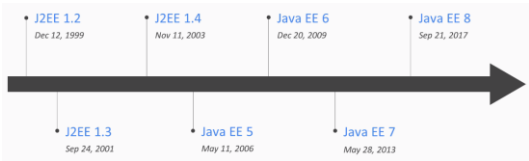
Server Side Development

Server Extensions

- Several different tools are available for extending the server capabilities
  - CGI scripting – LAMP
  - Active Server Pages (ASP)
  - VB .Net architecture
  - Java enterprise architecture (Servlet, JSP)
  - Node.js -- MEAN MERN
  - Ruby on Rails
  - …
- These tools process incoming requests from the user and generate custom/dynamic html pages

42

42



Java enterprise platform history

Platform version	Released	Specification	Java SE Support	Important Changes
Jakarta EE 10	2022-09-13[9]	10	Java SE 17 Java SE 11	Removal of deprecated items in Servlet, Faces, CDI and EJB (Entity Beans and Embeddable Container). CDI-Build Time.
Jakarta EE 9.1	2021-05-25[10]	9.1	Java SE 11 Java SE 8	JDK 11 support
Jakarta EE 9	2020-12-08[11]	9	Java SE 8	API namespace move from `javax` to `jakarta`
Jakarta EE 8	2019-09-10[12]	8	Java SE 8	Full compatibility with Java EE 8
Java EE 8	2017-08-31	JSR 366	Java SE 8	HTTP/2 and CDI based Security
Java EE 7	2013-05-28	JSR 342	Java SE 7	WebSocket, JSON and HTML5 support
Java EE 6	2009-12-10	JSR 316	Java SE 6	CDI managed Beans and REST JAXRS
Java EE 5	2006-05-11	JSR 244	Java SE 5	Java annotations
J2EE 1.4	2003-11-11	JSR 151	J2SE 1.4	WS-I interoperable web services[13] EL in JSP
J2EE 1.3	2001-09-24	JSR 58	J2SE 1.3	Java connector architecture[14]
J2EE 1.2	1999-12-17	1.2	J2SE 1.2	Initial specification release Servlet, JSP, EJB

43

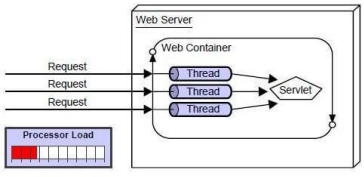
What is a Servlet?

- Servlet is Java's answer to CGI programming
- Servlet is a technology i.e. used to create web application.
- Servlet is a web component that is deployed on the server to create dynamic web page.
  - Program runs on Web server and builds pages on the fly
- Servlet is an API that provides many interfaces and classes including documentations.
- Servlet is an interface that must be implemented for creating any servlet.

44

What's a Servlet?

- Providing the functionalities of CGI scripts with a better API and enhanced capabilities.
- Against CGI : The web container creates threads for handling the multiple requests to the servlet.
- Better performance: because it creates a thread for each request not process.



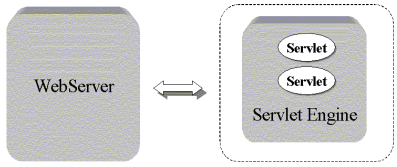
- Portability: because it uses java language.
- Robust: Servlets are managed by JVM so we don't need to worry about memory leak, garbage collection etc.
- Secure: because it uses Java language
- Supported by servers from Apache, Oracle, IBM…

46

46

Servlet Engine/container

- A servlet engine (or servlet container) provides the run-time environment in which a servlet is executed.
- The servlet engine manages the life-cycle of servlets (i.e., from their creation to their destruction).
- The servlet engine: loads, executes and destroyes servlets
- Apache Tomcat is a open source servlet container http://tomcat.apache.org
- GlassFish is another popular one



Relationships between Web server, Servlet engine and Servlets.

