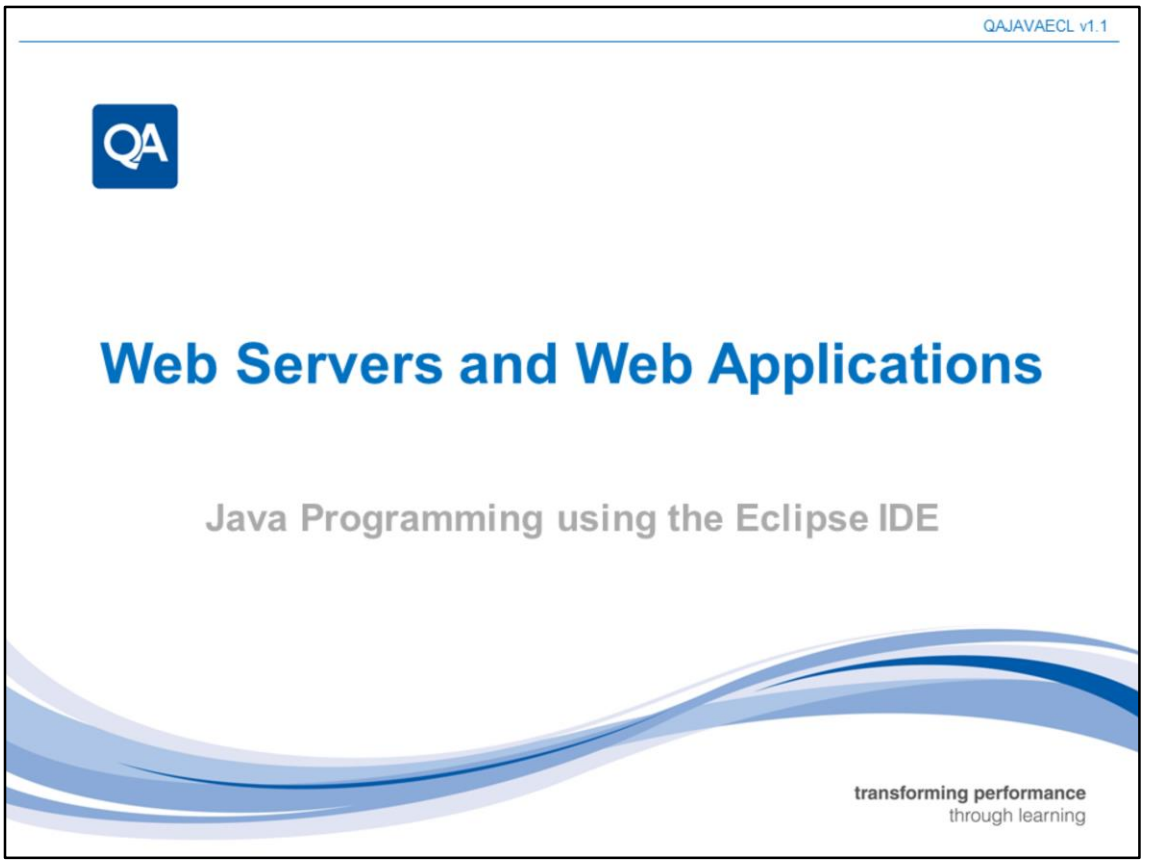

QAJAVAECL v1.1
Web Servers and Web Applications
Java Programming using the Eclipse IDE
transforming performance
through learning

Outline

- **Web Applications**
  - What are web servers?
  - What are web applications?
  - Servlets and JSPs
  - Creating a Java web application

Objectives

- **By the end of this session you should be able to:**
  - Create Java web applications
  - Deploy web applications to a server

Outline

- **Web Application**
  - What are web servers?
  - What are web applications?
  - Servlet and JSP APIs
  - Creating a web application in Java

Outline

- **Web Applications**
  - What are web servers?
  - What are web applications?
  - Servlets and JSPs
  - Creating a Java web application

What are webservers?

- **Webservers are the machines and programs that service requests coming into them**
  - Forward requests that come to the server to the correct applications and locations
  - Handle sessions
- **Apache**
  - Serve static HTML webpages
  - Also can process PHP and other languages
- **Tomcat/Glassfish**
  - Java webserver
  - Deploy and manage WAR
  - Run on a specific port on the machine (default is 8080)

What are web applications?

