JAVA BOOTCAMP

DEPLOYMENT

DEPLOYMENT

TOPICS:

- Intro to deployment
- Executable JAR files
- Java Web Start

INTRO TO DEPLOYMENT

WHAT IS DEPLOYMENT?

Once we've created a Java application, we need to make it available to our users. Today, we'll go over a few different ways to accomplish this task.

DEPLOYING AN APPLICATION

DEPLOYMENT STRATEGIES

1. Executable JAR file
2. Java Web Start
3. Installer program

INSTALLER PROGRAM

DESCRIPTION

- An *installer* program creates an install file for any operating system we might want our users to be able to install our application on.
- This is a particularly good option for professional applications with larger budgets.
- We will not go into much detail about this option, but you should know that it exists.

INSTALLER PROGRAM

Pros	Cons
The app installs and runs like a professional application.	The code is not automatically updated after the program is installed.
The correct version of Java can be installed as part of the application process.	Most installer programs are expensive commercial products.
There are no significant security restrictions.	This approach requires more work to set up and configure.

EXECUTABLE JAR FILE

EXECUTABLE JAR FILE

DESCRIPTION

- An *executable JAR (Java Archive)* file stores all the classes and resources required to run an application.
- One method of deploying an app is to distribute this JAR file and provide instructions on how to use it.
- This method requires additional setup of prerequisite files, so it isn't ideal for more complex applications or those with many users.

EXECUTABLE JAR FILE

Pros	Cons
There are no significant security restrictions.	The correct version of the JRE is not installed automatically.
	The code is not automatically updated.

EXECUTABLE JAR FILE

SYNTAX TO RUN EXECUTABLE JAR FILE

```
Java –jar JarName.jar
```

EXECUTABLE JAR FILE

A BATCH (.BAT) FILE THAT RUNS A CONSOLE APP ON WINDOWS

```
::Change to the directory that stores the Jar file
cd \murach\java\dist

::Use the java command to run the Jar file
Java –jar ch23_FutureValueConsole.jar
```

EXECUTABLE JAR FILE

A BASH (.SH) FILE THAT RUNS A CONSOLE APP ON MAC OS X/LINUX

```
#!/bin/bash

#Change to the directory that stores the Jar file
cd /murach/java/dist

#Use the java command to run the Jar file
Java –jar ch23_FutureValueConsole.jar
```

JAVA WEB START

JAVA WEB START

DESCRIPTION

- *Java Web Start* allows the user to install our application by downloading it from the web.
- This approach requires a bit more work up front, but may be worth the effort because it automates installation of the JRE and updates to the application.
- There are significant security restrictions applied to our application by default.

JAVA WEB START

Pros	Cons
The correct version of Java is automatically installed.	There are some significant security restrictions.
The code is automatically updated.	

JAVA WEB START

PROCEDURE

1. Create a JAR file that contains the class files and resources for the app
2. Create a JNLP file
3. Create an HTML file that includes a link to the JNLP file
4. Place the JAR, JNLP, HTML files in the right directories

JAVA WEB START

PROCEDURE

1. Display the HTML file in a browser, click on the link to the JNLP file
2. Modify the JNLP/HTML files to run on remote web server
3. Copy JAR/JNLP/HTML files to appropriate directories on web server to make them user-accessible
4. Display HTML file in browser; click link to JNLP file to launch app

JAVA WEB START

```
<jnlp>
<title>
<vendor>
<offline-allowed/>
<j2se>
<jar>
<application-desc>
<update>
```

RECAP

RECAP

WHAT YOU SHOULD KNOW AT THIS POINT

- What deployment means.
- The different ways to do deployment.
- Procedure to do Installer deployment.
- Procedure to do Executable Jar File.
- Procedure to do Java Web start.

JDBC

JDBC

TOPICS

- Working with JDBC
- Connection pooling

WORKING WITH JDBC

DATABASE DRIVERS

DOWNLOAD A DATABASE DRIVER

- For MySQL databases, we can download Connector/J from the MySQL website. This is a free, open-source, type-4 driver.
- For other databases, we can usually download a type-4 JDBC driver from the database's website.

DATABASE DRIVERS

MAKE THE DRIVER AVAILABLE TO AN APPLICATION

We need to make the driver available to our application. The easiest way to do so is to use the IDE to add the JAR file for the driver to the application.

CONNECTING TO A DATABASE

REQUIREMENTS

- We need to connect to the database before we can get or modify the data in it.
- We use the *getConnection* method of the *DriverManager* class to return a *Connection* object.
- The *getConnection* method requires a URL for the database, username, and password.

RESULTS SETS

RETURNING A RESULT SET

- We use the *createStatement* method of a *Connection* object to create a *Statement* object.
- We then use the *executeQuery* method of the Statement object to execute a *SELECT* statement that returns a *ResultSet* object.

RESULTS SETS

Method	Description
next()	Moves the cursor to the next row in the result set.
last()	Moves the cursor to the last row in the result set.
close()	Releases the result set's resources.
getRow()	Returns an int value that identifies the current row of the result set.

RESULTS SETS

RETRIEVE DATA FROM A RESULT SET

The *getXXX* methods can be used to return all eight primitive data types. They can also return strings, dates, and times.

Method	Description
getXXX(int columnIndex)	Returns data from the specified column number.
getXXX(String columnName)	Returns data from the specified column name.

RESULTS SETS

RETURNING RESULT SETS

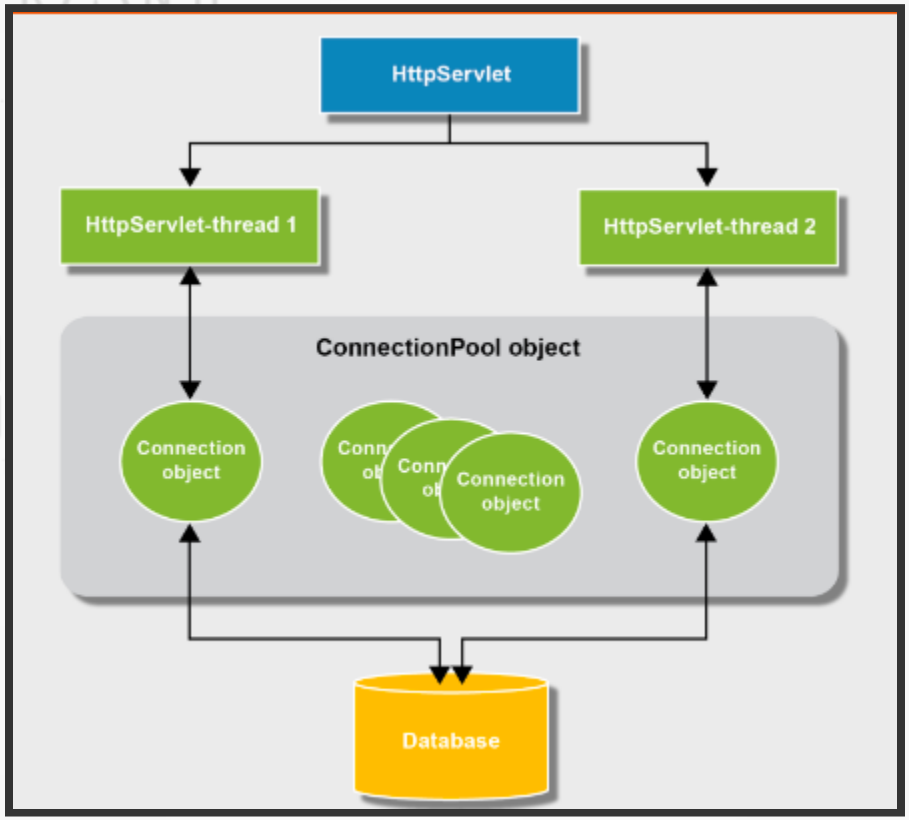
- We use the *createStatement* method of a *Connection* object to create a *Statement* object.
- We then use the *executeQuery* method of the Statement object to execute a *SELECT statement* that returns a *ResultSet* object.

CONNECTION POOLING

WHAT IS CONNECTION POOLING?

It's common practice to create a collection of Connection objects and store them in another object called a *database connection pool (DBCP)*.

CONNECTION POOLING



CONNECTION POOLING

USING A CONNECTION POOL

The *ConnectionPool* class provides the *getConnection* and *freeConnection* methods that make it easy for programmers to get connections and to return connections to the connection pool.

RECAP

RECAP

WHAT YOU SHOULD KNOW AT THIS POINT:

- Know the types of database drivers
- Declare and use database drivers
- Know how to connect to databases
- Know how to return result sets
- Define connection pooling and why it is important
- Using connection pooling

INTRODUCTION TO JSP

INRODUCTION TO JSP

TOPICS

- JSP Introduction
- JSP tags
- EL and JSTL
- Working with the JSTL core library

JSP INTRODUCTION

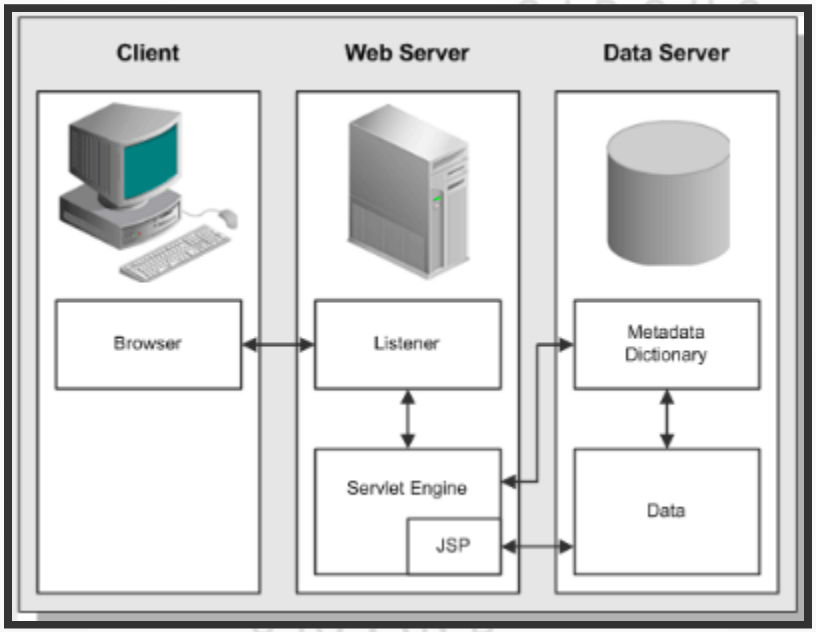
DYNAMIC WEB APPLICATIONS

WHAT IS A DYNAMIC WEB APPLICATION?

