4

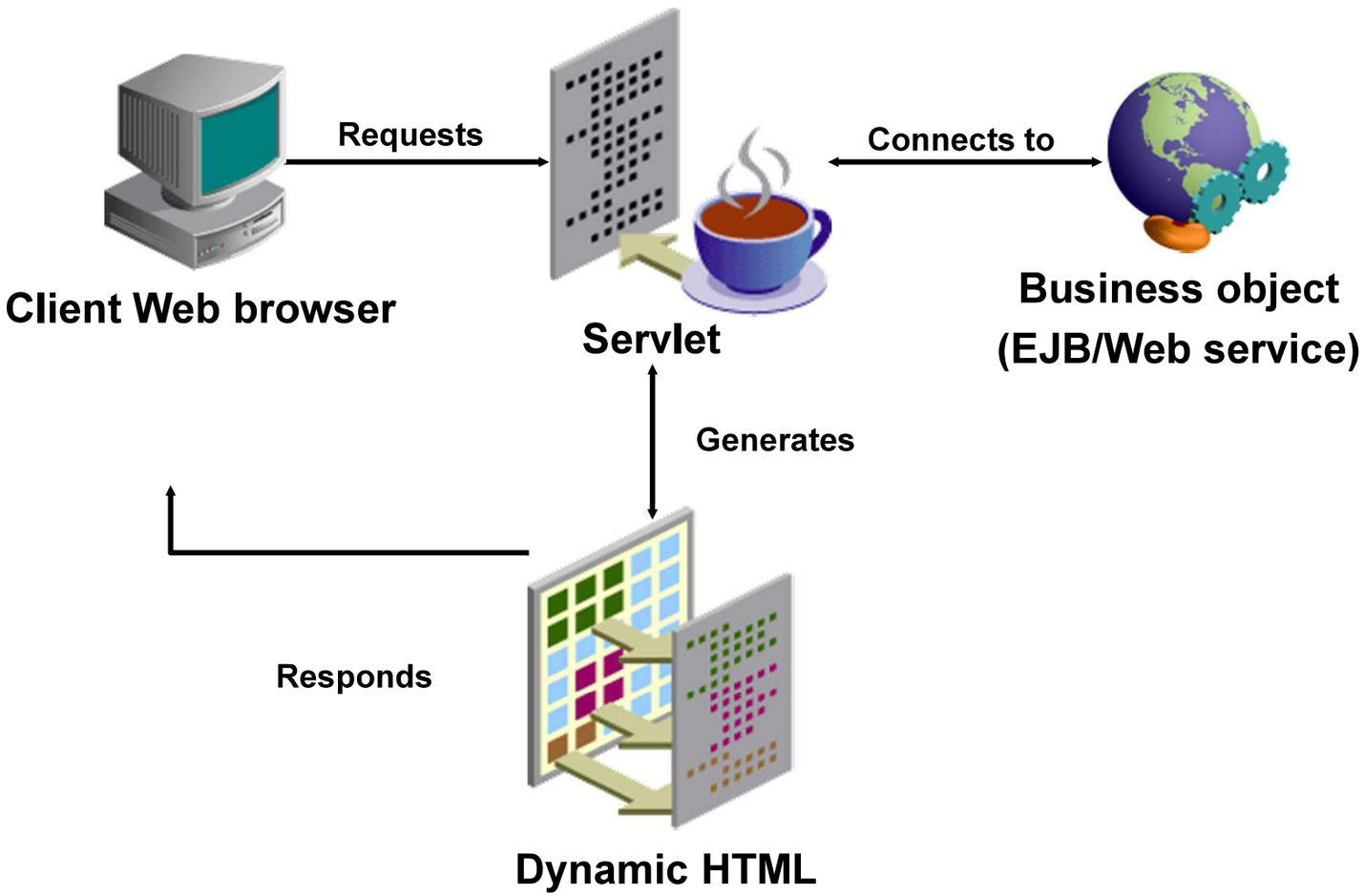
Developing a Web Applications Using Servlets

Objectives

After completing this lesson, you should be able to do the following:

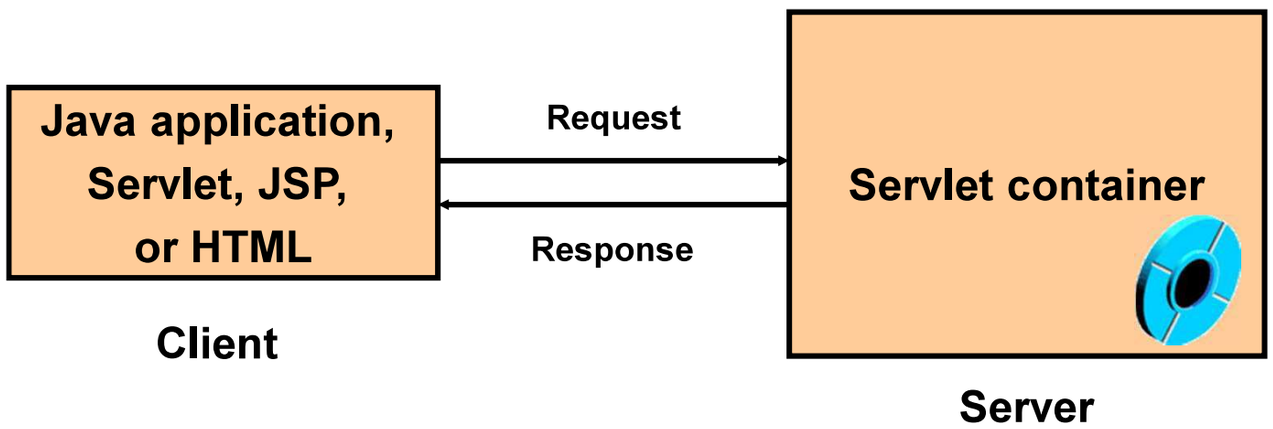
- Define the role of servlets in a Java EE application
- Describe the servlet life cycle
- Describe the request and response architecture
- Implement HTTP servlet methods
- List Java EE servlet mapping techniques
- Handle errors in a servlet

Servlets: Overview



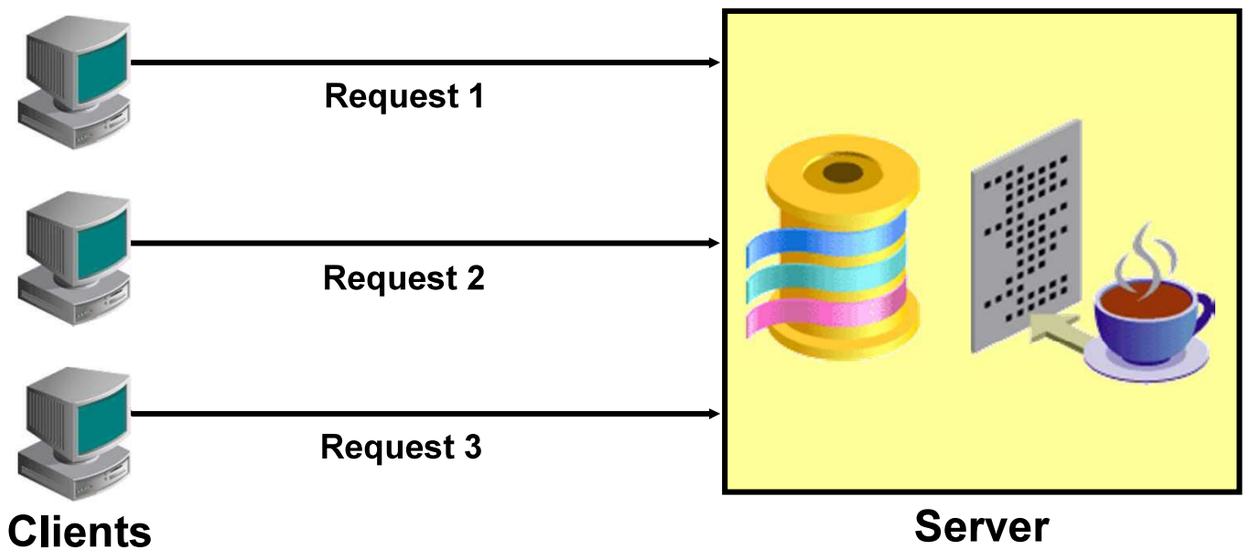
About Java Servlets

- A servlet is a Java class that implements the servlet interface.
- A servlet runs in the context of a special process called a servlet container.
- Servlets can be invoked simultaneously by multiple clients.



