

response.addCookie(cookie);

// Write output to browser.

response.setContentType("text/html");

PrintWriter pw = response.getWriter();

M.Sc. Semester – I (Computer Science) 23

Programming in Java Dr. M. T. Wanjari

pw.println("<B>MyCookie has been set to");

pw.println(data);

pw.close();

}

}

The source code for GetCookiesServlet.java is shown in the following code. It invokes the getCookies( ) method to read any cookies that are included in the HTTP GET request. The names and values of these cookies are then written to the HTTP response. Observe that the getName( ) and getValue( ) methods are called to obtain this information.

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class GetCookiesServlet extends HttpServlet {

public void doGet(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

// Get cookies from header of HTTP request.

Cookie[] cookies = request.getCookies();

// Display these cookies.

response.setContentType("text/html");

PrintWriter pw = response.getWriter();

pw.println("<B>");

for(int i = 0; i < cookies.length; i++) {

String name = cookies[i].getName();

String value = cookies[i].getValue();

pw.println("name = " + name + "; value = " + value);

}

pw.close();

}

}

Compile the servlet and perform these steps:

M.Sc. Semester – I (Computer Science) 24

Programming in Java Dr. M. T. Wanjari

1. Start Tomcat, if it is not already running.

2. Display AddCookie.htm in a browser.

3. Enter a value for MyCookie.

4. Submit the Web page.

After completing these steps will observe that a feedback message is displayed by the browser. Next, request the following URL via the browser:

http://localhost:8080/examples/servlet/GetCookiesServlet

Observe that the name and value of the cookie are displayed in the browser. In this example, an expiration date is not explicitly assigned to the cookie via the setMaxAge( ) method of Cookie. Therefore, the cookie expires when the browser session ends. We can experiment by using setMaxAge( ) and observe that the cookie is then saved to the disk on the client machine.

**Session Tracking**

HTTP is a stateless protocol. Each request is independent of the previous one. However, in some applications, it is necessary to save state information so that information can be collected from several interactions between a browser and a server. Sessions provide such a mechanism. A session can be created via the getSession( ) method of HttpServletRequest. An HttpSession object is returned. This object can store a set of bindings that associate names with objects. The setAttribute( ), getAttribute( ), getAttributeNames( ), and removeAttribute( ) methods of HttpSession manage these bindings. It is important to note that session state is shared among all the servlets that are associated with a particular client.

The following servlet illustrates how to use session state. The getSession( ) method gets the current session. A new session is created if one does not already exist. The getAttribute( ) method is called to obtain the object that is bound to the name “date”. That object is a Date object that encapsulates the date and time when this page was last accessed. (there is no such binding when the page is first accessed.) A Date object encapsulating the current date and time is then created. The setAttribute( ) method is called to bind the name “date” to this object.

import java.io.\*;

import java.util.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class DateServlet extends HttpServlet {

M.Sc. Semester – I (Computer Science) 25

Programming in Java Dr. M. T. Wanjari

public void doGet(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

// Get the HttpSession object.

HttpSession hs = request.getSession(true);

// Get writer.

response.setContentType("text/html");

PrintWriter pw = response.getWriter();

pw.print("<B>");

// Display date/time of last access.

Date date = (Date)hs.getAttribute("date");

if(date != null) {

pw.print("Last access: " + date + "<br>");

}

// Display current date/time.

date = new Date();

hs.setAttribute("date", date);

pw.println("Current date: " + date);

}

}

When first request this servlet, the browser displays one line with the current date and time information. On subsequent invocations, two lines are displayed. The first line shows the date and time when the servlet was last accessed. The second line shows the current date and time.

**JSP:**

**Introduction to JSP**

Java Server Pages (JSP) is a technology for developing web pages that supports dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which <% and end with %>.

A Java Server Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write

M.Sc. Semester – I (Computer Science) 26

Programming in Java Dr. M. T. Wanjari

JSPS as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically. JSP tags can be used for a variety of purposes, such as retrieving information from database or registering user preferences, accessing JavaBeans components, pass controls between pages and sharing information between requests, pages etc. JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression language, jstl etc.

A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc. **Why Use JSP?**

Java Server Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

• Performance is significantly better because JSP allows embedding dynamic Elements in HTML Pages itself instead of having separate CGI files.

• JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested. • Java Server Pages are built on top of the Java Servlets API, so like Servlet; JSP also access to all the powerful Enterprise Java APIs**,** including JDBC, JNDI, EJB, JAXP etc.

• JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

**Advantages of JSP**

Following is the list of other advantages of using JSP over other technologies:

M.Sc. Semester – I (Computer Science) 27

Programming in Java Dr. M. T. Wanjari

• **vs. Active Server Pages (ASP):** The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non Microsoft Web servers.

• **vs. Pure Servlets:** It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML**.**

• **vs. Server-Side Includes (SSI):** SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like. • **vs. JavaScript:** JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

• **vs. Static HTML:** Regular HTML, of course, cannot contain dynamic information. • **Extension to Servlet-** JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP that makes JSP development easy. • **Easy to maintain-** JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

• **Fast Development:** No need to recompile and redeploy- If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

• **Less code than Servlet-** In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc. **Types Of Jsp Tags (Scripting Tags)**

In JSP, Java code can be written inside the jsp page using the scriptlet tag. **JSP Scripting elements**

The scripting elements provide the ability to insert java code inside the jsp. There are three types of scripting elements:

• scriptlet tag

• expression tag

• declaration tag

M.Sc. Semester – I (Computer Science) 28

Programming in Java Dr. M. T. Wanjari

**JSP scriptlet tag-** A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

<% java source code %>

**Example of JSP scriptlet tag**

In this example, we are displaying a welcome message.

<html>

<body>

>% out.print("welcome to jsp"); %>

</body>

</html>

**Example of JSP scriptlet tag that prints the user name-** In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

**File: index.html**

<html>

<body>

<form action="welcome.jsp">

</form>

</body>

</html>

**File: welcome.jsp**

<html>

<body>

<%="Welcome "+request.getParameter("uname") %>

</body>

</html>

**JSP Declaration Tag**- The JSP declaration tag is used to declare fields and methods. The code written inside the jsp declaration tag is places outside the service() method of auto generated servlet.

**Syntax of JSP declaration tag**

The syntax of the declaration tag is as follows:

M.Sc. Semester – I (Computer Science) 29

Programming in Java Dr. M. T. Wanjari

<%! field or method declaration %>

**Difference between JSP Scriptlet tag and Declaration tag**

|  |  |
| --- | --- |
| **JSP Scriptlet Tag** | **Jsp Declaration Tag** |
| The jspscriptlet tag can only declare variables not methods. | The jsp declaration tag can declare variables as well as methods. |
| The declaration of scriptlet tag is  placed inside the jspService() method | The declaration of jsp declaration tag is placed outside the jspService() method |

**Example of JSP declaration tag that declares field-** In this example of JSP declaration tag, we are declaring the field and printing the value of the declared field using the jsp expression tag. **index.jsp**

<html>

<body>

<%! int data=50; %>

<%= "Value of the variable is:"+data %>

</body>

</html>

**Example of JSP declaration tag that declares method-** In this example of JSP declaration tag, we are defining the method which returns the cube of given number and calling this method from the jsp expression tag. But we can also use jsp scriptlet tag to call the declared method. **index.jsp**

<html>

<body>

<%!

int cube(int n){

return n\*n\*n\*;}

%>

<%= "Cube of 3 is:"+cube(3) %>

</body>

<input type=“text” name=“uname>

<input type=“submit” value=go”><br>

M.Sc. Semester – I (Computer Science) 30

Programming in Java Dr. M. T. Wanjari

</form>

</body>

</html>

**File: welcome.jsp**

<html>

<body>

<%

String name=request.getParameter(“uname”);

out.print(“welcome “+name);

%>

</form>

</body>

</htrnl>

**JSP expression tag-** The code placed within JSP expression tag is written to the output stream of the response. So you need not write out.print() to write data. It Is mainly used to print the values of variable or method.

Syntax of JSP expression tag

<%= statement %>

**Example of JSP expression tag –** In this example of jsp expression tag, we are simply displaying a welcome message.

<html>

<body>

<%= “welcome to jsp” %>

</body>

</html>

**Example of JSP expression tag that prints current time-** To display the current time, we have used the getTime() method of Calendar class. The getTime() is an instance method of Calendar class, so we have called it after getting the instance of Calendar class by the getInstance() method.

**index.jsp**

<html>

M.Sc. Semester – I (Computer Science) 31

Programming in Java Dr. M. T. Wanjari

<body>

Current Time: <%= java.util.Calendar.getInstance().getTime() %>

</body>

</html>

**Example of JSP expression tag that prints the user name-** In this example, we are printing the username using the expression tag. The index.html file gets the username and sends the request to the welcome.jsp file, which displays the username.

