Thakral College of technology Bhopal

SESSION 2011-2012

MAJOR TRAINING REPORT

ON

Developing J2EE Application using Web and Business Components

Submitted as partial fulfilment of the requirement for the award of

Degree of B.E. in Computer Science and Engg.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI PROUDYOGIKI VISHWAVIDHYALAYA, BHOPAL

SUBMITTED TO:
SUBMITTED BY:

MR. MANISH AGRAWAL SANCHITA HARNE

LECTURER, CSE DEPTT.
0126CS081090

Thakral College of technology

Bhopal

SESSION 2011-2012

DEPARTMENT OF COMPUTER SCIENCE AND ENGINNERING

CERTIFICATE

This is to certify that dissertation entitled “Computer Networking” has been carried out by Sanchita Harne of B.E. VII Semester in Computer Science and Engineering under our supervision and guidance.

She has submitted the Major Training Synopsis towards partial fulfilment of the award of the degree of Bachelor of Engineering of Rajiv Gandhi Vishwavidhyalaya, Bhopal (MP) during the academic year 2011-2012.

Mr. Manish Agrawal

Lecturer, CSE Dept.

ACKNOWLEDGEMENT

‘Knowledge is expression of experience gained in life. It is the choicest possession that should be happily shared with others.’

In this regard we feel great pleasure in submitting this Major Training Report on “Computer Networking”. During this training we received a lot of help, advice and co-operation from our esteemed faculty and other distinguished persons.

We wish to express our profound sense of gratitude to Ms. JayaSwamy our training instructors, for her valuable guidance through the course of training without whose encouragement the project wouldn’t have been a success. We would also like to thank our training guide Mr.Manish Agrawal, whose willingness and spontaneous suggestion encourage us tremendously to pursue our goal.

We are grateful to college authorities for their support and all those who have directly or indirectly helped us during the training session.

CONTENT

Abstract

The objective of undertaking the industrial training is to provide a work experience so that students engineering knowledge is enhanced. Industrial training also provides an opportunity to students to select an engineering problem and possibly an industry guide. It is essential to bridge the gap between the classroom and industrial environment. This will enrich their practical learning and will be better equipped to integrate the practical experiences with the classroom learning process.

With this Objective this major training session of 4 weeks has been successfully finished at the NIIT. Here we were taught about the various concepts present in J2EE. The technique to develop web based applications so that users from distant systems can access the application.

The training was full of practical as well as theoretical knowledge. The concepts were first taught in classroom and then these concepts were implemented in a program.

During the complete training period, we were taught to use all the classes, interfaces and the packages and implement them properly.

About JAVA ENTERPRISE EDITION

Java Platform, Enterprise Edition or **Java EE** is widely used platform for server programming in the Java programming language. The Java platform(Enterprise Edition) differs from the Java Standard Edition Platform (Java SE) in that it adds libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server.

Nomenclature, standards, and specifications

The platform was known as Java 2 Platform*, Enterprise Edition* or *J2EE* until the name was changed to *Java EE* in version 5. The current version is called *Java EE 6*.

Java EE is defined by its specification. As with other Java Community Process specifications, providers must meet certain conformance requirements in order to declare their products as *Java EE compliant*.

Java EE includes several API specifications, such as JDBC, RMI, e-mail, JMS, web services, XML, etc., and defines how to coordinate them. Java EE also features some specifications unique to Java EE for components. These include Enterprise JavaBeans, Connectors, servlets, portlets (following the Java Portlet specification), Java Server Pages and several web service technologies. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. A Java EE application server can handle transactions, security, scalability, concurrency and management of the components that are deployed to it, in order to enable developers to concentrate more on the business logic of the components rather than on infrastructure and integration tasks.

General APIs

The Java EE APIs includes several technologies that extend the functionality of the base Java SE APIs.