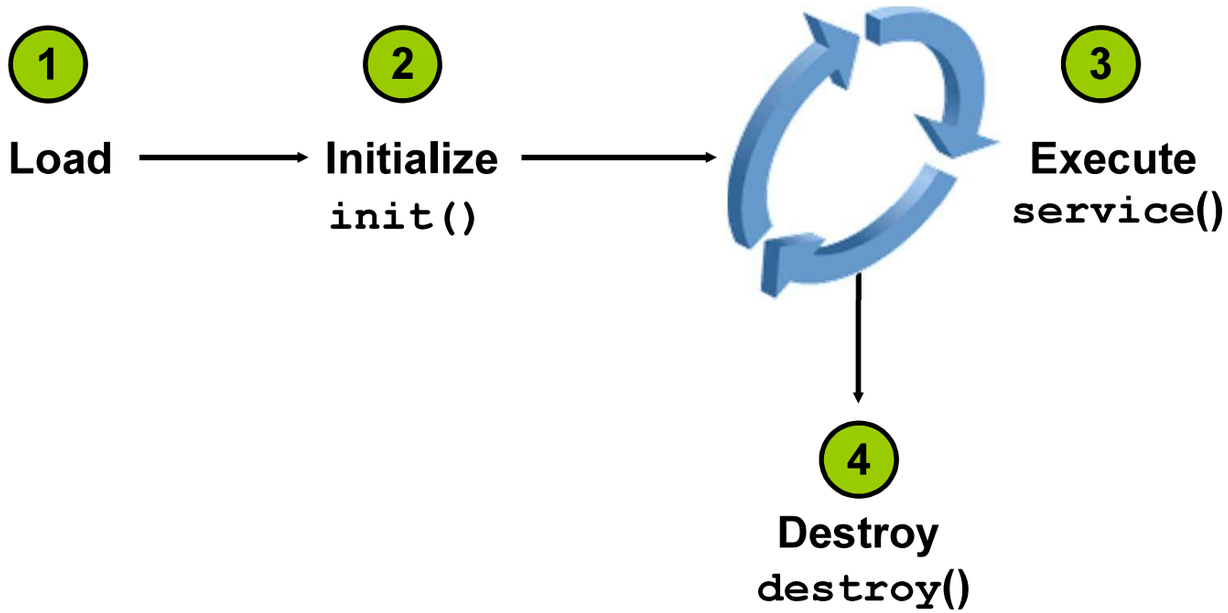
Principal Features of Servlets

- Concurrent requests are possible and common.
- Servlet methods are run in threads.
- Servlet instances are shared by multiple client requests.



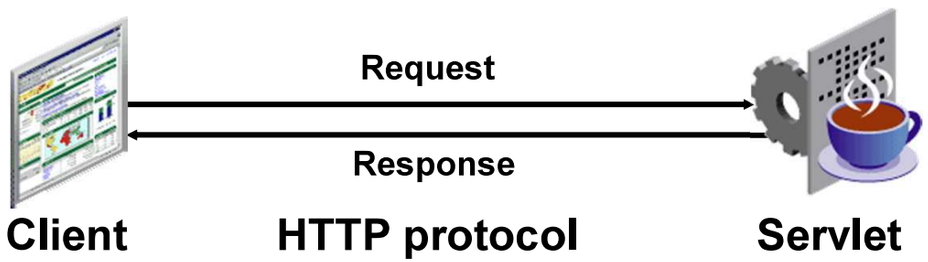
Life Cycle of Servlets

- All actions are carried out inside the server.
- After the initial setup, the response time is less.