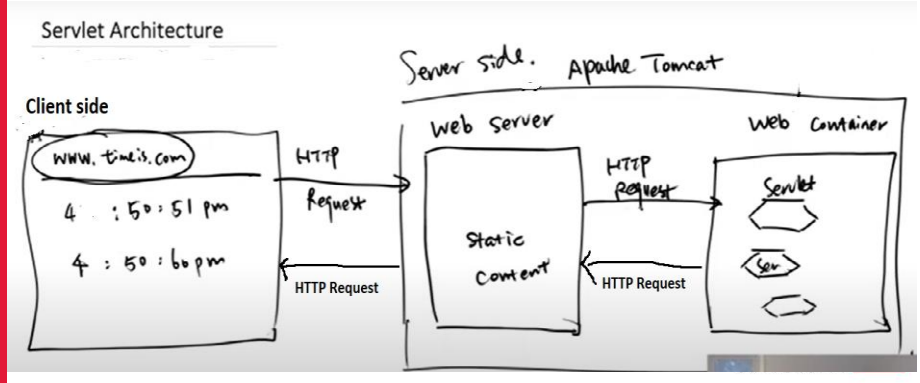
Not same as LAMP

47

47

Tomcat is a stand-alone web server and a servlet container
It is open source and free for usage ☺

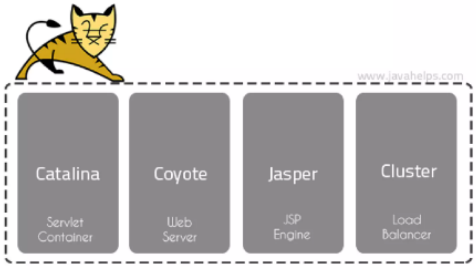
Tomcat is an open-source Java servlet container that implements many Java Enterprise Specs such as the Websites API, Java-Server Pages (JSP) and last but not least, the **Java Servlet**. The complete name of Tomcat is "Apache Tomcat"



48

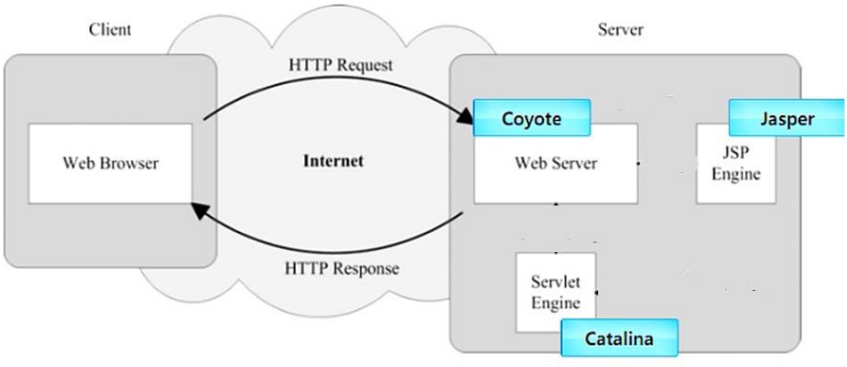
Tomcat is an open-source Java servlet container that implements many Java Enterprise Specs such as the Websites API, Java-Server Pages (JSP) and last but not least, the **Java Servlet**. The complete name of Tomcat is "Apache Tomcat"

Tomcat Architecture(Core Component)

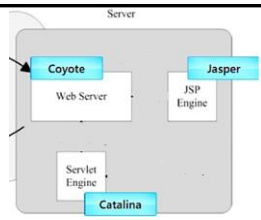


49

Tomcat is an open-source Java servlet container that implements many Java Enterprise Specs such as the Websites API, Java-Server Pages (JSP) and last but not least, the **Java Servlet**. The complete name of Tomcat is "Apache Tomcat"