**File: index.jsp**

<html>

<body>

<form action=“welcome.jsp”>

<input type=“text” name=“uname”><br>

<input type=“submit” value=“go”>

**Example: Write a program in java to insert product information in product table using html and jsp. Following is Product.html :**

*Enter the Product Details*

|  |  |
| --- | --- |
| Product Id |  |
| Description |  |
| Quantity |  |
| Rate |  |
| register reset | |

**Product.html**

<html>

<head> <title> Product Information </title>

</head>

<body background=”maroon”>

<form name=”form” method=”post” action=””>

<center>

<h1> Enter the product Details </h1>

<table>

M.Sc. Semester – I (Computer Science) 32

Programming in Java Dr. M. T. Wanjari

<tr> <td> Product Id: </td>

<td> <Input type=”textbox” name=”pi” size=”25” value=” “> </td> </tr>

<tr> <td> Description: </td>

<td> <Input type=”textbox” name=”desc” size=”25” value=” “> </td> </tr>

<tr> <td> Quality: </td>

<td> <Input type=”textbox” name=”quant” size=”25” value=” “> </td>

</tr>

<tr> <td> Rate: </td>

<td> <Input type=”textbox” name=”rate” size=”25” value=” “> </td> </tr>

<tr> <td> <Input type=”submit” value=”Submit”> </td>

<td> <Input type=”reset” value=”Reset”> </td>

</tr>

</table>

</center>

</form>

</body>

</html>

**Product.jsp**

<%@page import=”java.io.\*”%>

<%@page import=”java.sql.\*”%>

<%@page import=”javax.servlet.\*”%>

<%@page import=”javax.servlet.http.\*”%>

**<**html**>** <head> <title> Product Information </title>

M.Sc. Semester – I (Computer Science) 33

Programming in Java Dr. M. T. Wanjari

**<**body background=”maroon”**>**

**<**h1**>** Product Details </h1**>**

**<**%

try

{

Connection con;

Statement stmt;

ResultSet rs;

PreparedStatement pst;

String pid=request.getParameter(“pi”);

String description=request.getParameter(“desc”);

String quantity=request.getParameter(“quant”);

String rate=request.getParameter(“rate”);

Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);

con=DriverManager.getConnection(“jdbc:odbc:Data”); String query=”insert into product(Productid, description, quantity, rate) values(?,?,?,?)”;

pst=con.prepareStatement(query);

pst.setString(1, pid);

pst.setString(2, desc);

pst.setString(3, quantity);

pst.setString(4, rate);

out.println(“Record Successfully Inserted”) ;

pst.executeUpdate();

}

catch(Exception e)

{

M.Sc. Semester – I (Computer Science) 34

Programming in Java Dr. M. T. Wanjari

e.printStackTrace();

}

%>

<a href=”product.html”> Back </a>

</body>

</html>

**Introduction to JavaBeans**

**JavaBeans Concept**

Consider a scenario where a software company provides customized software solutions developed in java for different platforms. Typically, these software solutions have a module for verifying the login credentials and performing database connectivity. Every time a new application is developed, the developers have to write these modules. In this situation, some reusable module can be developed for verifying the login credentials and performing database connectivity. This module will reduce the development time and will be productive and efficient. To develop such reusable parts of an application, you can use software components.

**Software Components and JavaBeans**

A software component is reusable object that can be plugged into any software application. You can reuse or extend the functionality of a software component on the target application. For example, spell-check is a software component that can be used in multiple applications, such as MS Word and MS Excel. Other examples of pluggable software components can be a calculator, text editor or find and replace utility.

You can develop software components using various programming languages such as C, C++, Java and Visual Basic. JavaBeans are software components developed using java language. JavaBeans are reusable and self-contained software components, which take advantage of all the security and platform-independent features of java.

A JavaBean exists in one of the following three phases of development.

▪ Construction phase

▪ Build phase

▪ Execution phase

M.Sc. Semester – I (Computer Science) 35

Programming in Java Dr. M. T. Wanjari

The construction phase involves creation of a JavaBean and its user interface. A JavaBean cannot run by itself. In order to access its functionality, it must be placed inside some other element of a running application called the container. A container can be a frame window or an applet. The build phase involves placing the JavaBean into the container. When the container application is executed, the JavaBean enters the execution phase.