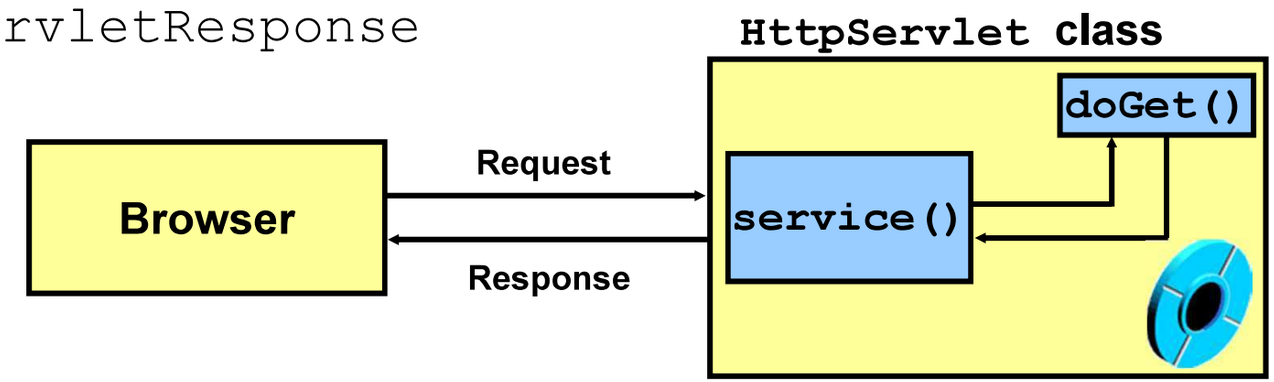
HTTP Servlets

- HTTP servlets extend the `HttpServlet` class, which implements the `Servlet` interface.
- A client makes an HTTP request, which includes a method type that:
  - Can be either a `GET` or `POST` method type
  - Determines what type of action the servlet performs
- The servlet processes the request and sends back a status code and a response.



Inside an HTTP Servlet

- The servlet overrides the `doGet()` or the `doPost()` method of the `HttpServlet` class.
- The servlet engine calls the `service()` method, which in turn calls one of the appropriate `doXxx()` methods.
- These methods take two arguments as input:
  - `HttpServletRequest`
  - `HttpServletResponse`



doGet() Method

- The most common HTTP request method type made to a Web server is `GET`.
- The `service()` method in your servlet invokes the `doGet()` method. The `service()` method is invoked on your behalf by the Web server and the servlet engine.
- The `doGet()` method receives two parameters as input:
  - `HttpServletRequest`
  - `HttpServletResponse`
- Pass parameters by appending them to the URL: http://www.oracle.com/servlet?param1=value1

doPost() Method

- The `doPost()` method can be invoked on a servlet from an HTML form by using:

```
<form method="post" action=…>
```

- The `service()` method in your servlet invokes the `doPost()` method. The `service()` method is invoked by the Web server and the servlet engine.
- The `doPost()` method receives two parameters as input:
  - `HttpServletRequest`
  - `HttpServletResponse`
- Pass parameters using the form field names:

```
<input type="text" name="param1">
```

HttpServletRequest Object

- The `HttpServletRequest` object encapsulates the following information about the client:
  - Servlet parameter names and values
  - The remote host name that made the request
  - The server name that received the request
  - Input stream data
- You invoke one of several methods to access the information:
  - `getParameter(String name)`
  - `getRemoteHost()`
  - `getServerName()`

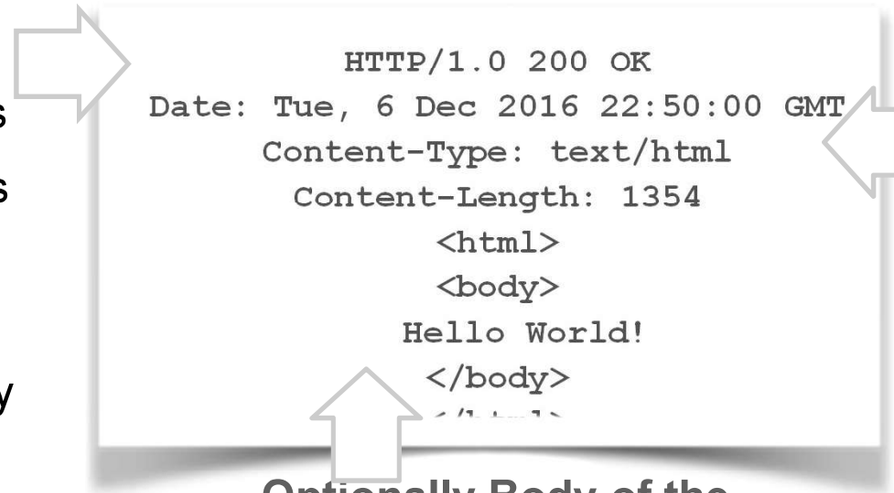
HttpServletResponse Object

- The `HttpServletResponse` object encapsulates information that the servlet has generated:
  - The content length of the reply
  - The Multipurpose Internet Mail Extension (MIME) type of the reply
  - The output stream
- You invoke one of several methods to produce the information:
  - `setContentLength(int length)`
  - `setContentType(String type)`
  - `sendRedirect()`