50



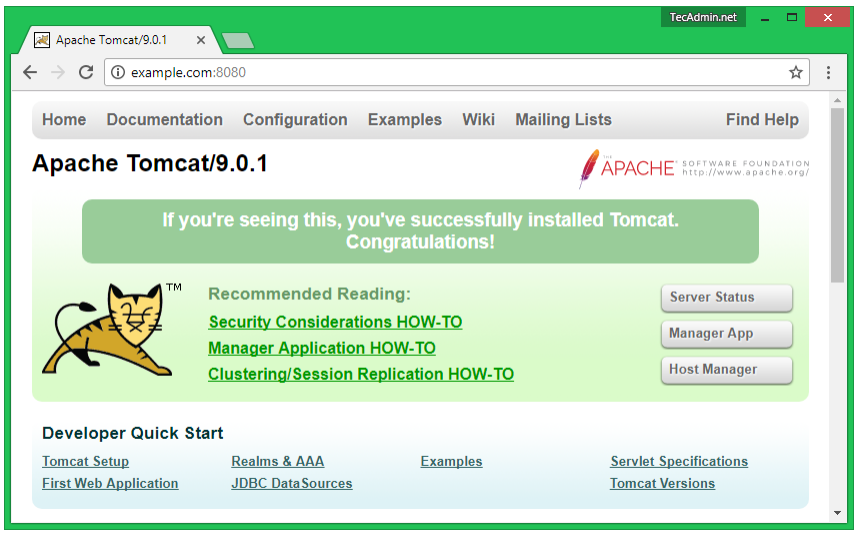
Catalina
Catalina is Tomcat's servlet container. Catalina implements Sun Microsystems' specifications for servlet and JavaServer Pages (JSP).

Coyote
Coyote is a Connector component for Tomcat that supports the HTTP 1.1 and 2 protocol as a web server. This allows Catalina, nominally a Java Servlet or JSP container, to also act as a plain web server that serves local files as HTTP documents. Coyote listens for incoming connections to the server on a specific TCP port and forwards the request to the Tomcat Engine to process the request and send back a response to the requesting client.

Jasper
Jasper is Tomcat's JSP Engine. Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them.

51

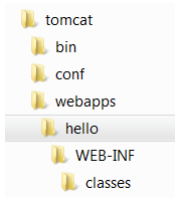
51

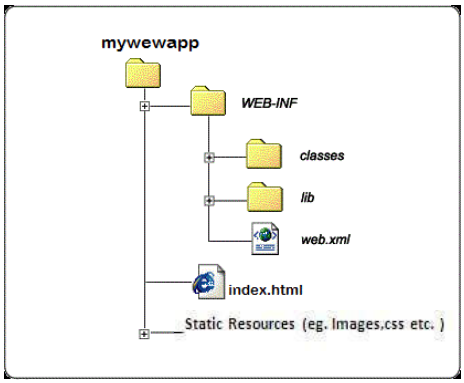


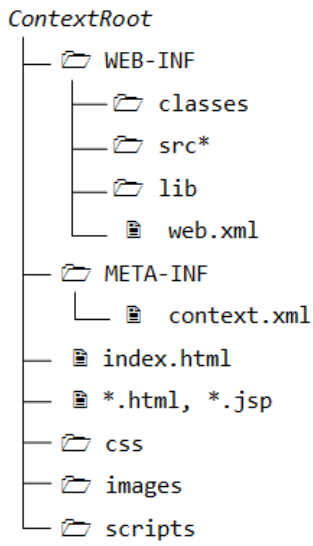
Download zip/tar file. Unzip.