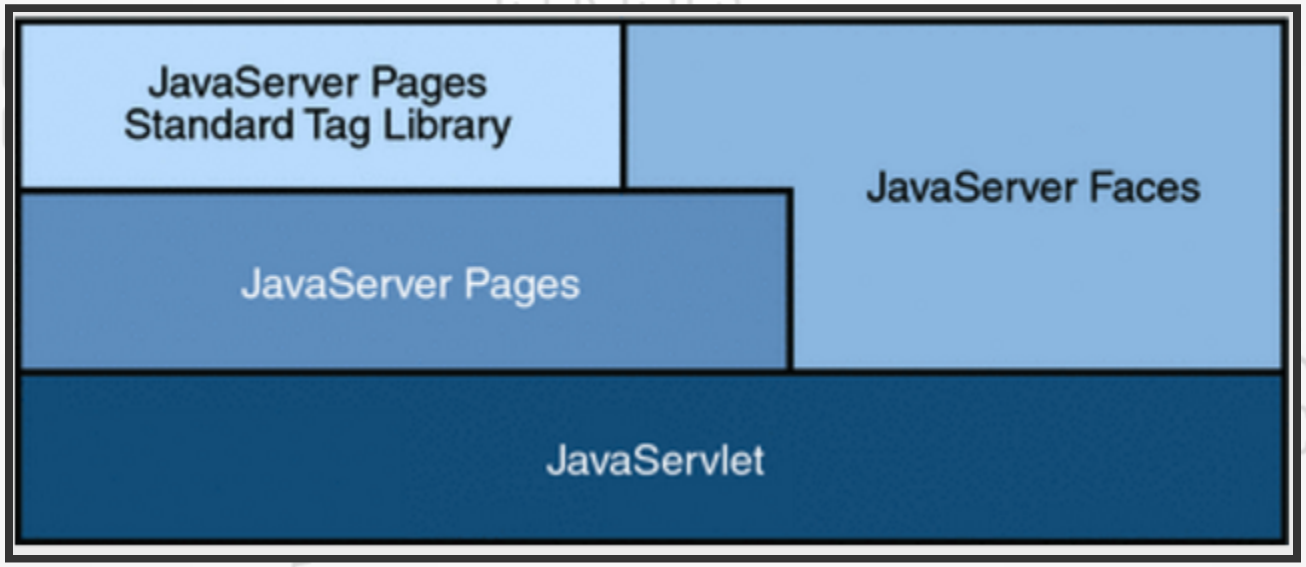
- The content of a web page can change based on parameters.
- Technologies: JSP, php, ASP.NET, Ruby on rails, ...

DYNAMIC WEB APPLICATIONS

WHAT IS A DYNAMIC WEB APPLICATION?



JAVA WEB APPLICATIONS TECHNOLOGIES

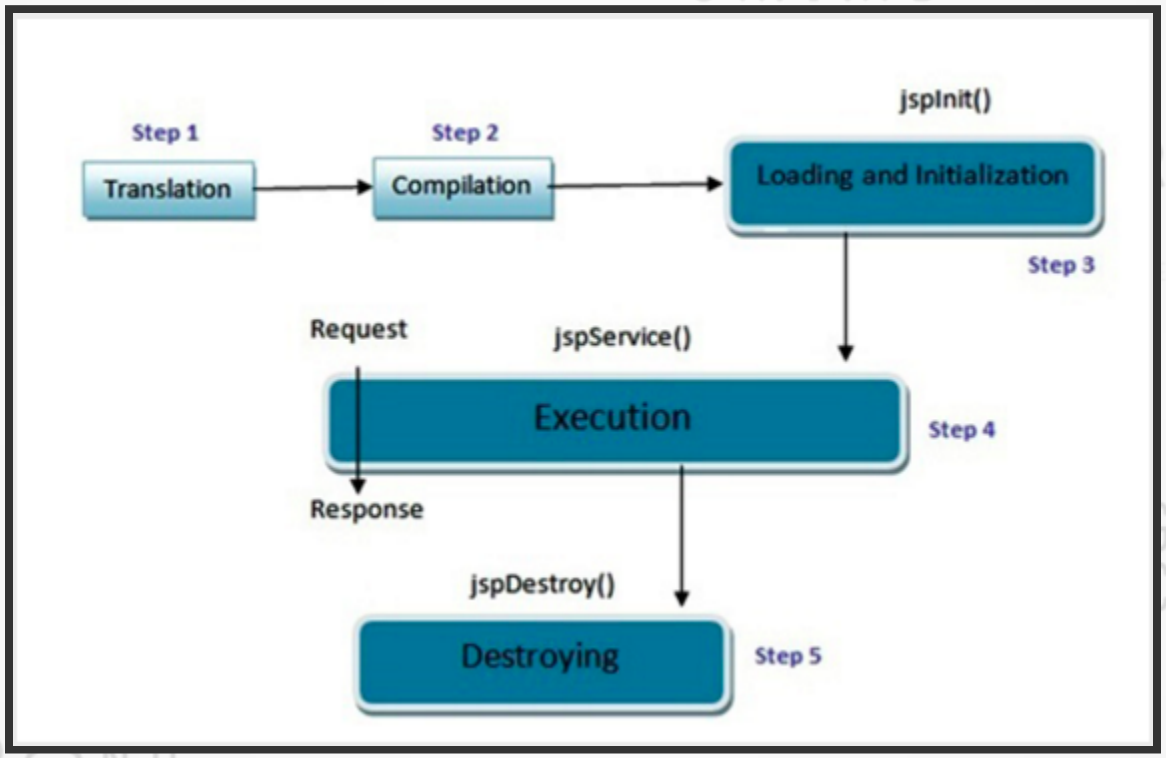


WHAT IS JSP?

JAVA SERVER PAGES

- JSP is a technology used to develop dynamic web pages.
- Java code is inserted into HTML using tags.
- JSPs are internally compiled into Java Servlets
- Servlet can be viewed as "HTML inside Java". JSP is "Java inside HTML“.

JSP PAGE LIFE CYCLE

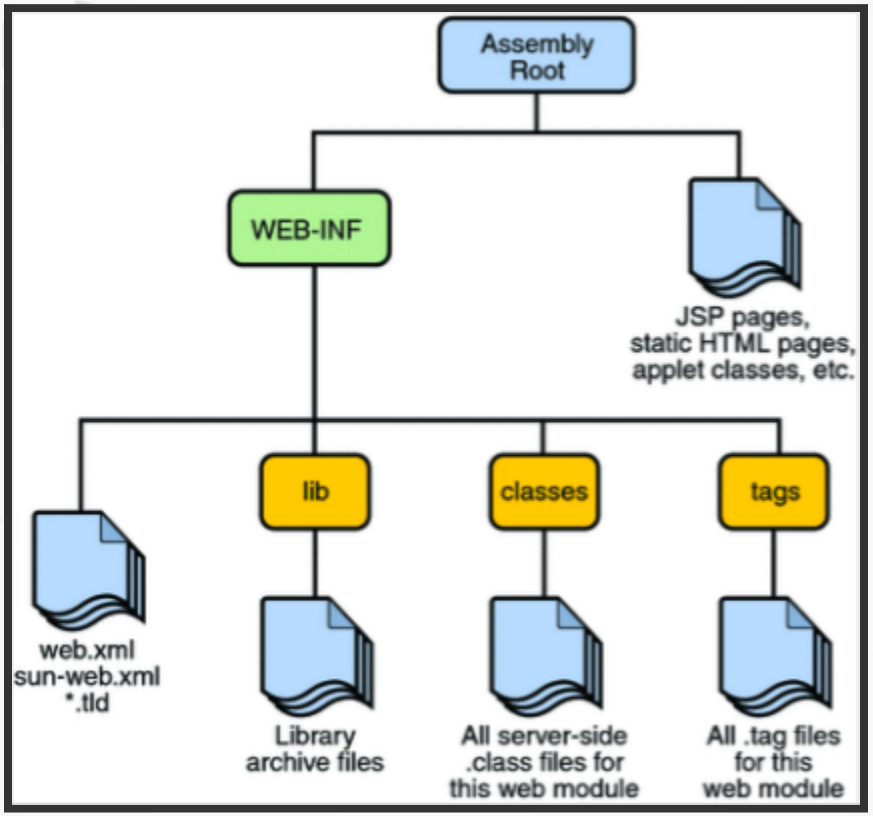


JSP HELLO WORLD!

HELLO WORLD JSP DEMO!

- Setup GlassFish web server.
- Setup Maven.
- Create a Hello World JSP page.

JSP PROJECT LAYOUT



JSP DIRECTIVES

JSP DIRECTIVE TYPES

- To set conditions that apply to the entire JSP.
- Format: <%@ directive attribute="value“ %>
- Types:
  - Page directives: apply to all page
  - Include directives: include other files
  - Taglib directives: add tag libraries

USING JSP TAGS

TYPES OF JSP TAGS

Tag	Name	Purpose
<% %>	JSP scriptlet	To insert a block of Java statements.
<%= %>	JSP expression	To display the string value of an expression.
<%-- --%>	JSP comment	To tell the JSP engine to ignore code.
<%! %>	declaration	To declare instance variables and methods.

USING JSP TAGS

JSP IMPLICIT OBJECTS

- *request* (instance of a javax.servlet.http.HttpServletRequest): To get the values of attributes or parameters that are passed to a JSP page.
- *response* (instance of a javax.servlet.http.HttpServletResponse): To send information to the client
- *out* (instance of a javax.servlet.jsp.JspWriter): send content in response
- Other examples: *session*, *application*.

USING JPS TAGS

COMMENTS

- When you code HTML comments, the comments are compiled and executed, but the browser doesn't display them.
- When you code *JSP comments*, the comments aren't compiled or executed.

EL AND JSTL

EL (EXPRESSION LANGUAGE)

EL (EXPRESSION LANGUAGE)

- The JSP Expression Language (EL) makes it easy to access attributes from a request object.
- Syntax: ${}
- Cannot be written inside scriptlets.

EL

EL (EXPRESSION LANGUAGE)

- Examples:
  - ${pageContext.request.queryString}
  - ${param["username"]}
  - < p>${header["user-agent"]}< /p>

More:
http://docs.oracle.com/javaee/1.4/tutorial/doc/JSPIntro7.

EL

EL IMPLICIT OBJECTS

- param: get request parameter
- paramValues: get all request parameters in an array
- header: get a header value.
- headerValues : get request header information
- cookie: get cookie value
- sessionScope: get attribute value in a session

ADVANTAGES OF EL

- El allows you to access nested properties.
- EL does a better job of handling null values.
- Write less code

JSTL

JSTL

- The *JSP Standard Tag Library (JSTL)* provides tags for common JSP tasks.
- Code a *taglib* directive that identifies the JSTL library and its prefix before using JSTL tags within a JSP.
- JSTL variable scope is not the same as the scriplet's scope.