HTTP Protocol Basics: Getting Responses

Status code that indicates the nature of the response to the client

- 1xx codes: Informational
  Example: 101 Switching Protocols
- 2xx codes: Successful Responses
  Example: 200 OK
- 3xx codes – Redirection
  Example: 301 Moved Permanently
- 4xx codes: Client Errors
  Example: 404 Not Found
- 5xx codes: Server Errors
  Example: 503 Service Unavailable



Optionally Body of the Response

Headers that describe metadata

- **Context Type**
- **Content Length**
- **Character Encoding**
- **Date and Time**
- **And so on**

Example: Response contains 200 OK code and body will be dependent on the method used in the request.

- **`GET`: Response body should contain requested resource content.**
- **`HEAD`: No response body is produced, just headers.**
- **`POST`: Response body should contain result of requested action.**
- **`TRACE`: Response body should echo the request message back.**

Create Servlet

Servlet class:

- Extends HTTPServlet
- Is mapped to urlPattern
- Provides HTTP Request Handing operations (see next page)

Described with:

- Annotations or
- `web.xml` file or
- Both

```
package demos;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
@WebServlet(name = "ProductDisplay", urlPatterns = {"/products"})
public class ProductDisplayServlet extends HttpServlet {
  // Override Servlet Methods (next page)
}
```

```
<?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>ProductDisplay</servlet-name>
    <servlet-class>demos.ProductDisplayServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ProductDisplay</servlet-name>
    <url-pattern>products</url-pattern>
  </servlet-mapping>
</web-app>
```

HttpSession

- The unique identity for the client is an `HttpSession` object.
- The object is created by using the `getSession()` method of the `HttpRequest` object.

```
HttpSession session = req.getSession(true);
```

- Any servlet that responds to a client request can create this object.
- An object can be potentially shared across several servlets. (Every servlet within an application can identify with this client.)

Session Objects

- With session objects, you can:
  - Put items into the object (values persist across multiple invocations from the same client)
  - Access items from the object
  - Obtain the session identity
  - Find out when the session was last accessed
- Items put in a session object can:
  - Implement the `Serializable` interface
  - Be relocated to a different server
  - Persist across servlet crashes

Methods for Invoking Servlets

- Invoke servlets from a client by:
  - Entering the servlet URL in a browser
  - Embedding the servlet URL in an HTML, or a JSP page, or another servlet (an `href` link)
  - Submitting a form to the servlet (using the action tag)
  - Using URL classes in client Java applications
- Invoke servlets inside the Java EE container by defining a chain of servlets or JSPs.

Quiz

`HttpSession` is an interface that provides a way to identify a user across more than one page request.

1. True
2. False

Handling Input: The Form

You can use an HTML form and the `doPost()` method to modify the `HelloWorld` servlet.

```
<html>
<body>
<form method="post" action="newhelloworld">
Please enter your name. Thank you.
<input type="text" name="firstName"> <P>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Handling Input: The Servlet

```
public class NewHelloWorld extends HttpServlet {
  public void doPost(
  HttpServletRequest  req, HttpServletResponse res)
  throws ServletException, IOException{
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println ("<html><body>");
    String name = req.getParameter("firstName");
    if ((name != null) && (name.length() > 0))
      out.println ("Hello: " + name +
      "  How are you?");
    else
      out.println ("Hello Anonymous!");
    out.println ("</body></html>");
  }}
```

Retrieve Parameters

Servlet can get parameters submitted by the client.

- Use

  method

  request parameter names.
- Iterate

  of

  get their values.