52

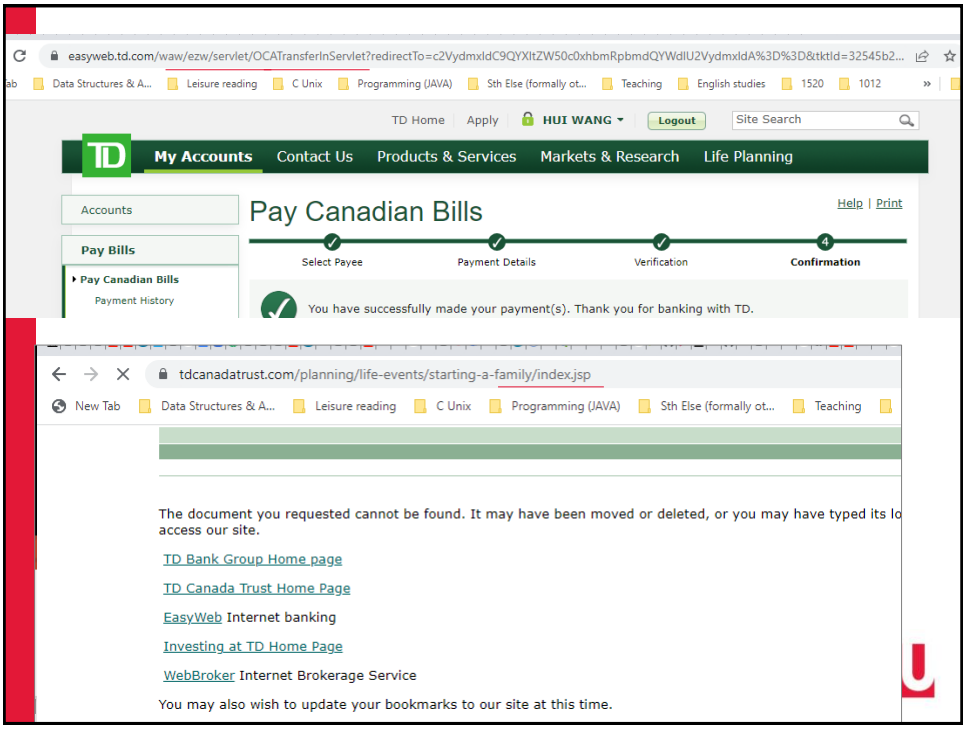
52


tomcat
bin
conf
webapps
hello
WEB-INF
classes


mywewapp
WEB-INF
classes
lib
web.xml
index.html
Static Resources (eg. Images,css etc.)


ContextRoot
WEB-INF
classes
src*
lib
web.xml
META-INF
context.xml
index.html
*.html, *.jsp
css
images
scripts

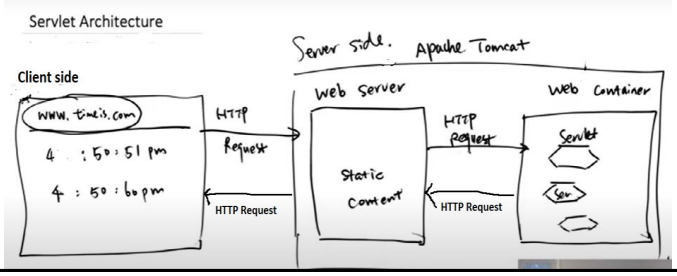
53


easyweb.td.com/waw/ezw/servlet/OCATransferInServlet?redirectTo=c2VydmxldC9QYXItZW50c0xhbmRpbmdQYWdlU2VydmxldA%3D%3D&tktId=32545b2...
Data Structures & A...
Leisure reading
C Unix
Programming (JAVA)
Sth Else (formally ot...
Teaching
English studies
1520
1012
TD Home
Apply
HUI WANG
Logout
Site Search
My Accounts
Contact Us
Products & Services
Markets & Research
Life Planning
Accounts
Pay Bills
Pay Canadian Bills
Payment History
Pay Canadian Bills
Help | Print
Select Payee
Payment Details
Verification
Confirmation
You have successfully made your payment(s). Thank you for banking with TD.
tdcanadatrust.com/planning/life-events/starting-a-family/index.jsp
New Tab
Data Structures & A...
Leisure reading
C Unix
Programming (JAVA)
Sth Else (formally ot...
Teaching
The document you requested cannot be found. It may have been moved or deleted, or you may have typed its lo
access our site.
TD Bank Group Home page
TD Canada Trust Home Page
EasyWeb Internet banking
Investing at TD Home Page
WebBroker Internet Brokerage Service
You may also wish to update your bookmarks to our site at this time.