**Elements of a JavaBean**

The elements of a JavaBean enable the multiple. JavaBean components in an application to interact with the container in which they are placed. There are three fundamental elements of the JavaBean components. The elements of the element of the JavaBean components are:

▪ **Properties:** refer to the private data members of a JavaBean that define the behavior of the JavaBean. For example, the chat JavaBean can have properties such as the backgroundColor, size and font to change the background color size of chat window and font of the text displayed in the chat window respectively.

▪ **Methods:** Refer to the public member function of a JavaBean that are used to modify the JavaBean properties. Other applications can access the JavaBean can contain methods that allows a user to chat or change the background color of the chat window.

▪ **Events:** Refer to the messages that one JavaBean component sends to another component. Events enable a JavaBean to communicate changes in the property values or state of a JavaBean to other JavaBeans. Events are generated when you perform an action such as clicking a mouse button. For example events are generated when you send a message in the chat JavaBean.

**The JavaBean Component Specification**

Sun Microsystems has provided specifications to develop JavaBeans. As per the JavaBean component specification, a java object must support the following features to become a JavaBean:

▪ **Customization:** Is the ability of a JavaBean to allow its properties to be changed during the build and executing phases. A JavaBean should allow some or all its properties to be customized by the target application. For example the yahoo messenger service bean allows you to specify what has to be done when a user is online (whether an alert message should be displayed or a beep sound should be produces) and configure friends list (whether to show a list of users who are online or not)

M.Sc. Semester – I (Computer Science) 36

Programming in Java Dr. M. T. Wanjari

▪ **Persistence:** Is the ability of a JavaBean to save its state to a disk or any other storage device when the execution of the container application containing the JavaBean is terminated. The JavaBean is reloaded with the same setting when the application is reloaded. For example, the state and configuration setting of the messenger service bean such as the user with whom you had a chat last time be saved to a file. The user and configuration setting are retrieved when the JavaBean is reloaded.

▪ **Communication:** Is the ability of a JavaBean to inform changes in the attribute to other JavaBeans and to the container application. For example, when you install yahoo messenger on your machine, a separate toolbar icon and a menu option gets added to the browser window (container). Also, when you disconnect your computer (system that hosts the bean) from the internet, the yahoo chat window also gets closed.

▪ **Introspection:** Is the ability of a JavaBean to allow an external application to query the properties, methods and events supported by it. This feature is mainly used by a programmer to obtain information about the class file of a JavaBean.

**Services of JavaBean Components**

Multiple JavaBean components can be grouped inside a container. For example, frame window or a Web page is a container that can be user to group various JavaBean components. The JavaBean components provide the following services:

▪ **Builder support:** Refer to the Graphical User Interface (GUI) that enables you create and group various JavaBeans in an application. You can group JavaBean with the help of events using the builder tools such as BeanBox.

▪ **Layout:** Allows multiple JavaBean components to be arranged in a develop environment.

▪ **Interface publishing:** Allows the multiple JavaBean components in an application to communicate with each other with help of public interfaces. The public interfaces are the methods of JavaBean that are accessible by the other JavaBean components in the application. When a JavaBean component calls a service that is provide by some other component JavaBean, the public interfaces are used to transfer the request and the response.

▪ **Event Handling:** Refers to the firing and handling of the events associated when the JavaBean components in an application. The various JavaBean components in an

M.Sc. Semester – I (Computer Science) 37

Programming in Java Dr. M. T. Wanjari

application can interact with one another and fire and handle events using the event handling services of JavaBeans.

▪ **Persistence:** Enables you to save the last state of a JavaBean. When the JavaBean is reloaded, the last saved configuration setting for the JavaBean are automatically set. **Types of JavaBeans**

The JavaBean API provide classes and interfaces that enable you to create three of JavaBeans. They are listed in the following table:

|  |  |
| --- | --- |
| **Bean type** | **Description** |
| Control JavaBeans | Are used to create GUI components that can be plugged into any application. You can create a control JavaBean by extending one of the swing component classes. An animated button control is an example of a control JavaBean. They are also reffered to as User Interface or visible JavaBeans. |
| Container JavBeans | Are used to hold other JavaBeans. You can create a container bean by extending one of the swing container classes. An Explorer window created by extending the JTree class is an example of a container bean. Also known as visible JavaBeans. |
| Invisible Runtime JavaBeans | Are used to create components that perform a specific task in the background of an application. They are invisible to the end user during the execution phase of the bean. A spell Checker is an example of this type of bean. A bean that provides connectivity to any kind of data source such as an SQL server database, an Excel worksheet or an flat file is also an example of an invisible runtime bean. They are also referred to as hidden JavaBeans. |

The JavaBeans, Sun Microsystems has provided a development environment, Bean Development Kit (BDK).