```
...
protected void processRequest(HttpServletRequest request,
                              HttpServletResponse response)
                              throws ServletException, IOException {
   response.setContentType("text/html;charset=UTF-8");
   PrintWriter out = response.getWriter();
   out.println("<html><body>");
   Enumeration<String> parameters = request.getParameterNames();
   while (parameters.hasMoreElements()) {
      String name = parameters.nextElement();
      String value = request.getParameter(name);
      out.println("Parameter:"+name+"="+value+"<br>");
   }
}
...
```

Initialization and Destruction

Servlets also define the `init()` and `destroy()` methods in addition to the `service()` method.

- `init()`:
  - Can be used to retrieve initialization parameters
  - Takes a `ServletConfig` object as a parameter
  - Is invoked when the servlet instance is created
  - Is useful for obtaining database connections from a connection pool
- `destroy()`:
  - Takes no arguments
  - Is invoked when the servlet is about to be unloaded
  - Is useful for releasing resources

Produce Different Content Types

- Output produced by the Servlet does not have to be HTML or text.
  - Set HTTP Headers to inform invoker about the nature of content your servlet is going to generate.
    - Content-Type
    - Content-Length (optional, but could be a good idea)
  - Use PrintWriter or ServletOutputStream objects.

- This example is artificial.
  - A static without the use of servlet.
  - Consider dynamically.