56

Servlet Overview and Architecture

- Client sends HTTP request
- Servlet container receives request, directs it to the appropriate servlet
- Servlet does processing (including interacting with databases)
- Servlet returns results to client in form of HTML document



57

Java networking capabilities

- Servlets and Java Server Pages (JSP)
  - Request-response model
  - Packages
    - **`javax.servlet`** **(servlets)**
    - **`javax.servlet.http`** **(servlets)**
    - **`javax.servlet.jsp`** **(JSPs)**
    - **`javax.servlet.tagext`** **(JSPs)**

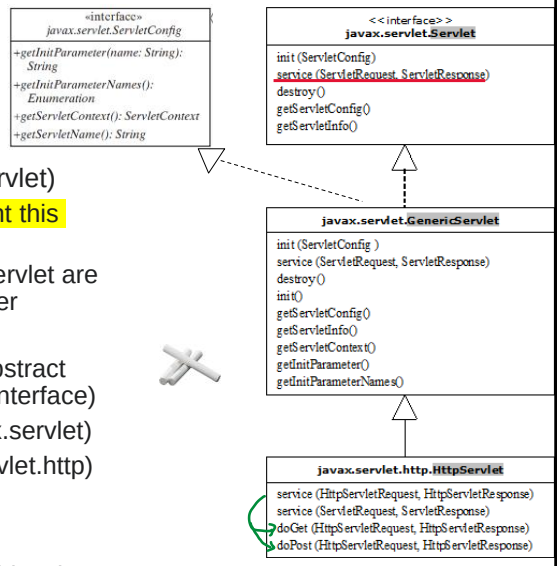
Jarkatar in Tomcat 10+

- Servlets – Web-based solutions
  - Secure access to Website
  - Interact with databases
  - Dynamically generate custom HTML documents
- JSPs provide some of same functionality without getting into details of servlets

58

58

Servlet interface

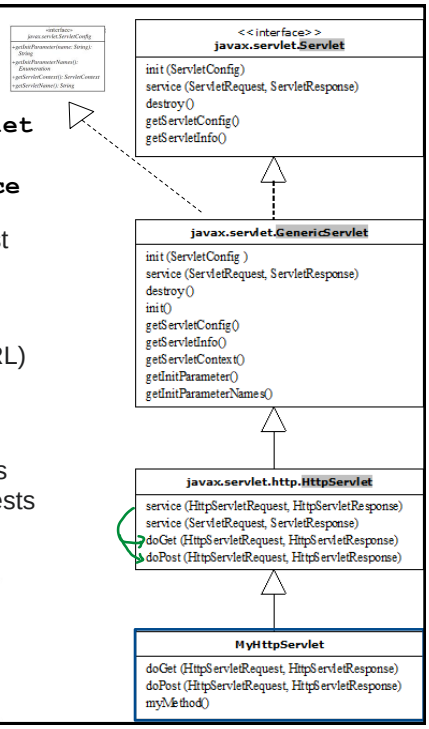


- Interface **`Servlet`** (javax.servlet)
  - All servlets must implement this interface
  - All methods of interface Servlet are invoked by servlet container
- Servlet implementation (two abstract classes implement **`Servlet`** interface)
  - **`GenericServlet`** (javax.servlet)
  - **`HttpServlet`** (javax.servlet.http)
  - **`service`** method
    - **`ServletRequest`** object is data from client
    - **`ServletResponse`** object is data to the client