JSTL

JSTL

- Before you can use JSTL tags within an application, you must make the required jars files available to the application.
- Before you can use JSTL tags within a JSP, you must code a taglib directive that identifies the JSTL library and its prefix.

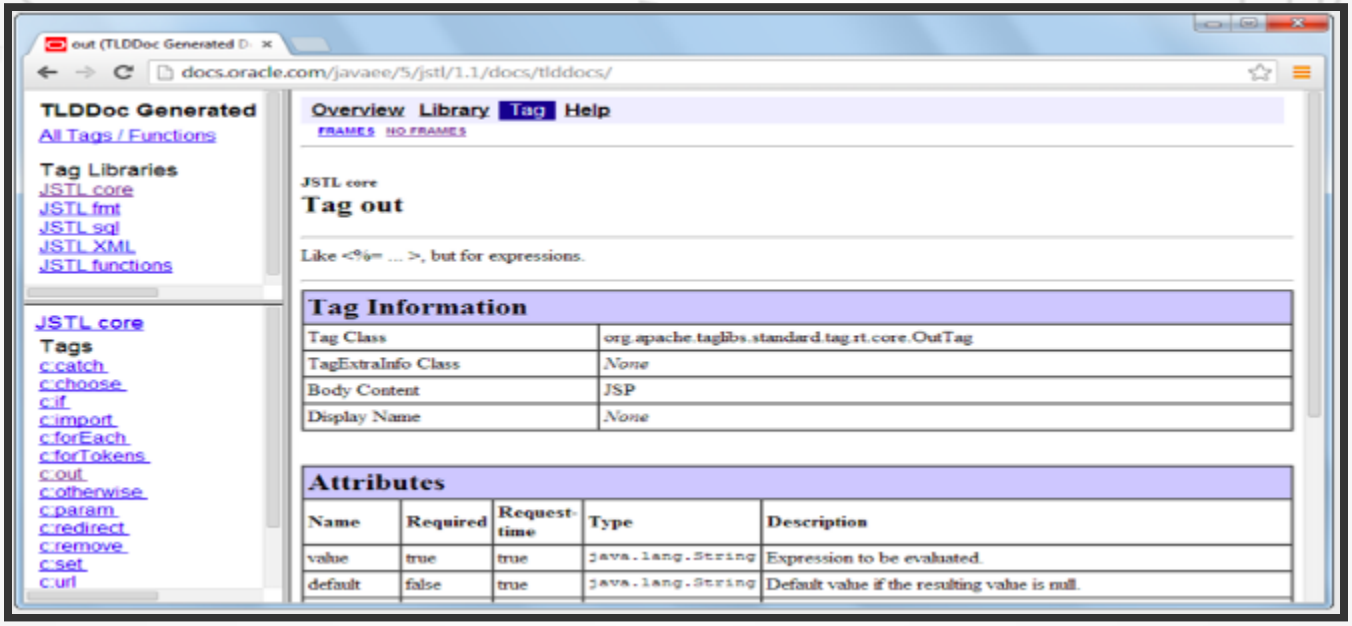
JSTL

THE JSTL LIBRARIES

Using the MVC pattern, tags in the core library will often be all that's necessary.

- *Core* – contains the core tags for common tasks such as looping and if/else statements
- *Formatting* – provides tags for formatting numbers, times, and dates so they work correctly with internalization.
- *SQL* – working with SQL queries and data sources
- *XML* – manipulating XML documents
- *Functions* – can be used to manipulate strings

INTRO TO JSTL



Viewing the documentation

CORE LIBRARY

FOREACH

- The *forEach* tag loops through most types of collections (i.e. arrays).
- The *var* attribute specifies the variable name that accesses each item within the collection.
- The *items* attribute specifies the collection that stores the data.
- *forEach* tags may be nested if necessary.

CORE LIBRARY

```
1   <c:forEach var="item" items="${cart.items}">
2   <tr>
3       <td>${item.quantity}</td>
4       <td>${item.product.description}</td>
5       <td>${item.product.priceCurrencyFormat}</td>
6       <td>${item.totalCurrencyFormat}</td>
7   </tr>
8   </c:forEach>
```

CORE LIBRARY

FORTOKENS

- The *forTokens* tag to loops through delimited values stored in a string
- The *var* attribute specifies the variable name that is used to access each delimited string and the *items* attribute specifies the string that stores the data.
- *forTokens* tags may be nested if necessary.

CORE LIBRARY

```
1   <p>Product codes</p>
2   <ul>
3   <c:forTokens var="productCode" items="${productCodes}" delims="," >
4       <li>${productCode}</li>
5   </c:forTokens>
6   </ul>
```

CORE LIBRARY

MORE ATTRIBUTES FOR LOOPING

The attributes below work for both the forEach and forTokens tags.

- *begin* – specifies the first index for the loop
- *end* – specifies the last index for the loop
- *step* – specifies the amount to increment the index each time through the loop

CORE LIBRARY

IF

The *if* tag performs conditional processing that's similar to an if statement. Use the *test* attribute to specify the Boolean condition and, if necessary, nest one if tag inside of another.

CORE LIBRARY

```
<c:if test="${cart.count == 1}">
    <p>You have 1 item in your cart.</p>
</c:if>
<c:if test="${cart.count > 1}">
    <p>You have ${cart.count} items in your cart.</p>
</c:if>
```

CORE LIBRARY

CHOOSE

The *choose* tag performs conditional processing that's similar to an if/else statement. Use the test attribute to specify the Boolean condition and, if necessary, nest one choose tag inside of another.

INCLUDE FILES IN JSP

FILE INCLUDES

- Use the include directive to include a file in a JSP at compile-time.
- When you use the include directive, code in the included file becomes part of the generated servlet. As a result, any changes to the included file won't appear in the JSP until the JSP is recompiled.
- When you include a file at runtime, any changes to the included file appear in the JSP the next time it is requested.

INCLUDE FILES IN JSP

FILE INCLUDES

Use the *include directive* to include a file in a JSP at *compile-time*. When you use the include directive, code in the included file becomes part of the generated servlet. As a result, any changes to the included file won't appear in the JSP until the JSP is recompiled. When you include a file at runtime, any changes to the included file appear in the JSP the next time it is requested.

INCLUDE FILES IN JSP

FILE INCLUDES EXAMPLE

<%@ include file="header.jsp" %> < center>

< p>Thanks for visiting my page.< /p>

< /center> <%@ include file="footer.jsp" %>

COMMON JSP ERRORS

TIPS FOR FIXING JSP ERRORS

- Make sure that the URL is valid and that it points to the right location for the requested page.
- Make sure all of the HTML, JSP, and Java class files are in the correct locations.
- Read the error page carefully to get all available information about the error.

RECAP

WHAT YOU SHOULD KNOW:

- What is JSP?
- JSP introduction?
- What is EL and JSTL?
- Using JSP tags?
- Know common JSP errors?
- Know some JSTL libraries

MVC

MVC

TOPICS

- MVC Pattern

MVC PATTERN

PATTERNS FOR SERVLET/JSP APPLICATIONS

TWO PATTERNS

A *pattern* is a standard approach that programmers use to solve common programming problems.

MODEL 1 PATTERN

The *model 1 pattern* uses a JSP to handle the request and response of the application, as well as all of the processing for the application.

PATTERNS FOR SERVLET/JSP APPLICATIONS

MODEL 1 PATTERN AND CLASSES

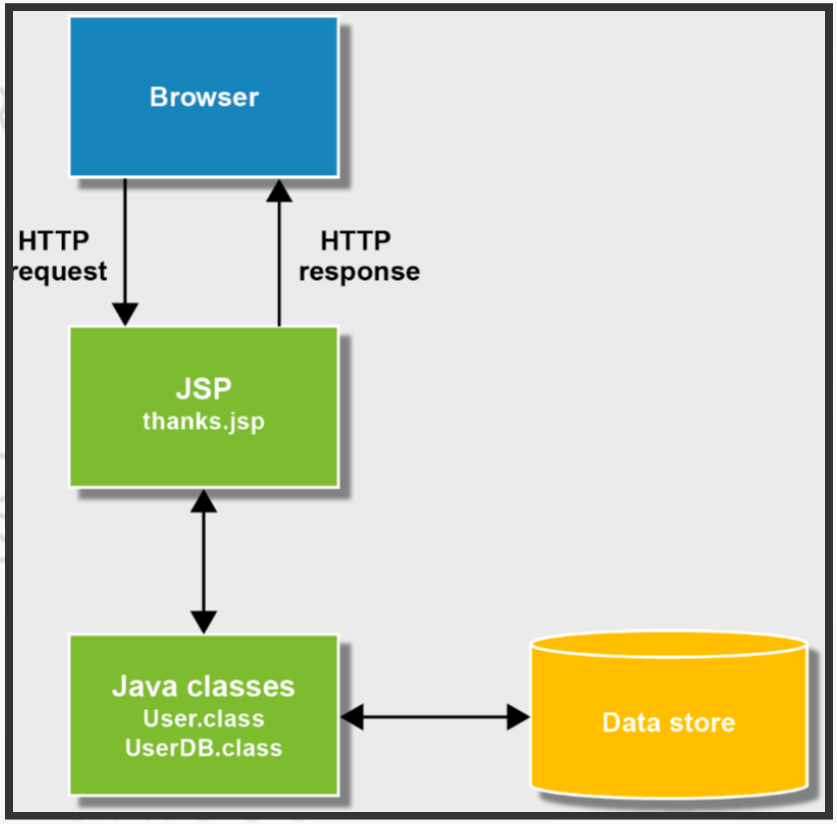
The JSPs can use regular Java classes to store the data of the application and to do the business processing of the application. In addition, they can use the *data classes* to work with the data store.

PATTERNS FOR SERVLET/JSP APPLICATIONS

DATA STORE

The *data store* is typically a database, but it can include disk files. This is often referred to as *persistent data storage* because it exists after the application ends.

PATTERNS FOR SERVLET/JSP APPLICATIONS



PATTERNS FOR SERVLET/JSP APPLICATIONS

MODEL 2 (MVC) PATTERN

The *Model 2 pattern* separates the code into a model, a view, and a controller; it's known as the *Model-View-Controller (MVC) pattern*. MVC is common for web applications with significant processing requirements.