**Advantages of Java Beans:**

• The java beans posses the property of “Write once and run anywhere”. Beans can work in different local platforms.

M.Sc. Semester – I (Computer Science) 38

Programming in Java Dr. M. T. Wanjari

• Beans have the capability of capturing the events sent by other objects and vice versa enabling object communication.

• The properties, events and methods of the bean can be controlled by the application developer.(ex. Add new properties)

• Beans can be configured with the help of auxiliary software during design time.(no hassle at runtime)

• The configuration setting can be made persistent.(reused)

• Configuration setting of a bean can be saved in persistent storage and restored later. **Beans Development Kit (BDK)**

Is a development environment to create, configure, and test JavaBeans. The features of BDK environment are:

• Provides a GUI to create, configure, and test JavaBeans.

• Enables you to modify JavaBean properties and link multiple JavaBeans in an application using BDK.

• Provides a set of sample JavaBeans.

• Enables you to associate pre-defined events with sample JavaBeans.

Identifying BDK Components

• Execute the run.bat file of BDK to start the BDK development environment. • The components of BDK development environment are:

• **ToolBox**

• **BeanBox**

• **Properties**

• **Method Tracer**

**ToolBox window**: Lists the sample JavaBeans of BDK. The following figure shows the **ToolBox** window:

M.Sc. Semester – I (Computer Science) 39

Programming in Java Dr. M. T. Wanjari

**BeanBox window**:

Is a workspace for creating the layout of JavaBean application.

• The following figure shows the **BeanBox** window:

**Properties window**:

Displays all the exposed properties of a JavaBean. You can modify JavaBean properties in the properties window.

The following figure shows the **Properties** window:

M.Sc. Semester – I (Computer Science) 40

Programming in Java Dr. M. T. Wanjari

**Method Tracer window**:

Displays the debugging messages and method calls for a JavaBean application. The following figure shows the **Method Tracer** window:

**User Defined JavaBean**

Besides using the sample JavaBeans of BDK, you can also create user defined JavaBeans using various classes and interfaces that are define in the java.beans pacage. To create user defined JavaBean class, the following coding conventions need to be followed.

▪ Implement java.io.serializable interface to develop persistant JavaBeans. ▪ Define an empty constrictor to initialize the instances of a JavaBeans.

▪ Define the accessor and mutator methods to retrieve and specify the values of private properties of a JavaBean, respectively.

To run the user-defined JavaBean class on BDK, it needs to be packaged. The packaging process involves compiling the user defined JavaBean and creating the manifest file and the Java Archive (JAR) file.

M.Sc. Semester – I (Computer Science) 41

Programming in Java Dr. M. T. Wanjari

**Accessor and Mutator Methods**

The JavaBean properties are the private data members of a JavaBean class. the accessor and mutator methods are used to publicly access the JavaBean properties. The accessor methods enable you to retrieve the value of a JavaBean property while the mutator method enables you to specify the value of JavaBean property. The accessor method is also known as the get method and the mutator method as set method. you can use the following code to define the accessor and mutator methods to access dictionary path and language properties of the SpellCheck JavaBean:

import java.beans.\*;

import java.io.Serializable;

public class SpellCheck implements java.io.Serializable

{

. .

. .

private String lang;

private String dictionaryPath;

. .

. .

/\* Define the get method to retrieve value of Lang property of the JavaBean\*/ public String getLang()

{

return lang;

/\* Define the set method to specify the value of Lang property of the JavaBean\*/ public void setLang(String s)

lang = s;

/\*Define the get method to retrieve the value of dictionaryPath property JavaBean\*/ public String getDictionaryPath()

return dictionaryPath;

/\* Define the set method to specify the value of dictionaryPath property of the JavaBean\*/ public void setDictionaryPath (String s)

{

dictionaryPath = s;

M.Sc. Semester – I (Computer Science) 42

Programming in Java Dr. M. T. Wanjari

}

. .

. .

}

In the above code, the SpellCheck JavaBean implements the java.io.Seriliazable interface. The getDictionaryPath() and setDictionaryPath() methods enable the user to retrieve and set the path of the dictionary for check respectively. Similarly, getLang() and setLang() methods allows user to retrieve and specify the language of the spell check. Above coding, you need to compile the program to generate the class file, which is classified in a manifest file.