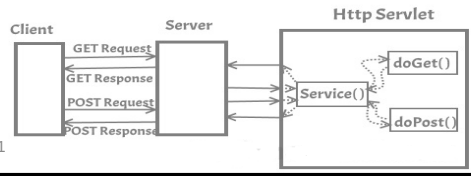
60

60

HttpServlet Class



- Web-based servlets typically implement **`Servlet`** by extending class **`HttpServlet`**
- **`HttpServlet`** overrides method **`service`**
  - Just call **`doGet()`**, **`doPost()`**, **`doPut()...`** according to the request method
- Two most common HTTP request types
  - *get* requests (parameter rides on URL)
  - *post* requests (parameter sent as message)
- Method **`doGet`**() responds to get requests
- Method **`doPost`**() responds to post requests



61

61

HttpServlet

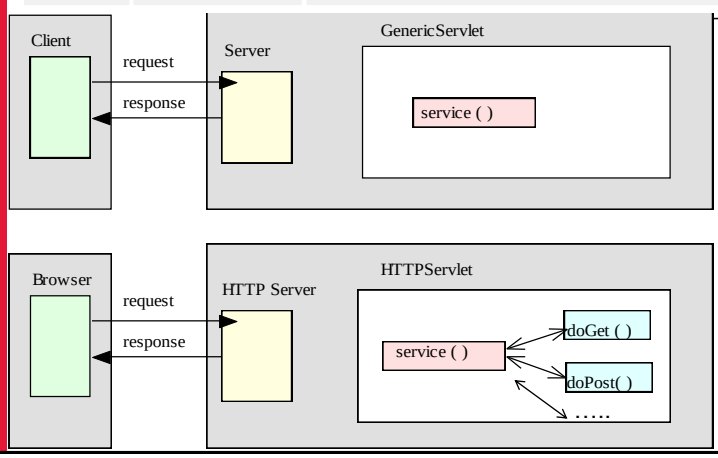
METHOD	SERVLET METHOD
GET	doGet()
HEAD	doHead()
POST	doPost()
PUT	doPut()
DELETE	doDelete()
OPTIONS	doOptions()
TRACE	doTrace()

```java
protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {

    String method = req.getMethod();

    if (method.equals(METHOD_GET)) {
        long lastModified = getLastModified(req);
        if (lastModified == -1) {
            // servlet doesn't support if-modified-since, no reason
            // to go through further expensive logic
            doGet(req, resp);
        } else {
            long ifModifiedSince;
            try {
                ifModifiedSince = req.getDateHeader(HEADER_IFMODSINCE);
            } catch (IllegalArgumentException iae) {
                // Invalid date header - proceed as if none was set
                ifModifiedSince = -1;
            }
            if (ifModifiedSince < (lastModified / 1000 * 1000)) {
                // If the servlet mod time is later, call doGet()
                // Round down to the nearest second for a proper compare
                // A ifModifiedSince of -1 will always be less
                maybeSetLastModified(resp, lastModified);
                doGet(req, resp);
            } else {
                resp.setStatus(HttpServletResponse.SC_NOT_MODIFIED);
            }
        }

    } else if (method.equals(METHOD_HEAD)) {
        long lastModified = getLastModified(req);
        maybeSetLastModified(resp, lastModified);
        doHead(req, resp);

    } else if (method.equals(METHOD_POST)) {
        doPost(req, resp);

    } else if (method.equals(METHOD_PUT)) {
        doPut(req, resp);

    } else if (method.equals(METHOD_DELETE)) {
        doDelete(req, resp);

    } else if (method.equals(METHOD_OPTIONS)) {
        doOptions(req,resp);

    } else if (method.equals(METHOD_TRACE)) {
        doTrace(req,resp);
```

62

62

METHOD	SERVLET METHOD	PURPOSE
GET	doGet()	Retrieves the resource at the specified URL
HEAD	doHead()	Identical to GET, except only the headers are returned
POST	doPost()	Typically used for web form submission
PUT	doPut()	Stores the supplied entity at the URL
DELETE	doDelete()	Deletes the resource identified by the URL
OPTIONS	doOptions()	Returns which HTTP methods are allowed
TRACE	doTrace()	Used for diagnostic purposes