PATTERNS FOR SERVLET/JSP APPLICATIONS

MVC BREAKDOWN

In the MVC pattern, the *model* consists of business objects like the User object, the *view* consists of HTML pages and JSPs, and the *controller* consists of servlets.

PATTERNS FOR SERVLET/JSP APPLICATIONS

DATA ACCESS CLASSES

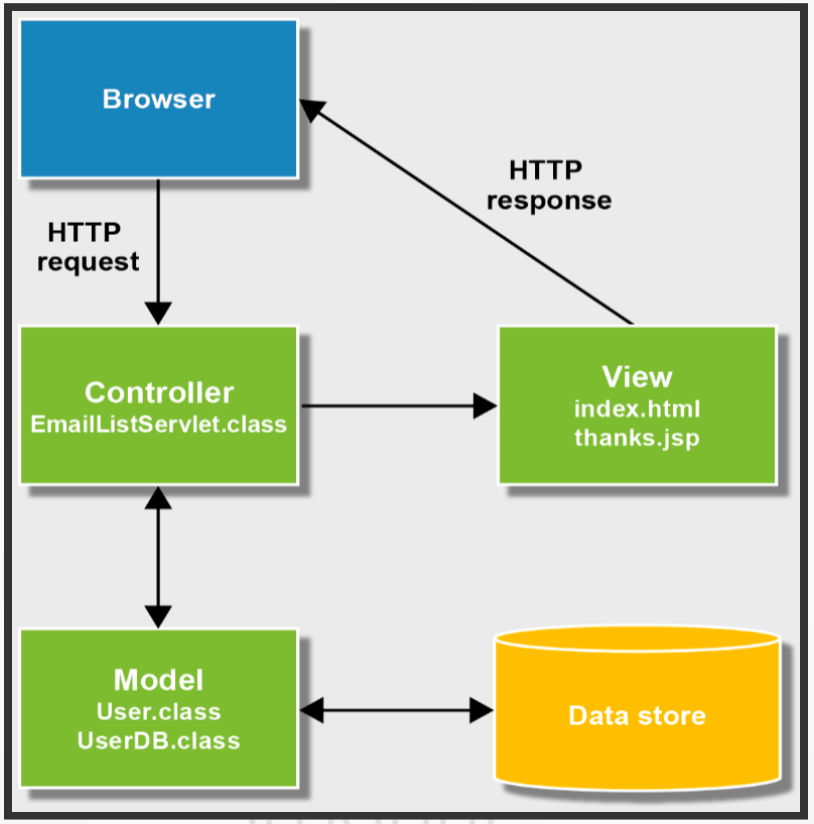
Usually, the methods of *data access classes* like the UserDB class are used to read and write business objects like the User object to and from the data store.

PATTERNS FOR SERVLET/JSP APPLICATIONS

INDEPENDENT LAYERS

When you use the MVC pattern, you try to construct each layer so it's as independent as possible. Then, if you need to make changes to one layer, any changes to the other layers are minimized.

PATTERNS FOR SERVLET/JSP APPLICATIONS



PATTERNS FOR SERVLET/JSP APPLICATIONS

A SERVLET/JSP APPLICATION USING MVC

The MVC pattern will be explained by showing a sample application which includes each component and the relevant code.

PATTERNS FOR SERVLET/JSP APPLICATIONS

THE USER INTERFACE

The *user interface* for this web application is a two page web application that allows the user to join an email list.

PATTERNS FOR SERVLET/JSP APPLICATIONS

HTML PAGE

The first page of this application is a static HTML page that allows the user to enter his or her email address, first name, and last name. When the user clicks the "Join Now" button an HTTP request is sent to the server.

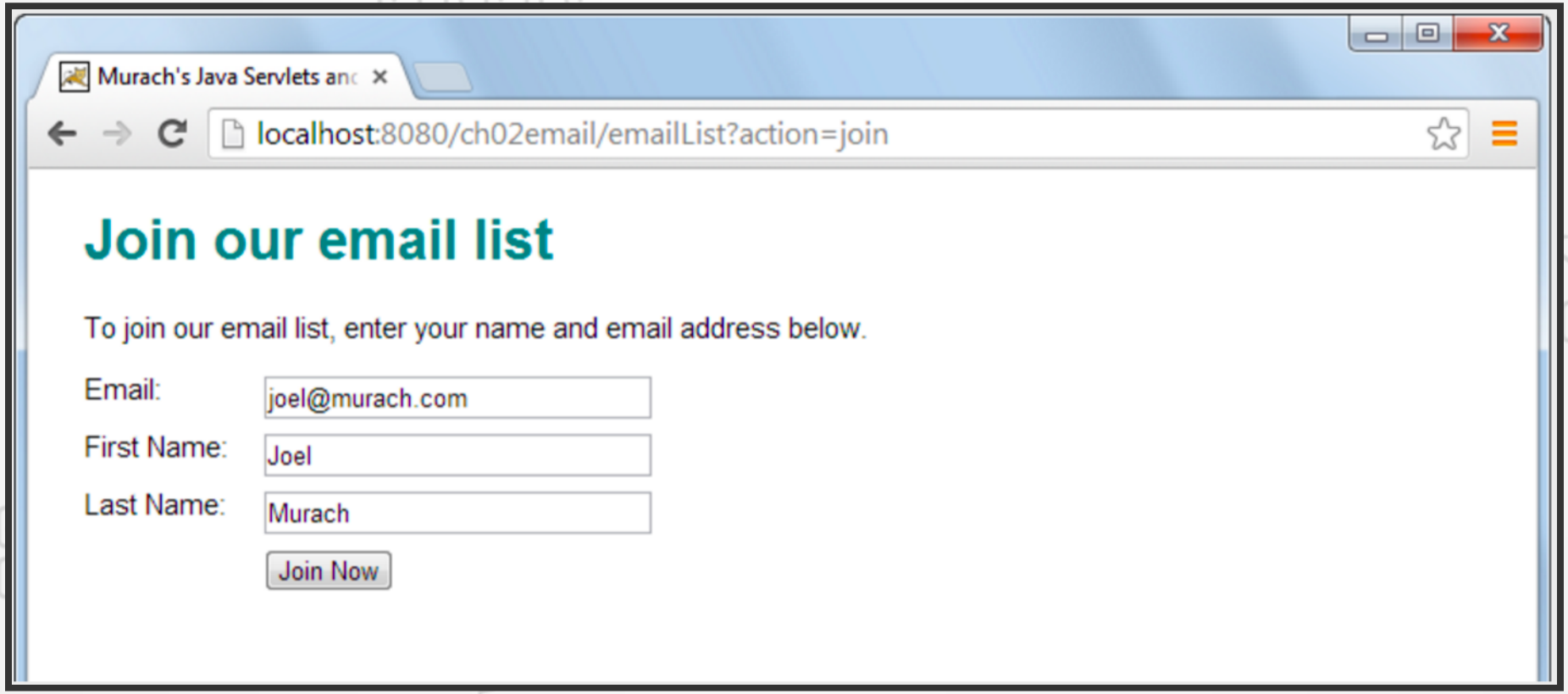
PATTERNS FOR SERVLET/JSP APPLICATIONS

JSP PAGE

The HTTP response is a dynamic HTML page or JSP that includes the includes the data the user entered on the first page.

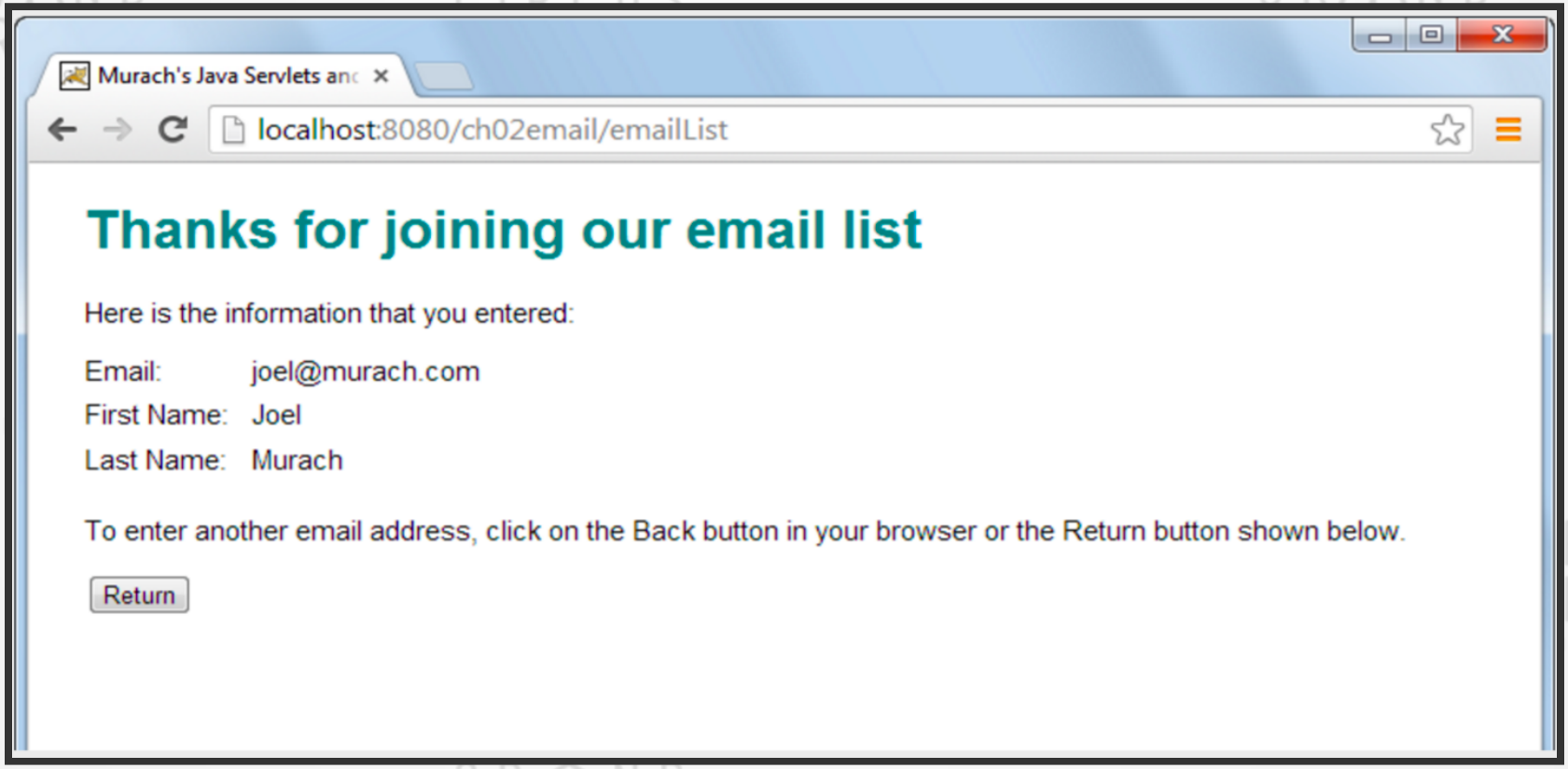
PATTERNS FOR SERVLET/JSP APPLICATIONS

THE HTML PAGE THAT GETS DATA FROM THE USER



PATTERNS FOR SERVLET/JSP APPLICATIONS

THE JSP THAT DISPLAYS THE DATA



SERVLET/JPS APPLICATION USING MVC

HTML FOR THE FIRST PAGE

An HTML file contains the tags that define the content of the web page.