**Steps to Develop a User-Defined JavaBean:**

1. Create a directory for the new bean

2. Create the java bean source file(s)

3. Compile the source file(s)

4. Create a manifest file

5. Generate a JAR file

6. Start BDK

7. Load Jar file

8. Test.

**1. Create a directory for the new bean**

Create a directory/folder like C:\Beans

**2. Create bean source file - MyBean.java**

import java.awt.\*;

public class MyBean extends Canvas

{

public MyBean()

{

setSize(70,50);

setBackground(Color.green);

}

}

**3. Compile the source file(s)**

M.Sc. Semester – I (Computer Science) 43

Programming in Java Dr. M. T. Wanjari

C:\Beans >Javac MyBean.java

**4. Create a manifest file and Generate a JAR file**

**Manifest and JAR Files**

The manifest file for a JavaBean application contains a list of all the class files that make up a JavaBean. There is a special entry for the JavaBean classes in the manifest file. The entry in the manifest file enables the target application to recognize the JavaBean classes for an application. You can compress and group all the files for a java application within a Java Archive (JAR) files. The JAR files are platform independent files that are used to store the classes and the associated resources of a java application. For example, an application can contain multiple files such as image files, sound files and Hyper Text Markup Language (HTML) files. You can all the files of an application in a JAR file.

**Manifest File**

Manifest file is a text file that contains information on the content of a JAR file, as the names of the JavaBean class files. The manifest files are the text files added in the text editor, such as notepad. These files have .mtf extension. You need to specify the name of the class file for the JavaBean in the manifest file. You can use the following code to define the information in a manifest file:

Name: SpellCheck.class

Java-Bean: True

In the preceding code, the statement, Java-Bean: True indicates that the SpellCheck.class is a JavaBean.

You need to ensure the following rules while creating a manifest file:

▪ You need to press the Enter key after typing each line in the manifest file. ▪ There should be space after the colon.

▪ There should be a hyphen between Java and Bean.

▪ There should not be any blank line separating the Name and the Java-Bean entry. **Java Archive File**

The files of JavaBean application are compressed and packaged as a JAR file to reduce the size and the download time of the files. J2SDK provides a utility called JAR utility that enables you to create and manage a JAR file for a Java application. JAR utility is a command

M.Sc. Semester – I (Computer Science) 44

Programming in Java Dr. M. T. Wanjari

line tool that is located in the bin directory of J2SDK. The syntax to invoke the JAR utility from the command prompt is:

jar <option> <file\_name>

In the preceding code syntax, jar is a command that is used to invoke the JAR utility and option refers to the option for using the JAR utility. The filr names is a list of files for a JavaBean application that are stored in the JAR file created using the JAR utility. The name of the JAR file that you need to create is also specified as the first file in the file\_names list. The various option that you can specify while creating a JAR file are:

▪ c: Indicates that a new JAR file is created.

▪ f: Indicates that the first file in the file\_names list is the name of the JAR file. ▪ m: Indicates that the second file in the filr\_names list is the name of the manifest file for the JavaBean application.

▪ t: Indicates that all the files and resources in the JAR file are to be displayed in a tabular format.

▪ v: Indicates that the JAR file should generate a verbose output. The verbose output displays the list of all the files that are compressed in the JAR file such as the manifest file. In addition, all the metadata information about the JAR file is also displayed as output.

▪ x: Indicates that the files and resources of a JAR file are to be extracted. ▪ o: Indicates that the JAR file should not be compressed.

▪ M: Indicates that the manifest file is not created.

You can use the following command to create a JAR file with the name SpellCheck.jar that contains the class file and manifest file for the JavaBean publication:

jar cfm SpellCheck.jar SpellCheck.mft SpellCheck.class

**5. Start BDK**

Go to->

C:\bdk1\_1\beans\beanbox

Click on **run.bat** file. When we click on run.bat file the BDK software automatically started. **6. Load Jar file**

Go to

M.Sc. Semester – I (Computer Science) 45

Programming in Java Dr. M. T. Wanjari

Beanbox->File->Load jar. Here we have to select our created jar file when we click on ok, our bean(userdefined) MyBean appear in the ToolBox.

**7. Test our created user defined bean**

Select the MyBean from the ToolBox when we select that bean one + simple appear then drag that Bean in to the Beanbox. If you want to apply events for that bean, now we apply the events for that Bean.

**Introspection:**

▪ Introspection can be defined as the technique of obtaining information about bean properties, events and methods.

▪ Basically introspection means analysis of bean capabilities.