- Java EE 6 Platform Packages
- Java EE 5 Platform Packages

javax.faces.*

This package defines the root of the Java Server Faces (JSF) API. JSF is a technology for constructing user interfaces out of components

javax.faces.component.*

This package defines the component part of the Java Server Faces (JSF) API. Since JSF is primarily component oriented, this is one of the core packages. The package overview contains a UML diagram of the component hierarchy.

javax.servlet.*

The servlet specification defines a set of APIs to service mainly HTTP requests. It includes the Java Server Pages specification.

javax.enterprise.inject.*

These packages define the injection annotations for the contexts and Dependency Injection (CDI) API.

javax.enterprise.context.*

These packages define the context annotations and interfaces for the Contexts and Dependency Injection (CDI) API.

javax.ejb.*

The Enterprise JavaBean (EJB) specification defines a set of lightweight APIs that an object container (the EJB container) will support in order to provide transactions (using JTA), remote procedure calls (using RMI or RMI-IIOP), concurrency control, dependency injection and access control for business objects. This package contains the Enterprise JavaBeans classes and interfaces that define the contracts between the enterprise bean and its clients and between the enterprise bean and the ejb container.

javax.validation

This package contains the annotations and interfaces for the declarative validation support offered by the Bean Validation API. Bean Validation provides a unified way to provide constraints on beans (e.g. JPA model classes) that can be enforced cross-layer. In Java EE, JPA honors bean validation constraints in the persistence layer, while JSF does so in the view layer.

javax.persistence

This package contains the classes and interfaces that define the contracts between a persistence provider and the managed classes and the clients of the Java Persistence API (JPA).

javax.transaction

This package provides the Java Transaction API (JTA) API that contains the interfaces to interact with the transaction support offered by Java EE. Even though this API abstracts from the really low-level details, it is itself also considered somewhat low-level and the average application developer in Java EE is assumed to be relying on transparent handling of transactions by the higher level EJB abstractions.

javax.jms.*

This package defines the Java Message Service (JMS) API. The JMS API provides a common way for Java programs to create, send, receive and read an enterprise messaging system's messages.

javax.resource.*

This package defines the Java EE Connector Architecture (JCA) API. Java EE Connector Architecture (JCA) is a Java-based technology solution for connecting application servers and enterprise information systems (EIS) as part of enterprise application integration (EAI) solutions. This is a low-level API aimed at vendors that the average application developer typically does not come in contact with.

TRAINING DETAIL

NIIT
technologies
Trust us to find the way

NIIT Technologies is a leading IT solutions organization, servicing customers in North America, Europe, Middle East, Asia and Australia. The Company offers services in Application Development and Maintenance, Managed Services, Cloud Computing and Business Process Outsourcing to organizations in the Financial Services, Insurance, Travel, Transportation and Logistics, Manufacturing and Distribution and Government sectors.

The Company’s deep domain knowledge and new approaches to customer experience management with robust outsourcing capabilities, and a dual shore delivery model, have made NIIT Technologies a preferred IT partner for global majors in these chosen industries. Profound and enduring customer engagements have become a hallmark of NIIT Technologies.

NIIT Technologies vision is to be the “First Choice” of services for the focused segments serviced. The Company has a simple strategy - to focus and differentiate. It competes on the strength of its specialization.Over the years the Company has forged extremely rewarding relationships with global majors, a testimony to mutual commitment and its ability to retain marquee clients, drawing repeat business from them. Whether it is global banking and insurance major, ING, leading Asset Management solutions provider, SEI, the Number Two cement manufacturer, Holcim or travel big-wigs, British Airways and Sabre, NIIT Technologies has been able to scale its interactions with these marquee clients into extremely meaningful, multi-year "collaborations."