```
...
protected void processRequest(HttpServletRequest request,
                              HttpServletResponse response)
                              throws ServletException, IOException {
   response.setContentType("image/jpeg");
   File f = new File("somePicture.jpeg");
   response.setContentLength((int)f.length());
   ServletOutputStream out = response.getOutputStream();
   Files.copy(f.toPath(), out);
   out.close();
}
...
```

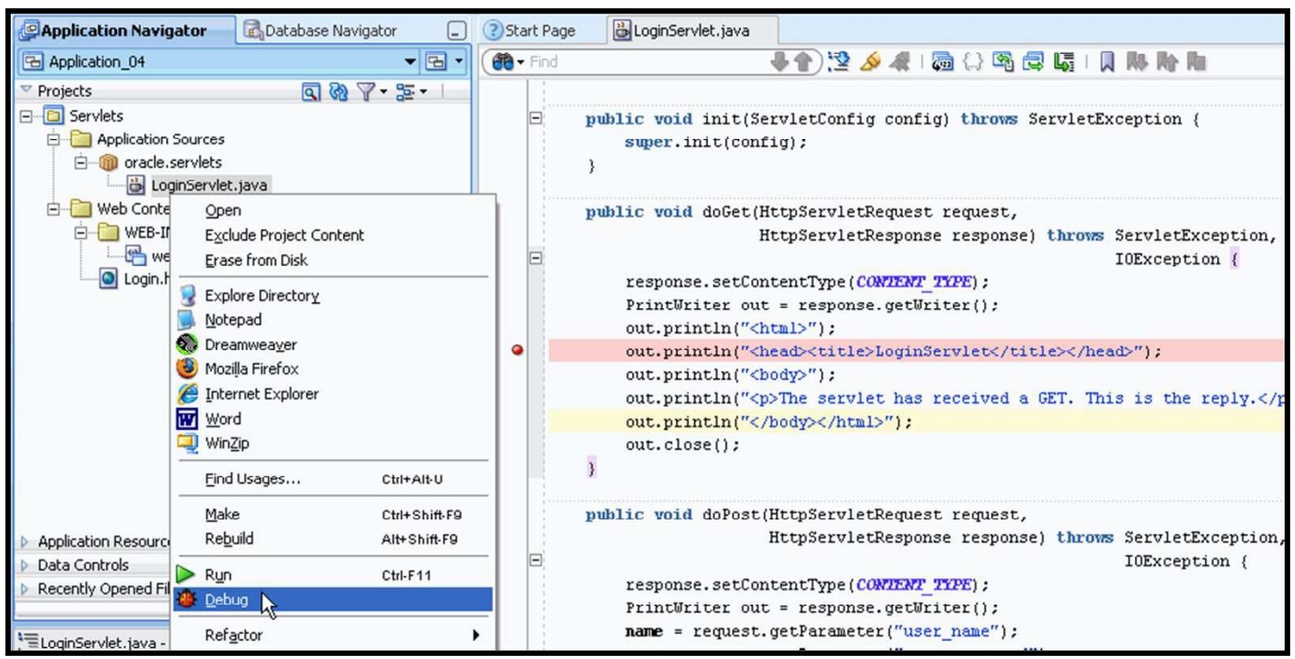
Error Handling

- `ServletException`:
  - Is generated to indicate a generic servlet problem
  - Is subclassed by `UnavailableException` to indicate that a servlet is unavailable, either temporarily or permanently
  - Is handled by the servlet engine in implementation-dependent ways
- `IOException` is generated if there is an input or output error when processing the request.

Debugging a Servlet

Servlets can be debugged in the following ways:

- By setting breakpoints and using the debugger in IDE
- By viewing the source of the generated HTML



IDE Environment

The Servlet Wizard in JDeveloper makes it easy for you to write servlets.

The wizard:

- Provides the `doGet()` and `doPost()` method skeletons
- Provides an environment for running the servlet within the integrated development environment (IDE)