63

HttpServletRequest HttpServletResponse

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
```

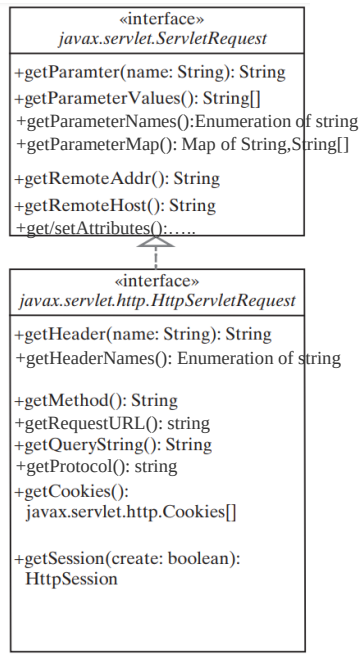
- **HttpServletRequest** and **HttpServletResponse** objects enable interaction between client and server

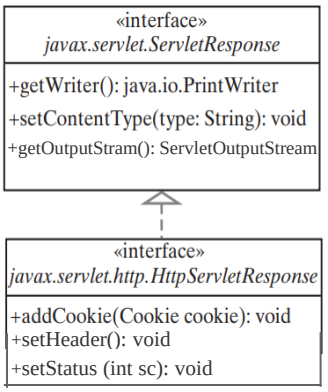
- When a request is received, Servlet container:
  - creates an **HttpServletRequest** object
  - passes it to the servlet's **service** method (that is, **doGet** or **doPost)**
  - **HttpServletRequest** object contains the request info from the client (e.g., header info, body)

  - creates an **HttpServletResponse** object
  - passes it to the servlet's **service** method **(**that is, **doGet or doPost)**
  - Allow to format and send response back to client

64

64





67

revisit

67

Servlets Writing a Servlet

- Create a servletclass
  - extend **`HttpServlet`**
- Implement the **`doGet`**() or **`doPost`**() method (or others due to method e.g.,**`doPut`**)
  - Both methods accept two parameters
    - **`HttpServletRequest`**
    - **`HttpServletResponse`**
- Generally actions in the following order
  1. Set the HTTP Content-Type header of the response. The MIME type portion of this header is typically **`text/html`**
  2. Obtain the (Print) writer from the response object
     `getWriter();`
  3. Process input data and generate output (in html format) and write to the writer
     - If needed, obtain parameters from the request object using
       - **`getParameter(String name)`**
       - **`getParameterValues(String name)`**
       - **`getParameterNames()`**
       - …
  4. Close the writer

70

70

Servlets handles form data parsing automatically using the following methods depending on the situation −

`getParameter(String name)` − You call **`request`**.**`getParameter`**() method to get the value of a form parameter.

`getParameterValues`(**`String name`**) − Call this method if the parameter appears more than once and returns multiple values, for example checkbox.

`getParameterNames`() − Call this method if you want a complete list of all parameters in the current request.

71

HelloWorld

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

public class HelloServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                 throws IOException, ServletException
    {
        response.setContentType("text/html"); //step1

        PrintWriter out = response.getWriter();//step2

        out.println("<html>"); //step3
        out.println("<head>");
        out.println("<title>Hello 4413!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello 4413!</h1>");
        out.println("</body>");
        out.println("</html>");

        out.close();    // step 4
    }
}
```

All servlets we will write are subclasses of `HttpServlet`

Server calls doGet() in response to GET request

Interfaces implemented by request/response objects

First two things done by typical servlet; must be in this order

Production servlet should catch these exceptions

Do backend computations

Simple version, use server to generate output

72

Generated Markup	HelloServlet.java
`<html>`	`out.println("<html>");`
`<head>`	`out.println("<head>");`
`<title>Hello World!</title>`	`out.println("<title>Hello World!</title>");`
`</head>`	`out.println("</head>");`
`<body>`	`out.println("</body>");`
`<h1>Hello World!</h1>`	`out.println("<h1>Hello World!</h1>");`
`</body>`	`out.println("</body>");`
`</html>`	`out.println("</html>");`