TECHNOLOGY

<u>System requirements</u>

System requirements for Windows Virtual PC:

- Computer running Windows 7 (all editions except Starter)
- 1+ GHz processor (32- or 64-bit)
- 1.25 GB memory required, 2 GB recommended
- Additional 15 GB of hard disk space per virtual Windows environment recommended
- Optional: if the processor supports hardware-assisted virtualization technology such as AMD-V or Intel-VT, it will be used. Before March 19, 2010, such a processor was mandatory.

Day-Wise Report

DAY 1

Objective: To understand the basics of J2EE and the topics to be covered in the training module.

J2EE is Java, optimized for enterprise computing. Officially J2EE stands for Java 2 Platform, Enterprise Edition. Unlike the traditional Java, which is often used to build client enhancements, J2EE is designed to build server applications. Officially, the J2EE platform is "a set of coordinated specifications and practices that together enable solutions for developing, deploying, and managing" such server applications.

As an enterprise platform, the J2EE environment extends basic Java with tools that "provide a complete, stable, secure, and fast Java platform to the enterprise level."

TOPICS TO BE COVERED DURING THE TRAINING :

- JDBC 4.0
- HTML
- CSS
- JAVA SCRIPT
- SERVLET
- JSP (JavaServer Pages)
- Java Beans
- EL (Expression language)
- Custom / Simple Tags
- JSTL
- Introduction to Struts

DAY 2

Objective: Understanding JDBC 4.0

JDBC 4.0 adds some functionality to the core API. These features are present only in a JDK 1.6 or higher environment.

These features are:

- Data Sources: To support the JDBC 4.0 ease of development, Derby introduces new implementations of *javax.sql.DataSource*.
- Auto loading of JDBC drivers: In earlier versions of JDBC, applications had to manually register drivers before requesting Connections. With JDBC 4.0, applications no longer need to issue a Class.forName() on the driver name; instead, the DriverManager will find an appropriate JDBC driver when the application requests a Connection.
- SQLExceptions: JDBC 4.0 introduces refined subclasses of SQLException.
- Wrappers: JDBC 4.0 introduces the concept of wrapped JDBC objects. This is a formal mechanism by which application servers can look for vendor-specific extensions inside standard JDBC objects like Connections, Statements, and ResultSets. For Derby, this is a vacuous exercise because Derby does not expose any of these extensions.
- Statement events: With JDBC 4.0, Connection pools can listen for Statement closing and Statement error events. New methods were added to *javax.sql.PooledConnection*: *addStatementEventListener* and*removeStatementEventListener*.
- Streaming APIs: JDBC 4.0 adds new overloads of the streaming methods in CallableStatement, PreparedStatement, and ResultSet. These are the *setXXX* and *updateXXX* methods which take*java.io.InputStream* and *java.io.Reader* arguments. The new overloads allow you to omit the length arguments or to specify *long* lengths.

- New methods: New methods were added to the following interfaces: java.sql.Connection, java.sql.DatabaseMetaData, java.sql.Statement.

Sample Code:

```
import java.sql.*
 public class JDBCSample {
 public static void main( String args[]) {

String connectionURL =
"jdbc:postgresql://localhost:5432/movies;user=java;password=samples
";
try {
    // Load the Driver class.
   Class.forName("org.postgresql.Driver");

   //Create the connection using the static getConnection method
Connection con = DriverManager.getConnection (connectionURL);

   //Create a Statement class to execute the SQL statement
   Statement stmt = con.createStatement();

   //Execute the SQL statement and get the results in a Resultset
   ResultSet rs = stmd.executeQuery("select moviename, releasedate
from movies");

   while (rs.next())
      System.out.println("Name= " + rs.getString("moviename") + "
Date= " + rs.getString("releasedate");
}
catch (SQLException e) {
   e.printStackTrace();
}
catch (Exception e) {
   e.printStackTrace();
}
finally {
   // Close the connection
   con.close();
}
}
}
```

DAY 5

Objective: About HTML

(HTML): HTML is the building block for web pages.

HTML is a format that tells a computer how to display a web page. The documents themselves are plain text files with special "tags" or codes that a web browser uses to interpret and display information on your computer screen.