▪ Introspection is the automatic process by which a builder tool finds out which properties, methods, and events a bean supports.

▪ Introspection describes how methods, properties, and events are discovered in the beans that you write.

▪ This process controls the publishing and discovery of bean operations and properties ▪ Without introspection, the JavaBeans technology could not operate.

**BDK Introspection:**

▪ Allows automatic analysis of a java beans component

▪ Enables a builder tool to analyze how a bean works.

(Or)

▪ A mechanism that allows classes to publish the operations and properties they support and a mechanism to support the discovery of such mechanism.

▪ Introspection can be defined as the technique of obtaining information about bean properties, events and methods.

▪ Basically introspection means analysis of bean capabilities.

There are two ways in which the developer of a Bean can indicate which of its properties, events, and methods should be exposed by an builder tool. With the first method, simple naming conventions are used. These allow the introspection mechanisms to infer information about a Bean. In the second way, an additional class is provided that explicitly supplies this information.

M.Sc. Semester – I (Computer Science) 46

Programming in Java Dr. M. T. Wanjari

**Design patterns for JavaBean Properties:-**

A property is a subset of a Bean’s state. A bean *property* is a named attribute of a bean that can affect its behavior or appearance. Examples of bean properties include color, label, font, font size, and display size. Properties are the private data members of the JavaBean classes.

Properties are used to accept input from an end user in order to customize a JavaBean. Properties can retrieve and specify the values of various attributes, which determine the behavior of a JavaBean.

**Types of JavaBeans Properties**

· Simple properties

· Boolean properties

· Indexed properties

**Simple Properties:**

Simple properties refer to the private variables of a JavaBean that can have only a single value. Simple properties are retrieved and specified using the get and set methods respectively. A read/write property has both of these methods to access its values. The **get method** used to read the value of the property .The **set method** that sets the value of the property. The setXXX() and getXXX() methods are the heart of the java beans properties mechanism. This is also called getters and setters. These accessor methods are used to set the property . The syntax of get method is:

**public return\_type get<PropertyName>()**

**public T getN();**

**public void setN(T arg)**

\_ N is the name of the property and T is its type

**Ex:**

public double getDepth()

{

return depth;

}

Read only property has only a get method.

The syntax of set method is:

**public void set<PropertyName>(data\_type value)**

M.Sc. Semester – I (Computer Science) 47

Programming in Java Dr. M. T. Wanjari

**Ex:**

public void setDepth(double d)

{

Depth=d;

}

Write only property has only a set method.

**Boolean Properties**:

A Boolean property is a property which is used to represent the values True or False. Have either of the two values, TRUE or FALSE. It can identified by the following methods: Syntax:

Let N be the name of the property and T be the type of the value then

public boolean isN();

public void setN(boolean parameter);

public Boolean getN();

• public boolean is<PropertyName>()

• public boolean get<PropertyName>()

First or second pattern can be used to retrieve the value of a Boolean.

• public void set<PropertyName>(boolean value)

For getting the values isN() and getN() methods are used and for setting the Boolean values setN() method is

used.

**Ex**:

public boolean dotted=false;

public boolean isDotted()

{

return dotted;

}

public void setDotted(boolean dotted)

{

this.dotted=dotted;

}

M.Sc. Semester – I (Computer Science) 48

Programming in Java Dr. M. T. Wanjari

**Indexed Properties**:

Indexed Properties are consists of multiple values. If a simple property can hold an array of value they are no longer called simple but instead indexed properties. The method’s signature has to be adapted accordingly. An indexed property may expose set/get methods to read/write one element in the array (so-called ’index getter/setter’) and/or so-called ’array getter/setter’ which read/write the entire array.

Indexed Properties enable you to set or retrieve the values from an array of property values. Indexed Properties are retrieved using the following get methods:

**Syntax:** public int[] get<PropertyName>()

**Ex:**

private double data[];

public double getData(int index)

{

return data[index];

}

**Syntax:**public property\_datatype get<PropertyName>(int index)

**Ex:**

public void setData(int index,double value)

{

Data[index]=value;

}

Indexed Properties are specified using the following set methods:

**Syntax**:

public void set<PropertyName>(int index, property\_datatype value)

**EX**:

public double[] getData()

{

return data; }

**Syntax** :

public void set<PropertyName>(property\_datatype[] property\_array)

**Ex**:

M.Sc. Semester – I (Computer Science) 49

Programming in Java Dr. M. T. Wanjari

public void setData(double[] values)

{ }

The properties window of BDK does not handle indexed properties. Hence the output can not be displayed here.