The action and method attributes of the form tag set up a request for the URL that's executed when the user clicks the submit button.

SERVLET/JPS APPLICATION USING MVC

HTML FOR THE FIRST PAGE

The three text boxes represent parameters that are passed to the servlet when the user clicks the submit button

The parameter names are firstName, lastName, and emailAddress, and the parameter values are strings that the user enters into the text boxes.

SERVLET/JPS APPLICATION USING MVC

HTML FOR THE FIRST PAGE

```
< !DOCTYPE html>
< html>
< head>
    < meta charset="utf-8">
    < title>Murach's Java Servlets and JSP< /title>
    < link rel="stylesheet" href="styles/main.css" type="text/  css"/>
< /head>
< body>
    < h1>Join our email list< /h1>
    < p>To join our email list, enter your name and
       email address below.< /p>
```

SERVLET/JPS APPLICATION USING MVC

HTML FOR THE FIRST PAGE (CONT.)

```
< form action="emailList" method="post">
        < input type="hidden" name="action" value="add”>
        < label>Email:<!-- label-->
        < input type="email" name="email" required>< br>
        < label>First Name:< /label>
        < input type="text" name="firstName" required>< br>
        < label>Last Name:< /label>
        < input type="text" name="lastName" required>< br>
        < label> < /label>
        < input type="submit" value="Join Now" id="submit">
    < /form>
< /body>
< /html>
```

SERVLET/JPS APPLICATION USING MVC

THE MAIN.CSS FILE

A CSS (Cascading Style Sheet) file contains the formatting for the web pages.

SERVLET/JPS APPLICATION USING MVC

THE MAIN.CSS FILE

```
body {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 11pt;
    margin-left: 2em;
    margin-right: 2em;
}
h1 {
    color: teal;
}
label {
    float: left;
    width: 6em;
    margin-bottom: 0.5em;
}
```

SERVLET/JPS APPLICATION USING MVC

THE MAIN.CSS FILE

```
input[type="text"], input[type="email"] {
    width: 15em;
    margin-left: 0.5em;
    margin-bottom: 0.5em;
}
br {
    clear: both;
}
#submit {
    margin-left: 0.5em;
}
```

SERVLET/JPS APPLICATION USING MVC

SERVLETS

Servlets contain Java code for a web application. When a servlet controls the flow of the application, it's known as a controller.

SERVLET/JPS APPLICATION USING MVC

THE EMAILLIST SERVLET CLASS

```
package murach.email;

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

import murach.business.User;
import murach.data.UserDB;

var myName = name;
public class EmailListServlet extends HttpServlet  {
```

SERVLET/JPS APPLICATION USING MVC

THE EMAILLIST SERVLET CLASS(CONT.)

```
@Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
                          throws ServletException, IOException {

        String url = "/index.html";

        // get current action
        String action = request.getParameter("action");
        if (action == null) {
            action = "join";  // default action
        }
```

SERVLET/JPS APPLICATION USING MVC

THE EMAILLIST SERVLET CLASS(CONT.)

```
// perform action and set URL to appropriate page
        if (action.equals("join")) {
            url = "/index.html";    // the "join" page
        }
        else if (action.equals("add")) {
            // get parameters from the request
            String firstName = request.getParameter("firstName");
            String lastName = request.getParameter("lastName");
            String email = request.getParameter("email");

            // store data in User object and save User object in db
            User user = new User(firstName, lastName, email);
            UserDB.insert(user);
```

SERVLET/JPS APPLICATION USING MVC

THE EMAILLIST SERVLET CLASS(CONT.)

```
// set User object in request object and set URL
            request.setAttribute("user", user);
            url = "/thanks.jsp";   // the "thanks" page
        }

        // forward request and response objects to specified URL
        getServletContext()
            .getRequestDispatcher(url)
            .forward(request, response);
    }
```

SERVLET/JPS APPLICATION USING MVC

THE EMAILLIST SERVLET CLASS(CONT.)

```
@Override
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
                         throws ServletException; IOException {
        doPost(request, response);
    }
}
```

SERVLET/JPS APPLICATION USING MVC

THE WEB.XML FILE

The web.xml file, or *deployment descriptor (DD)*, describes how the web application should be configured when it's deployed.

SERVLET/JPS APPLICATION USING MVC

THE WEB.XML FILE

```
< ?xml version="1.0" encoding="UTF-8"?>
< web-app version="3.1"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">

    < servlet>
        < servlet-name>EmailListServlet< /servlet-name>
        < servlet-class>murach.email.EmailListServlet< /servlet-class>
    < /servlet>
    < servlet-mapping>
        < servlet-name>EmailListServlet< /servlet-name>
```

SERVLET/JPS APPLICATION USING MVC

THE WEB.XML FILE

```
< url-pattern>/emailList< /url-pattern>
    < /servlet-mapping>

    < session-config>
        < session-timeout>30< /session-timeout>
    < /session-config>
< welcome-file-list>
        < welcome-file>index.html< /welcome-file>
        < welcome-file>index.jsp< /welcome-file>
    < /welcome-file-list>

< /web-app>
```

SERVLET/JPS APPLICATION USING MVC

JAVABEAN

A *JavaBean*, or *bean*, is a Java class that (1) provides a zero-argument constructor, (2) provides get and set methods for all of its instance variables, and (3) implements the Serializable or Externalizable interface.

SERVLET/JPS APPLICATION USING MVC

THE USER CLASS

```
package murach.business;
import java.io.Serializable;
public class User implements Serializable {
    private String firstName;
    private String lastName;
    private String email;

    public User() {
        firstName = "";
        lastName = "";
        email = "";
    }
```

SERVLET/JPS APPLICATION USING MVC

THE USER CLASS

```
public User(String firstName, String lastName, String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }
public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
```

SERVLET/JPS APPLICATION USING MVC

THE USER CLASS (CONT.)

```
public String getLastName() {
        return lastName;
    }
 public void setLastName(String lastName) {
        this.lastName = lastName;
    }
 public String getEmail() {
        return email;
    }
 public void setEmail(String email) {
        this.email = email;
    }
}
```

SERVLET/JPS APPLICATION USING MVC

THE THANKS.JSP FILE

A *JavaServer Page (JSP)* consists of special Java tags such as Expression Language (EL) tags that are embedded within HTML code. An EL tag begins with a dollar sign ($).

SERVLET/JPS APPLICATION USING MVC

THE THANKS.JSP FILE

```
< !doctype html>
< html>
< head>
    < meta charset="utf-8">
    < title>Murach's Java Servlets and JSP< /title>
    < link rel="stylesheet" href="styles/main.css" type="text/css"/>
< /head>
 < body>
    < h1>Thanks for joining our email list< /h1>
    < p>Here is the information that you entered:< /p>
    < label>Email:< /label>
    < span>${user.email}< /span>< br>
    < label>First Name:< /label>
```

SERVLET/JPS APPLICATION USING MVC

THE THANKS.JSP FILE (CONT.)

```
< span>${user.firstName}< /span>< br>
    < label>Last Name:< /label>
    < span>${user.lastName}< /span>< br>
    < p>To enter another email address, click on the Back
    button in your browser or the Return button shown
    below.< /p>
    < form action="" method="get">
        < input type="hidden" name="action" value="join">
        < input type="submit" value="Return">
    < /form>
 < /body>
< /html>
```

RECAP

RECAP

WHAT YOU SHOULD KNOW AT THIS POINT

- What are the patterns for JSP/Servlet applications
- What is MVC?
- Programming JSP/Servlet applications using MVC

SESSIONS AND COOKIES

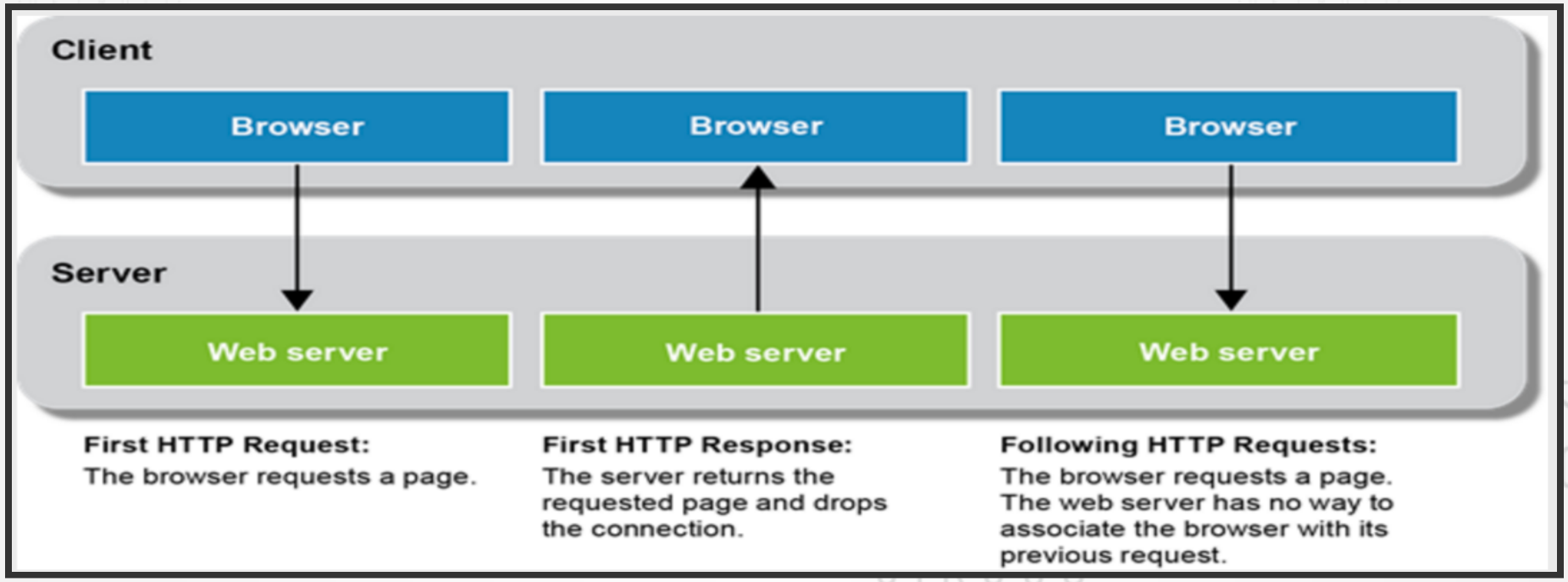
SESSIONS AND COOKIES

TOPICS:

- Introduction to session tracking
- Working with sessions
- Working with cookies
- Working with Application

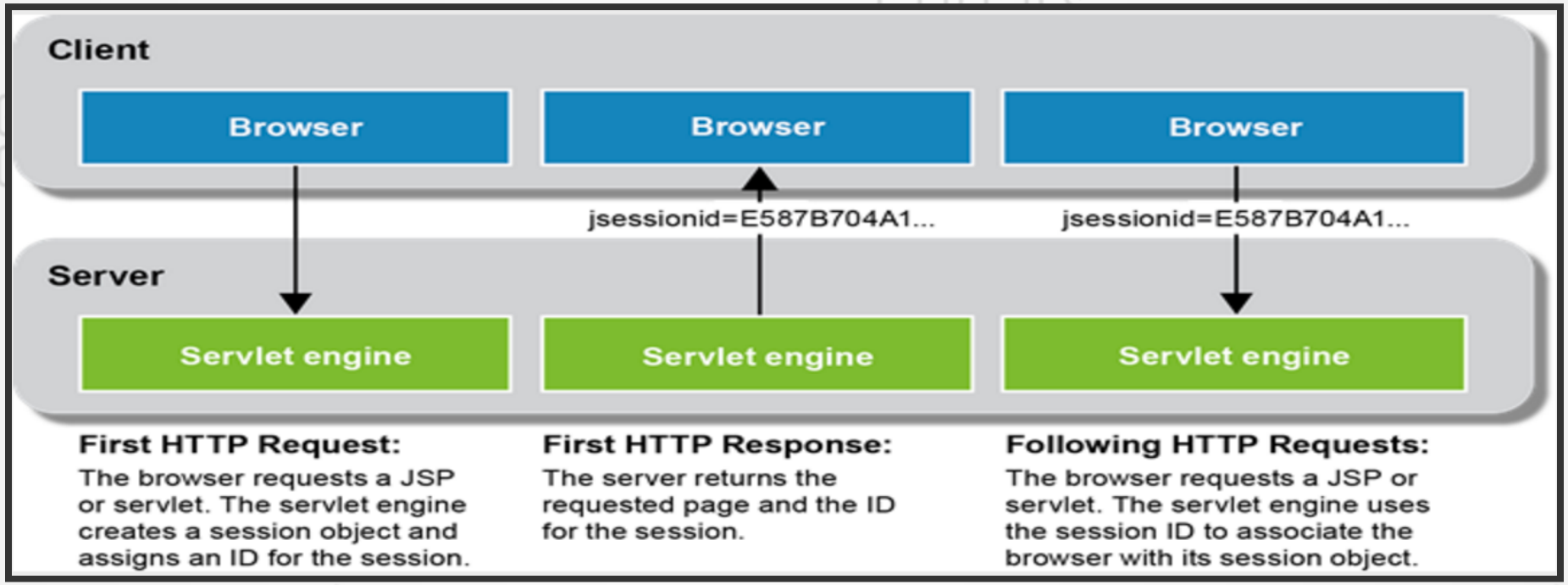
INTRODUCTION TO SESSION TRACKING

SESSION TRACKING



Why Session Tracking is Difficult with HTTP

SESSION TRACKING



How Java Keeps Track of Sessions

SESSION TRACKING

SESSION TRACKING AND COOKIES

- HTTP is a *stateless protocol*. Once a browser makes a request, it drops the connection to the server. To maintain *state*, a web application must use *session tracking*.
- By default, the servlet API uses a cookie to store a session ID in each browser. The browser passes the cookie to the server with each request.

SESSION TRACKING

SESSION TRACKING AND COOKIES (CONT'D)

- To provide session tracking when cookies are disabled in the browser, use *URL encoding* to store the session ID in the URL for each page of an application, but not considered a best practice.
- Setting is in web.xml in the Apache conf folder
- Example :

```
< session-config>
        < session-timeout>30< /session-timeout>
    < tracking-mode>URL< /tracking-mode>
  < /session-config>
```

WORKING WITH SESSIONS

SESSIONS

SESSIONS

- *Persistent cookies* are stored on the user's PC. Per-session cookies are deleted when the session ends.
- A session object is created when a browser makes the first request and destroyed when the session ends.
- A session ends when a specified amount of time elapses without another request or when the user exits the browser.

SESSIONS

PROVIDING THREAD-SAFE ACCESS TO THE SESSION

- Each servlet creates one session object that exists for multiple requests from a single client.
- If the client has one browser window open, access to the session object is thread-safe.
- If the client has multiple browser windows open, two threads from the same client could access the session object at the same time. The session object is NOT thread-safe.

SESSIONS

APPLICATION IN JSP

- Application is used to share values across the entire JSP application.
- Example: Hit Count!

COOKIES

COOKIES

HOW COOKIES WORK

- A *cookie* is a name/value pair stored in a browser.
- A web application on the server creates a cookie and sends it to the browser. The browser on the client saves the cookie and returns it to the server every time it accesses a page from that server.
- Cookies can be set to persist for up to 3 years.

TYPICAL USES FOR COOKIES

ALLOW USERS TO SKIP LOGIN/REGISTRATION

i.e. when sites gather data such as username, password, address, credit card number, etc.

CUSTOMIZE PAGES

Users can choose to display weather, sports scores, stock quotations, etc.

TYPICAL USES FOR COOKIES

TARGETED ADVERTISING

Ads and banners can target the user's interests

COOKIES

- A per-session cookie that holds the session ID is automatically created each session. It is used to relate the browser to the session object.
- You can also create and send other cookies to a user's browser, and use them to access user-specific data stored in a file or database.

COOKIES

```
1   Cookie c = new Cookie("userIdCookie", userId);
2   c.setMaxAge(60*60*24*365*2);    // set age to 2 years
3   c.setPath("/");                 // allow access by entire app
4   response.addCookie(c);
```

Create a cookie & set in a response

FORMATTING PHOTO WITH CAPTION

```
1   Cookie[] cookies = request.getCookies();
2   String cookieName = "userIdCookie";
3   String cookieValue = "";
4   for (Cookie cookie: cookies) {
5     if (cookieName.equals(cookie.getName()))
6           cookieValue = cookie.getValue();
```

Get a cookie value from a request

COOKIES

DELETING

To delete a persistent cookie from a browser, set the age of a cookie to 0.

COOKIES

FOUR METHODS OF THE COOKIE CLASS

All of the following set methods have corresponding get methods.

setPath(String path)

setDomain(String domainPattern)

setSecure(boolean flag)

WORKING WITH HIDDEN FIELDS

HIDDEN FIELDS

WHAT ARE HIDDEN FIELDS

We can use hidden fields to pass parameters to a servlet or JSP. To do that, we code hidden fields within a form tag.

HIDDEN FIELDS

```
1   <form action="cart" method="post">
2     <input type="submit" value="Add To Cart">
3     <input type="hidden" name="productCode" value="8601">
4   </form>
```

An example of hidden fields

HIDDEN FIELDS

LIMITATIONS

Because hidden fields are displayed in the source code for the page returned to the browser, anyone can view the parameters. As a result, hidden fields aren't appropriate for sensitive data like passwords.

RECAP

RECAP

WHAT YOU SHOULD KNOW AT THIS POINT:

- What is state management?
- What are sessions?
- How sessions work
- Session tracking
- What are cookies?
- How cookies work
- How to use sessions and cookies
- URL rewriting
- Hidden fields (uses and limitations)
- How application works

SPRING FRAMEWORK

SPRING FRAMEWORK

TOPICS

- Java Frameworks
- Spring

JAVA FRAMEWORKS

JAVA FRAMEWORKS

FRAMEWORKS

- Frameworks are a group of tools and models that are built to help programmers to build *complex and extensible applications*.
- Programmers use frameworks to *reduce* the development time and make the programming process *easier*.

JAVA FRAMEWORKS

MOST POPULAR JAVA FRAMEWORKS

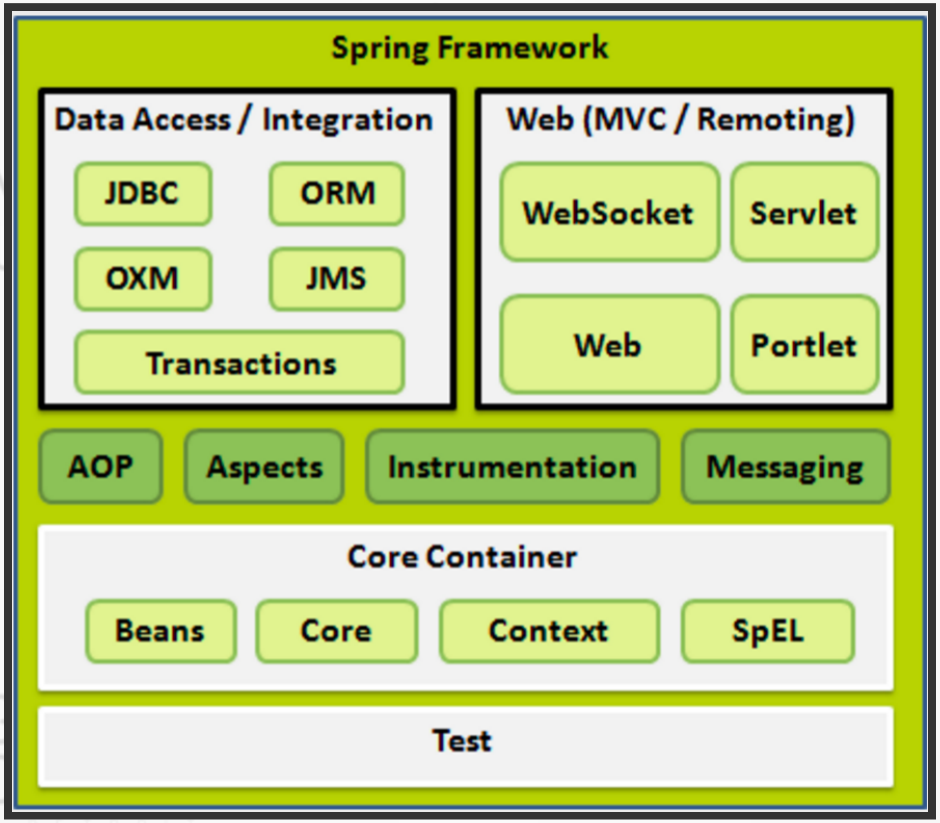
- Spring
- JSF
- Struts

JAVA FRAMEWORKS - SPRING

SPRING

- Spring is the most used Java framework!
- Large group of libraries to program enterprise applications.
- Has projects for Data, Security, Big Data, Mobile, Web services, and more.
- https://spring.io/