- ƒ HTML stands for Hyper Text Markup Language
- ƒ An HTML file is a text file containing small markup tags
- ƒ The markup tags tell the Web browser how to display the page
- ƒ An HTML file must have an htm or html file extension

SAMPLE:

```
<html>

<head>

<title>My First Webpage</title>

</head>

<body>

This is my first homepage. <b>This text is bold</b>

</body>

</html>
```

DAY 7

Objective: What is CSS and why it is used?

A CSS (cascading style sheet) file allows you to separate your web sites (X) HTML content from its style. As always you use your (X) HTML file to arrange the content, but all of the presentation (fonts, colours, background, borders, text formatting, link effects & so on…) are accomplished within a CSS.

CSS is used either internally or externally.

Internal Stylesheet

First we will explore the internal method. This way you are simply placing the CSS code within the <head></head> tags of each (X) HTML file you want to style with the CSS. The format for this is shown in the example below.

```
<head>
<title><title>
<style type=”text/css”>
CSS Content Goes Here
</style>
</head>
<body>
```

With this method each (X)HTML file contains the CSS code needed to style the page. Meaning that any changes you want to make to one page will have to be made to all. This method can be good if you need to style only one page, or if you want different pages to have varying styles.

External Stylesheet

Next we will explore the external method. An external CSS file can be created with any text or HTML editor such as “Notepad” or “Dreamweaver”. A CSS file contains no (X)HTML, only CSS. You simply save it with the .css file extension. You can link to the file externally by placing one of the following links in the head section of every (X)HTML file you want to style with the CSS file.

```
<link rel=”stylesheet” type=”text/css” href=“Path To stylesheet.css” />
```

SAMPLE CODE:

```
<html>
<head>
<style type="text/css">
body
{
background-color:#b0c4de;
}
</style>
</head>

<body>

<h1>My CSS web page!</h1>
<p>Hello world! This is a W3Schools.com example.</p>

</body>
</html>
```

DAY 9

Objective: What is Java Script?

JavaScript is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

JavaScript was formalized in the ECMAScript language standard and is primarily used in the form of client-side JavaScript, implemented as part of a Web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment.

JavaScript's use in applications outside Web pages — for example in PDF documents, site-specific browsers, and desktop widgets — is also significant. Newer and faster JavaScript VMs and frameworks built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications.

JavaScript uses syntax influenced by that of C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages.

DAY 11

Objective: Everything about Servlet

A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by Web servers. Thus, it can be thought of as a Java Applet that runs on a server instead of a browser.

A **Servlet** is a Java class in Java EE that conforms to the **Java Servlet API**, a protocol by which a Java class may respond to requests. They are not tied to a specific client-server protocol, but are most often used with the HTTP protocol. Therefore, the word "Servlet" is often used in the meaning of "HTTP Servlet". Thus, a software developer may use a servlet to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets are the Java counterpart to non-Java dynamic Web content technologies such as CGI and ASP.NET. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or URL rewriting.

To deploy and run a Servlet, a Web container must be used. A Web container is essentially the component of a Web server that interacts with the servlets.

The servlet API, contained in the Java package hierarchy javax.servlet, defines the expected interactions of the Web container and a servlet.

A Servlet is an object that receives a request and generates a response based on that request. The basic servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package javax.servlet.http defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the Web server and a client. Servlets may be packaged in a WAR file as a Web application.

DAY 13

Objective: Explaining JavaServer Pages

JavaServer Pages (**JSP**) is a Java technology that helps software developers serve dynamically generated web pages based on HTML, XML, or other document types. It was released in 1999 as Sun's answer to ASP and PHP.JSP was designed to address the perception that the Java programming environment didn't provide developers with enough support for the Web.

To deploy and run, a compatible web server with servlet container is required. The Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems and the JCP (Java Community Process) must both be supported by the container.