- Dynamically creates the `web.xml` file for running the servlet from the IDE

Servlet Mapping

- Mapping a servlet refers to how a client can access a servlet.
- You can map a servlet by using the mapped URL:

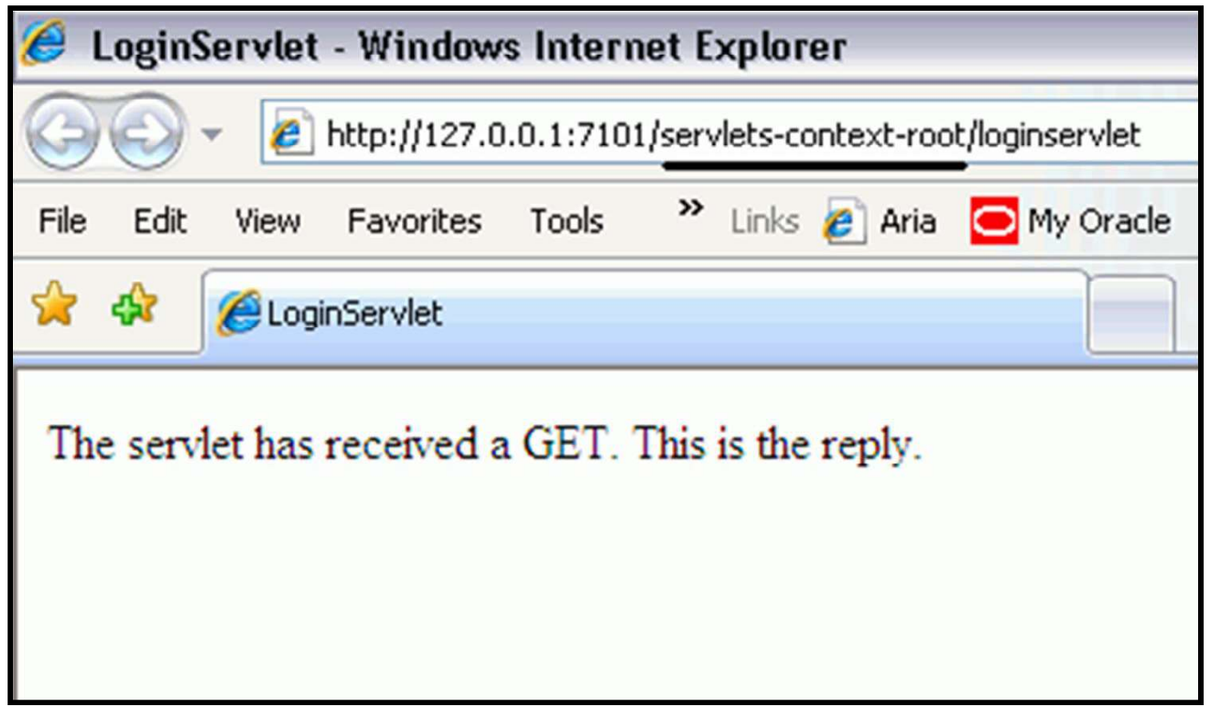
  http://host:port/<context-root>/<mappedservletname>
- <context-root> is the mapping for the Web module.

Servlet Mapping in IDE

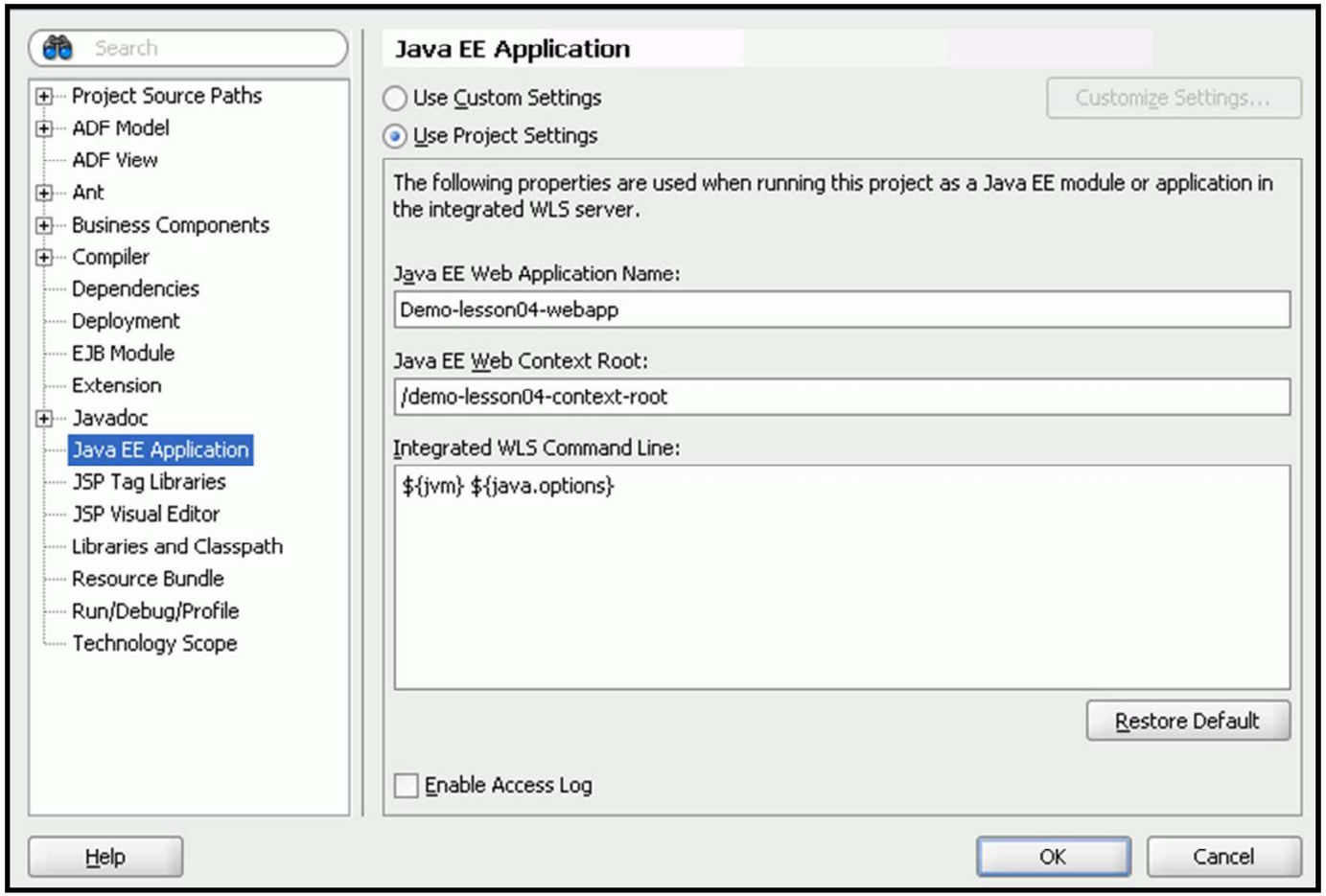
IDE provides the standard Java EE model for mapping servlets by using the `web.xml` file:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.2//EN" "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
   <servlet>
      <servlet-name>LoginServlet</servlet-name>
      <servlet-class>oracle.servlets.LoginServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>LoginServlet</servlet-name>
      <url-pattern>/loginservlet</url-pattern>
   </servlet-mapping>
   …
</web-app>
```

Invoking a Servlet



Specifying Java EE Web Module Settings



Summary

In this lesson, you should have learned how to:

- Describe the servlet life cycle
- Develop and run a servlet in IDE
- Map a servlet in a Java EE server
- Collect information from a client
- Respond to the client
- Handle errors in a servlet