JAVA FRAMEWORKS - SPRING



SPRING

SPRING MVC

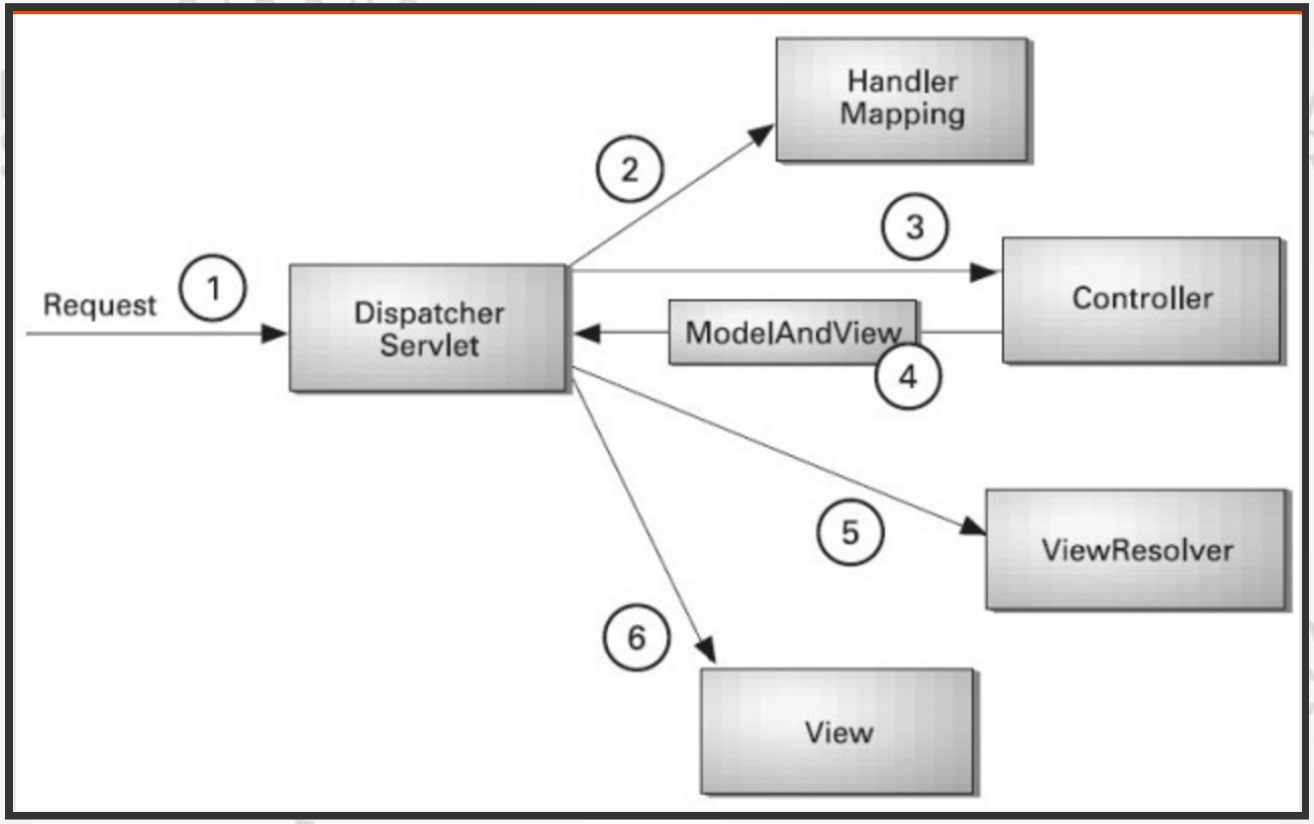
- Spring has an MVC framework that can be used for more easily creating web applications.
- Model – View – Controller Implementation using annotations and configuration files.

SPRING MVC

SPRING MVC

The Spring Web MVC framework is designed around a *DispatcherServlet* that dispatches requests to handlers, with configurable handler mappings.

SPRING MVC


Handler Mapping
2
3
Request
1
Dispatcher Servlet
ModelAndView
Controller
4
5
ViewResolver
6
View

SPRING MVC

HANDLING REQUESTS

1. Incoming request is handled by the DispatcherServlet (front controller).
2. The DispatcherServlet looks at the web.xml to find the handler mapping.
3. The DispatcherServlet forwards the request to the controller.

SPRING MVC

HANDLING REQUESTS (CONT.)

1. The controller processes the request, and then returns the Model and the View to the DispatcherServlet.
2. The DispatcherServlet then checks the web.xml again at the view resolver section to know the specified view component.

SPRING MVC

WEB.XML

- Web.xml will have settings to configure the dispatcher!
- Web.xml can be used to set the default page of the web application.
- Also, we can use it to do the mappings of the client requests.

SPRING MVC

WEB.XML

```
< welcome-file-list>
  < welcome-file>index.jsp< /welcome-file>
< /welcome-file-list>
  < servlet>
    < servlet-name>dispatch< /servlet-name>
    < servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    < /servlet-class>
    < load-on-startup>1< /load-on-startup>
  < /servlet>
< servlet-mapping>
  < servlet-name>dispatch< /servlet-name>
    < url-pattern>*.html< /url-pattern>
< /servlet-mapping>
< /web-app>
```

SPRING MVC

DISPATCHER

```
< context:component-scan base-package="com.testSpring2.controller" />
< context:component-scan base-package="com.testSpring2.controller2" />

< bean id="viewResolver"
class="org.springframework.web.servlet.view.UrlBasedViewResolver">
< property name="viewClass"
value="org.springframework.web.servlet.view.JstlView" />
< property name="prefix" value="/WEB-INF/jsp/" />
< property name="suffix" value=".jsp" />
< /bean>
```

SPRING MVC

CONTROLLER

```
@Controller
public class testSpring2HelloWorld {

  @RequestMapping("/welcome")
  public ModelAndView helloWorld() {

  String message = "Hello World!";
  return new ModelAndView("welcome", "message", message);
  }
}
```

SPRING MVC

CONTROLLERS

```
@RequestMapping(value = "/index", method = RequestMethod.GET)
```

WHAT IS MAVEN?

APACHE MAVEN

- Maven is a software project management tool that is used to manage the project's build, reporting, documentation, and testing.
- https://maven.apache.org/
- POM (Project Object Model) is the file that Maven uses to load the required dependencies.
- Similar software: Gradle, Ant

SPRING MVC

SPRING MVC

- Install Spring (using Maven)
- Example with Instructor.

POM.XML

DEPENDENCIES:

```
< dependencies>
  < dependency>
    < groupId>org.springframework< /groupId>
    < artifactId>spring-context< /artifactId>
    < version>4.2.4.RELEASE< /version>
  < /dependency>
< /dependencies>
```

< groupId>:< artifactId>:< version> is the full name for the dependency

HIBERNATE

HIBERNATE

TOPICS

- Hibernate

HIBERNATE

HIBERNATE

OBJECT RELATIONAL MAPPING (ORM)

- Working with object-oriented systems along with a relational database causes a mismatch. Java represents data as interconnected objects while relational databases represent it in a tabular format.
- ORM is a technique for converting data between relational databases and object-oriented programming languages.

HIBERNATE

OBJECT RELATIONAL MAPPING (ORM)

- Java developers are not database developers! So using an ORM will reduce the need for developers to extensively know database programming.

HIBERNATE

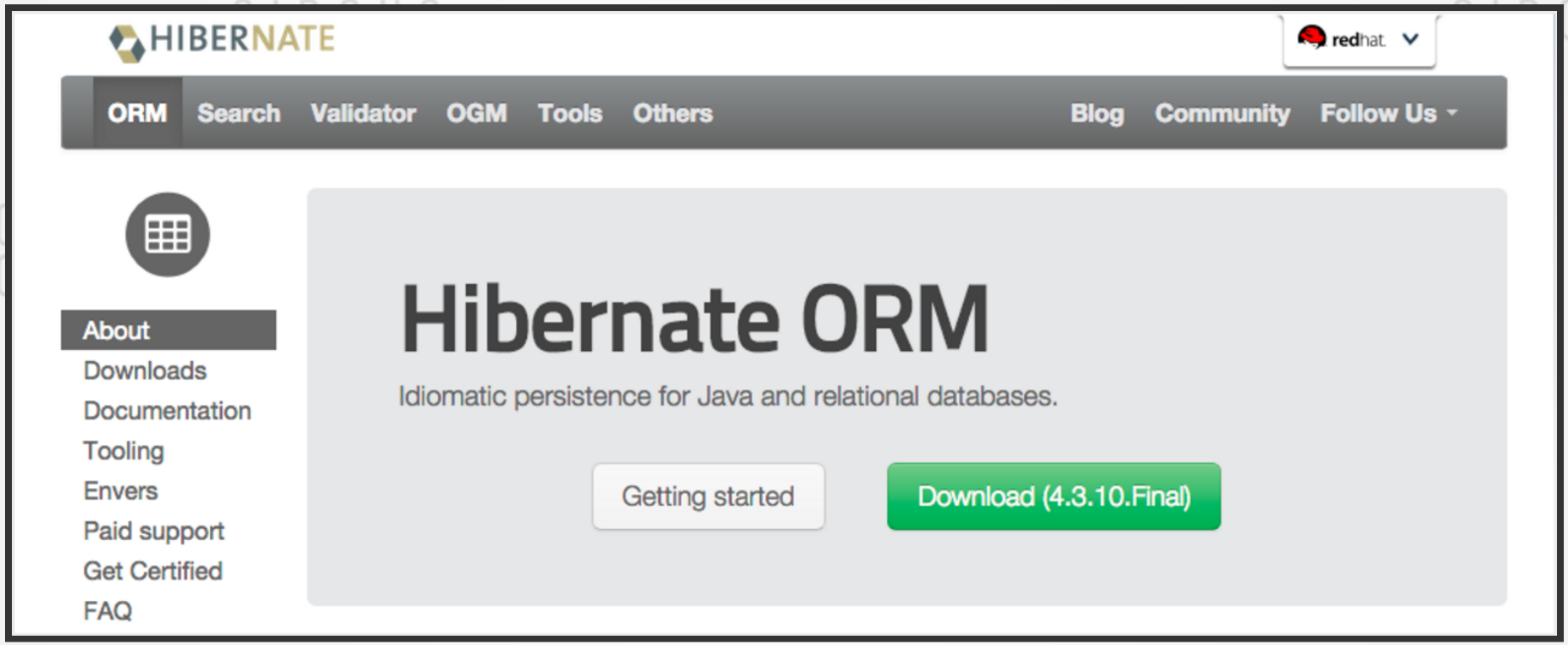
ADVANTAGES OF AN ORM SYSTEM

- Lets business code access objects rather than tables
- Hides details of SQL queries from OO logic
- Based on JDBC
- No need to deal with database implementation
- Quick development

HIBERNATE

Hibernate sits between Java objects and the database server

HIBERNATE



Download Hibernate at
hibernate.org/orm/downloads

AUTHENTICATION & AUTHORIZATION

AUTHENTICATION & AUTHORIZATION

TOPICS

- What are Credentials?
- What is Authentication?
- What is Authorization?
- How to secure a Web application.
- The purpose of HTTPS
- How to configure Tomcat for Basic and Form authentication

WHAT ARE CREDENTIALS

WHAT ARE CREDENTIALS?