- **Web applications run over the internet responding to requests from desktop applications or browsers**
  - Exchange messages
  - Deliver and respond to information

Presentation Oriented	Service Oriented

• Display everything using webpages
• JSP/HTML
	• Respond with data, rather than interfaces
	• SOAP (Simple Object Access Protocol)
	  • Uses XML to describe objects
	• REST (Representation State Transfer)
	  • More lightweight than SOAP (no envelope)
	  • Not only XML

Presentation Type Web applications

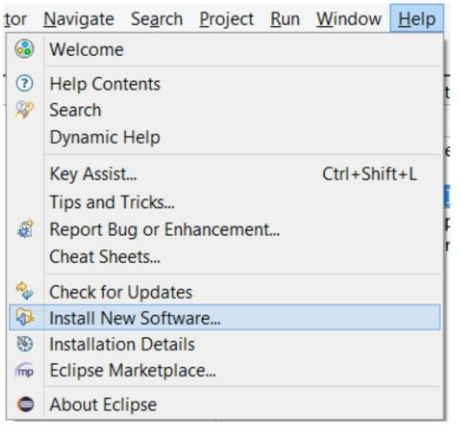
- **These are the original form of web application**
  - Respond to requests on a particular path by processing the request and returning a web page of some type for the browser to display
  - Java uses Servlets as the controllers and JSP pages for presenting the response

Verb Design

- **Four operations**
  - GET, POST, PUT, DELETE
- **Our aim is to keep the different URLs to a minimum**
  - /user
    - Get – retrieves the users
    - Post – adds a new user
    - Put – edit a user
    - Delete - …
- **Try to avoid things like /GetUser?id=1**
  - The same can be achieved with
    - /user/1

Configuring Eclipse

- **Eclipse has a different type of project available for web projects – the Dynamic Web Project**
  - Sets up the folders correctly for the webserver to understand
  - Includes a web server
  - Different context menus for running and debugging
- **This isn't included in the default eclipse installation**
  - Part of the Web Tools Platform
  - Install the plugin using the Help Menu
  - On the next window select "All sites" to work with
  - Search for "Java EE" and "Eclipse Java EE Developer Tools" and install them both



If you downloaded the EE version at the start (which we should have done!) then this is already included.

Java Servlets

- **Servlets are web components which are responsible for generating the dynamic web content**
- **Websites are classified into two types**
  - Static
    - These are created by using HTML
  - Dynamic
    - These websites are created by using server-side programming concepts such Servlets, JSP, PHP and ASP
- **Java Servlet API provides several classes and interfaces that enable the web applications to generate the dynamic content**
  - A servlet can be created by implement javax.servlet.Servlet interface
  - Servlets can also be created by extending either GenericServlet or HttpServlet class

Java Servlets

- **Servlet is a simple Java class**
- **Servlets handles an HTTP request and provide HTTP response**
- **Features of Servlets**
  - A servlet sends an HTTP response in an HTML page form
  - Java servlets are used to create responsive web applications
  - Servlets have access to a variety of APIs
  - Servlets support session management
  - A servlet can communicate with the selected database or backend system
  - A servlet can communicate with another servlet and this concept is known as Servlet-to-Servlet communication
  - A servlet class can have HTML code written in println() method

Java Server Pages (JSP)

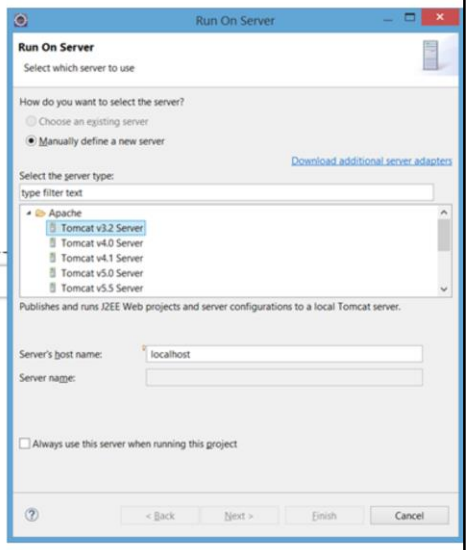
- **JSP is another form of Servlet**
- **JSP are compiled into Java servlet**
- **Features of JSP**
  - Web application development work can be divided across the developers
  - JSPs enable you to embed Java directly into your HTML pages7
  - JSP file can have both HTML tags and JSP tags
  - JSPs enable you to separate the dynamic content from the presentation login
  - JSPs enable you to build custom tags which can directly communicate with Java Beans (Models)
  - JSP support Expression Language
  - JSPs can be deployed on a variety of platforms, including WebLogic Server because JSPs are part of Java EE standard