**Bound Properties**:

A bean that has a bound property generates an event when the property is changed. Bound Properties are the properties of a JavaBean that inform its listeners about changes in its values. Bound Properties are implemented using the **PropertyChangeSupport** class and its methods. Bound Properties are always registered with an external event listener.

The event is of type ***PropertyChangeEvent*** and is sent to objects that previously egistered an interest in receiving such notifications bean with bound property - Event source Bean implementing listener -- event target In order to provide this notification service a JavaBean needs to have the following two methods:

public void addPropertyChangeListener(PropertyChangeListener p) {

changes.addPropertyChangeListener(p);

}

public void removePropertyChangeListener(PropertyChangeListener p) { changes.removePropertyChangeListener(p);

}

PropertyChangeListener is an interface declared in the java.beans package. Observers which want to be notified of property changes have to implement this interface, which consists of only one method:

public interface PropertyChangeListener extends EventListener {

public void propertyChange(PropertyChangeEvent e );

}

**Constrained Properties:**

It generates an event when an attempt is made to change it value Constrained Properties are implemented using the ***PropertyChangeEvent*** class. The event is sent to objects that previously registered an interest in receiving an notification those other objects have the ability to veto the proposed change This allows a bean to operate differently according to the runtime environment. A bean property for which a change to the property results in validation by another

M.Sc. Semester – I (Computer Science) 50

Programming in Java Dr. M. T. Wanjari

bean. The other bean may reject the change if it is not appropriate. Constrained Properties are the properties that are protected from being changed by other JavaBeans. Constrained Properties are registered with an external event listener that has the ability to either accept or reject the change in the value of a constrained property. Constrained Properties can be retrieved using the get method. The prototype of the get method is:

**Syntax:** public string get<ConstrainedPropertyName>()

Can be specified using the set method. The prototype of the set method is: **Syntax** :public string set<ConstrainedPropertyName>(String str)throws PropertyVetoException **Design Patterns for Events:**

**Handling Events in JavaBeans:**

Enables Beans to communicate and connect together. Beans generate events and these events can be sent to other objects. Event means any activity that interrupts the current ongoing activity.

**Example: mouse clicks, pressing key…**

User-defined JavaBeans interact with the help of user-defined events, which are also called custom events. You can use the Java event delegation model to handle these custom events. The components of the event delegation model are:

• **Event Source**: Generates the event and informs all the event listeners that are registered with it. • **Event Listener**: Receives the notification, when an event source generates an event. • **Event Object**: Represents the various types of events that can be generated by the event sources.

**Creating Custom Events:**

The classes and interfaces that you need to define to create the custom JavaBean events are:

• An event class to define a custom JavaBean event.

• An event listener interface for the custom JavaBean event.

• An event handler to process the custom JavaBean event.

• A target Java application that implements the custom event.

**Creating the Event Class:**

The event class that defines the custom event extends the EventObject class of the java.util package. For example,

M.Sc. Semester – I (Computer Science) 51

Programming in Java Dr. M. T. Wanjari

public class NumberEvent extends EventObject

{

public int number1,number2;

public NumberEvent(Object o,int number1,int number2)

{

super(o);

this.number1=number1;

this.number2=number2; } }

Beans can generate events and send them toother objects.

**Creating Event Listeners**

• When the event source triggers an event, it sends a notification to the event listener interface. • The event listener interface implements the java.util.EventListener interface. • Syntax:

• public void addTListener(TListener eventListener);

• public void addTListener(TListener eventListener)throws TooManyListeners; • public void removeTListener(TListener eventListener);

• The target application that uses the custom event implements the custom listener. For example, public interface NumberEnteredListener extends EventListener

{

public void arithmeticPerformed(NumberEvent mec);

}

**Creating Event Handler**

Custom event handlers should define the following methods:

• addXXListener(): Registers listeners of a JavaBean event.

• fireXX(): Notifies the listeners of the occurrence of a JavaBean event.

• removeXXListener(): Removes a listener from the list of registered listeners of a JavaBean. **The code snippet to define an event handler for the custom event NumberEvent is:** public class NumberBean extends JPanel implements ActionListener

{

public NumberBean() { }

NumberEnteredListener mel;

public void addNumberListener(NumberEnteredListener mel) {

this.mel = mel; } }

**---------------------------------------------------------------------------------------------------------------------**

M.Sc. Semester – I (Computer Science) 52