<u>Sample Code:</u>

```
<%@ page errorPage="myerror.jsp" %>

 <%@ page import="com.foo.bar" %>

  <html>

 <head>

 <%! int serverInstanceVariable = 1;%>

 <% int localStackBasedVariable = 1; %>

 <table>

 <tr><td><%= toStringOrBlank( "expanded inline data " + 1 )
%></td></tr>
```

<u>DAY 15</u>

Objective: Understanding JavaBeans

JavaBeans are reusable software components for Java. Practically, they are classes written in the Java programming language conforming to a particular convention. They are used to encapsulate many objects into a single object (the bean), so that they can be passed around as a single bean object instead of as multiple individual objects.

Advantages of JavaBeans

- A Bean obtains all of the benefits of Java's "write once, run anywhere" paradigm.
- The properties, events, and methods of a Bean that are exposed to another application can be controlled.
- Auxiliary software can be provided to help configure a Bean.
- The configuration settings of a Bean can be saved in a persistent storage and can be 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.

DAY 18

Objective: What is Expression Language?

Expression Language (EL) is a scripting language which allows access to Java components (JavaBeans) through JSP. Since JSP 2.0, it has been used inside JSP tags to separate Java code from JSP, and to allow easier access to Java components (than in Java code).

Evolution of EL occurred to make scripting easier for web-content designers who have little or practically no knowledge of the core Java Language. This scripting language makes JSP a scripting language in the true sense. Before EL, JSP consisted of some special tags like scriptlets, expressions etc within which Java code was written explicitly. With EL the web-content designer needs only to know how to make proper calls to core Java methods and can enjoy the true scripting flavor of a scripting language.

EL is, both syntactically and semantically, similar to JavaScript expressions:

- there is no typecasting
- type conversions are usually done implicitly
- double and single quotes are equivalent
- object.property has the same meaning as object['property']

EL also liberates the programmer from having to know the particularities of how the values are actually accessed: object.property can mean (depending on what the object is) eitherobject.get("property") or object.getProperty("property") or object. getProperty() etc.

<u>DAY 20</u>

Objective: Working on Custom/Simple tags

CUSTOM TAGS: The standard JSP tags simplify JSP page development and maintenance. Custom tags are distributed in a **tag**

library, which defines a set of related custom tags and contains the objects that implement the tags. The object that implements a custom tag is called a **tag handler**. JSP technology defines two types of tag handlers: simple and classic. **Simple** tag handlers can be used only for tags that do not use scripting elements in attribute values or the tag body.

SIMPLE TAGS: A simple tag handler subclasses a support class called 'SimpleTagSupport'. This class is a very handy implementation of the 'SimpleTag' interface. It provides implementations of all 5 of this interface's methods, the most important of which is
the doTag() method. The doTag() method
in SimpleTagSupport actually does nothing — it's up to you, the developer, to override this method and code your tag's functionality.

```
package demo.tags;

import javax.servlet.jsp.tagext.*;

import javax.servlet.jsp.*;
<br>
public class Greeter extends SimpleTagSupport
{ <br>  <br>
public void doTag() throws JspException
{ <br><br>
PageContext pageContext = (PageContext) getJspContext(); <br>
JspWriter out = pageContext.getOut(); <br> <br>
try {
<br> <br>
out.println("Hello World");
<br><br>
    } catch (Exception e)
       { <br>
         // Ignore. <br> }<br><br> } <br> }
```

DAY 23

Objective: What do you understand by JSTL

JavaServer Pages

Standard Tag Library

The JavaServer Pages Standard Tag Library (JSTL) encapsulates, as simple tags, core functionality common to many JSP applications. For example, instead of suggesting that you iterate over lists using a scriptlet or different iteration tags from numerous vendors, JSTL defines a standard tag that works the same everywhere. This standardization lets you learn a single tag and use it on multiple JSP containers.JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization and locale-sensitive formatting tags, and SQL tags. It also introduces a new expression language to simplify page development, and it provides an API for developers to simplify the configuration of JSTL tags and the development of custom tags that conform to JSTL conventions.