Creating a dynamic web project in eclipse

- **Create a new dynamic web project**
  - You may need to set a target runtime in the dialog box
  - The webserver is setup for you
  - Right click the project and select “Run as” then “Run on Server”
- **The server is started on port 8080**
  - You will have given your application a context when it was started
  - Point a browser at: localhost:8080/contextName to see the content
- **Index.jsp is loaded by default**
- **To add more pages we use servlets**
  - These map sets of behaviour to URLs
  - You can respond with http directly or use JSPs

Running the Default Project

- **Create a new Dynamic Web Project**
- **Right click and select run on server**
- **You will then be asked which server to run on**
  - Create a new Apache Tomcat 7 Server
  - You may need to download and install it, but eclipse will help with that
  - Click finish and you will see a webpage



15

Creating web applications in Java

- **Web Applications are controlled by Servlets**
  - Create a new servlet via the new → other menu
  - Creates a skeleton servlet

```
@WebServlet("/HomeController")
public class HomeController extends HttpServlet {

public HomeController() {
        super();
}
protected void doGet(HttpServletRequest request,
                           HttpServletResponse response)
                           throws ServletException, IOException {
}
protected void doPost(HttpServletRequest request,
                           HttpServletResponse response)
                           throws ServletException, IOException {
}
}
```


The @WebServlet line is an annotation, these were introduced in Java 5. This line means that any traffic coming into the web server on the address /HomeController will be handled by this class.

The constructor is run when the controllers are setup as part of the server coming online

doGet and doPost are called when a GET or POST request is received at this address

Creating web applications in Java

- **The Request object contains all the information coming in from the request**
  - Form parameters
  - Cookies
  - Authentication
  - Headers
- **The response object is how we return something to the browser**
  - We can write out to the webpage directly

```
response.getWriter().println("Oh Hai");
```

  - But often it is better to use a JSP (java server page) as a response
    - Separation of control and view code
    - We can add objects to the response and retrieve them in the JSP for display

Creating web applications in Java

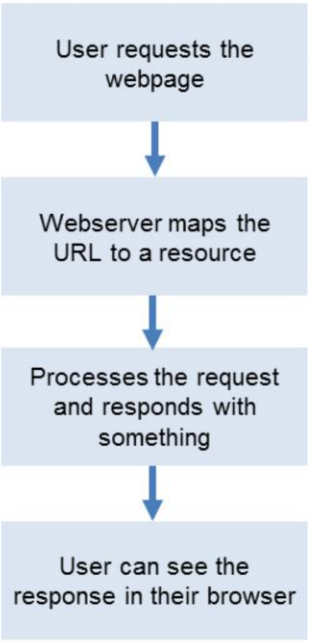
- **The request and response object can be forwarded to a JSP page**

```
//create a user
User user = new User(0, "Alice", "Alice@Email.com");

//pass it to the JSP page
request.setAttribute("user", user);
request.setAttribute("userList", userList);

//output the JSP page
request.getRequestDispatcher("/test.jsp")
                                    .forward(request, response);
```

- **In the JSP page we can get access to these objects**

```
<h1>This is a page all about a User</h1>
<p>ID : ${user.getUserID()}</p>
<p>Name : ${user.getName()}</p>
<p>Email:${user.getEmail() }</p>
```

Control Flow



- **Go to a specific URL**
  - localhost:8080/WebApplication
- **Looks for a servlet or a JSP page with a match**
  - Servlet mapping annotation: @WebServlet("/")
  - runs doGet() or doPost() depending on request type
- **Process the request, create or get objects and pass it to the correct JSP**
  - request.setAttribute(…) adds objects to the request
  - request.getRequestDispatcher("/test.jsp").forward(request, response);
- **The response is generated from the jsp we forwarded to**
  - To get objects from the request we can use ${user.getUserID()}

19

Exercise

- **Create a presentation-style web application**
  - Create a dynamic web project in eclipse
  - Respond to requests
  - Add objects to the request
  - Using JSPs

Summary

- **Web Applications**
  - What are web servers?
  - What are web applications?
  - Servlets and JSPs
  - Creating a Java web application