CREDENTIALS

- Information used for identity verification1
- Usually consist of User name and Password

WHAT ARE CREDENTIALS?

CREDENTIAL CATEGORIES/FACTORS

- Knowledge factors (ex. Username, password or secret question)
- Possession factors (ex. Token, OpenID)
- Inherence factor (ex. Biometric data)

WHAT IS AUTHENTICATION?

WHAT IS AUTHENTICATION?

AUTHENTICATION

- Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be.
- Authentication is a process in which the credentials provided are compared to those on file in a database of authorized users' information on a local operating system or within an authentication server.

WHAT IS AUTHENTICATION?

AUTHENTICATION (CONT.)

- If the credentials match, the process is completed and the user is granted authorization for access.

WHAT IS AUTHORIZATION?

WHAT IS AUTHORIZATION?

AUTHORIZATION

- Once a user is Authenticated they can be granted access to secure resources by Authorization

ROLES

- Roles define memberships with permission to access some secure resource (admin, user, guest or premium member)

WHAT IS AUTHORIZATION?

USERS AND GROUPS

Users can have multiple roles and can be members of groups with specific roles

SECURING WEB APPLICATIONS

SECURING WEB APPLICATIONS

WEB AUTHENTICATION METHODS

- Basic
- Form
- Digest

BASIC AUTHENTICATION

DESCRIPTION

In Basic Authentication the Web server request that the client browser obtain the user credentials . The credentials are sent to the server for Authentication.

PRECAUTIONS

Basic Authentication is not secure. User credentials are encoded but not encrypted when sent to the server.

FORM AUTHENTICATION

DESCRIPTION

- The server sends a custom login form to the client. The completed form is posted to the server by the client and authentication by the server. If authentication fails the client redirects to a custom error page.
- Username and password form fields must be labeled j_username and j_password, respectively.
- Form action must be j_security_check (built into the servlet container).

FORM AUTHENTICATION

PRECAUTIONS

Form Authentication is not secure. User credentials are submitted as plain text to the server.

DIGEST AUTHENTICATION

DESCRIPTION

Digest Authentication is similar to Basic Http authentication except that password is encrypted before it is sent to the server3

THE PURPOSE OF HTTPS

HTTPS AND SSL

SSL

Secure Socket Layer – is an encrypted link between the client and server

HTTPS

When the HTTP protocol is sent over SSL is is called HTTPS

HTTPS AND SSL

APPLICATIONS

Form and Basic authentication become more secure with the HTTP protocol

OPEN ID

DESCRIPTION

- OpenID is based on Oauth 2.0 which is a federated authorization mechanism5
- Third party providers authenticate users and provide access tokens for authorization to secured resources
- Third party providers such as Amazon, Google, Facebook, etc... provide OpenID APIs in various programming languages for use in securing Web applications

HOW TO CONFIGURE TOMCAT FOR BASIC AND FORM AUTHENTICATION

TOMCAT CONTAINER MANAGED SECURITY

AUTHORIZATION USING TOMCAT

Tomcat can be configured to support Basic and Form authentication and authorization

TOMCAT SECURITY CONFIGURATIONS

- web.xml application deployment descriptor
- tomcat-users.xml
- Realms in the server.xml

TOMCAT REALMS CONFIGURATION

TOMCAT REALM

- Specifies the database type used to store username and password credentials
- Realms also contain the list of roles used in authorization

TOMCAT REALMS CONFIGURATION

TOMCAT REALMS

- JDBCRealm – Relational DB via JDBC driver
- DataSourceRealm* - Relational DB via JNDI JDBC DataSource
- JNDIRealm - LDAP based directory server, accessed via a JNDI provider

TOMCAT REALMS CONFIGURATION

TOMCAT REALMS

- UserDatabaseRealm - XML document (conf/tomcat-users.xml)
- MemoryRealm - stored in an in-memory object collection XML document (conf/tomcat-users.xml)
- JAASRealm - Java Authentication & Authorization Service (JAAS) framework

TOMCAT FORM AUTHENTICATION EXAMPLE

IMPLEMENTATION

1. Create login.jsp and error.jsp
2. Create Form element in login.jsp with text and password input fields
3. Specify secure resource with < security-constraint> element in application web.xml
4. Specify authentication method with < login-config> element in application web.xml
5. Specify roles with < security-role> element in application web.xml

RECAP

WHAT YOU SHOULD KNOW:

- The difference between Authentication and Authorization.
- Credential categories.
- Web application Authentication methods.
- How to implement Basic and Form authentication

RESTFUL WEB SERVICES

RESTFUL WEB SERVICES

TOPICS

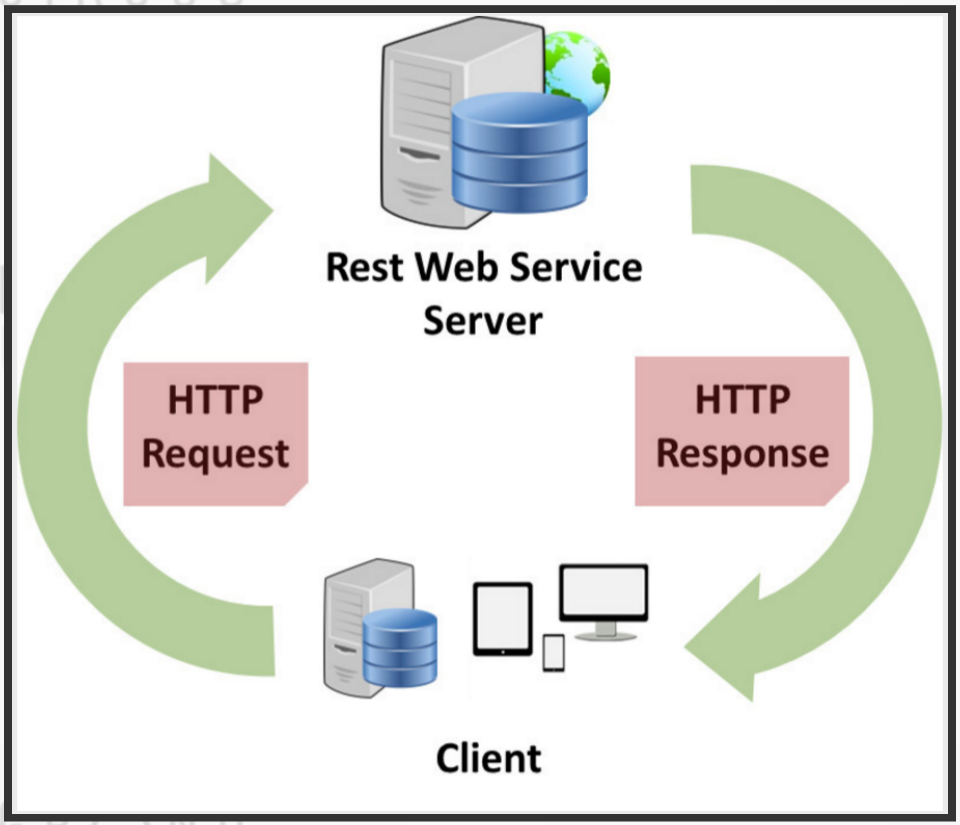
- RESTful web services

RESTFUL WEB SERVICES

RESTFUL WEB SERVICES

- REST stands for Representational State Transfer
- REST is a pattern that is used to develop web services and APIs
- REST web services use HTTP to communicate. Examples of used HTTP vocabulary:
  - Methods (GET, POST, etc.)
  - HTTP URI syntax (paths, parameters, etc.)
  - Media types (xml, Json, html, plain text, etc)
  - HTTP Response codes.

RESTFUL WEB SERVICES



RESTFUL HTTP REQUEST

- The client sends an HTTP request to start using the web service
- The client also needs to
  - Identify the location of a resource
  - Specify the HTTP verb that will be used to use the resource
  - Supply optional additional information to process the request

RESTFUL HTTP REQUEST EXAMPLE

Sample Client Requests: A typical client GET request:

```
GET /view?id=1 HTTP/1.1
User-Agent: Chrome
Accept: application/json
[CRLF]
```

RESTFUL HTTP REQUEST EXAMPLE

Sample Client Requests: A typical client POST request:

```
POST /save HTTP/1.1
User-Agent: IE
Content-Type: application/x-www-form-urlencoded
[CRLF]
name=x&id=2
```

RESTFUL HTTP RESPONSE

The HTTP response is sent from the server to the client

- Gives the status of the processed request.
- Supplies response headers (name-value pairs)
- Supplies an optional response body (html, xml, binary data, etc.)

RESTFUL HTTP RESPONSE SAMPLE

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1337
[CRLF]
<html>
  <!-- Some HTML Content. -->
</html>
```

RESTFUL HTTP RESPONSE SAMPLE

```
HTTP/1.1 500 Internal Server Error
```

```
HTTP/1.1 201 Created
Location: /view/7
[CRLF]
Some message goes here.
```

BUILD A RESTFUL WEB SERVICE

- Follow the instructor to build a simple Hello World web service
- Use: https://jersey.java.net/

RECAP

WHAT YOU SHOULD KNOW:

- What is a RESTful web service
- HTTP requests and responses
- How to build and use a RESTful web service