Name	Description	Jar File Name
JSTL API classes	JSTL API classes	jstl.jar
JSTL implementation classes	Standard Taglib JSTL implementation classes	standard.jar

The standard tag library also has the following dependencies:

- JAXP 1.2
- Xalan 2.5
- JDBC Standard Extension 2.0

DAY 26

Objective: Introduction to Struts

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 libaries 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.

Almost all tags provided by the Struts framework use the following attributes:

Attribute	Used for
id	the name of a bean for temporary use by the tag
name	the name of a pre-existing bean for use with the tag
property	the property of the bean named in the name attribute for use with the tag
scope	the scope to search for the bean named in the name attribute

DAY 28

Objective: Practical lectures
Every concept that was taught had to be implemented in these days. At least two programs per concept were made to implement.
A sample code that was implemented is:

```
package jsp_servlet;

 import java.util.*;

 import java.io.*;
 import javax.servlet.*;

 import javax.servlet.http.*;

 import javax.servlet.jsp.*;

 import javax.servlet.jsp.tagext.*;


 import com.foo.bar; // Imported as a result of <%@ page import="com.foo.bar"
%>

 import …


 class _myservlet implements javax.servlet.Servlet,
javax.servlet.jsp.HttpJspPage {

   // Inserted as a

   // result of <%! int serverInstanceVariable = 1;%>

   int serverInstanceVariable = 1;

   …


   public void _jspService( javax.servlet.http.HttpServletRequest request,

   javax.servlet.http.HttpServletResponse response )

   throws javax.servlet.ServletException,

   java.io.IOException

   {

      javax.servlet.ServletConfig config = …; // Get the servlet config

      Object page = this;

      PageContext pageContext = …;           // Get the page context for this
request

      javax.servlet.jsp.JspWriter out = pageContext.getOut();

      HttpSession session = request.getSession( true );

      try {
            out.print( "<html>\r\n" );
            out.print( "<head>\r\n" );
            …
            // From <% int localStackBasedVariable = 1; %>
            int localStackBasedVariable = 1;
            …
            out.print( "<table>\r\n" );
            out.print( " <tr><td>" );
            // From <%= toStringOrBlank( "expanded inline data " + 1 ) %>
            out.print( toStringOrBlank( "expanded inline data " + 1 ) );
            out.print( " </td></tr>\r\n" );
            …
        } catch ( Exception _exception ) {
            // Clean up and redirect to error page in <%@ page
errorPage="myerror.jsp" %>
        }
    }
}
```

Summary

The Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications. The J2EE platform simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically, without complex programming.

The J2EE platform takes advantage of many features of the Java 2 Platform, Standard Edition (J2SE), such as "Write Once, Run

Anywhere" portability, JDBC API for database access, CORBA technology for interaction with existing enterprise resources, and a security model that protects data even in internet applications. Building on this base, the Java 2 Platform, Enterprise Edition adds full support for Enterprise JavaBeans components, Java Servlets API, JavaServer Pages and XML technology. The J2EE standard includes complete specifications and compliance tests to ensure portability of applications across the wide range of existing enterprise systems capable of supporting the J2EE platform. In addition, the J2EE specification now ensures Web services interoperability through support for the WS-I Basic Profile.

REFERENCES

For creating the report, we made use of the Internet, Books provided by the institute and the guidance of our faculty.

The websites visited for this purpose are:

- http://www.wikipedia.org
- http://www.roseindia.com
- http://www.java.sun.com
- http://www.webopedia.com
- http://www.j2eebrain.com

The books referenced are:

- Black book
- Java for the Web with Servlets, JSP, and EJB
- CodeNotes for J2EE: EJB, JDBC, JSP, and Servlets