73

YORK UNIVERSITÉ UNIVERSITY

73

Servlets vs. Java Applications

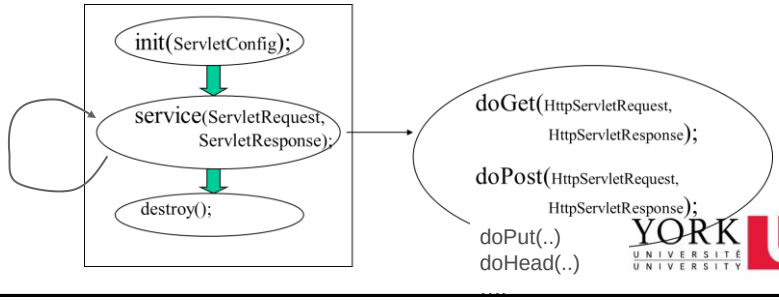
- Servlets do not have a main()
  - The **main**() is in the server
  - Entry point to servlet code is via call to a method (**doGet**() in the example)
- Servlet interaction with end user is indirect via *request/response* object APIs
  - Actual HTTP request/response processing is handled by the server
- Primary servlet output is typically HTML

74

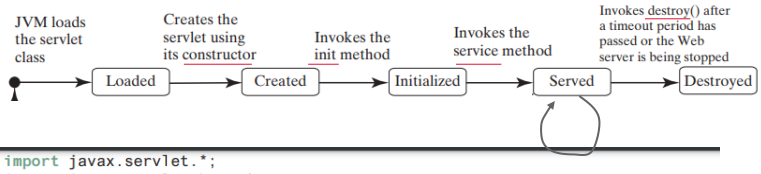
Servlet Life Cycle

Servlet API life cycle methods

- Create by constructor
- **init()**: called when servlet is instantiated; must return before any other methods will be called
- **service()**: method called directly by server when an HTTP request is received; default **service**() method calls **doGet**() **doPost** (or related methods)
- **destroy():** called when server shuts down

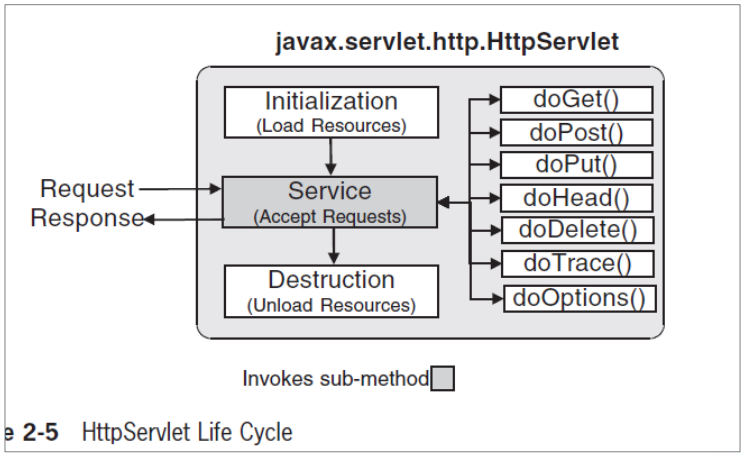


75




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

public class MyServlet extends HttpServlet {
  /** Called by the servlet engine to initialize servlet */
  public void init() throws ServletException {
    ...
  }

  /** Process the HTTP Get request */
  public void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    ...
  }

  /** Process the HTTP Post request */
  public void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    ...
  }

  /** Called by the servlet engine to release resource */
  public void destroy() {
    ...
  }
```

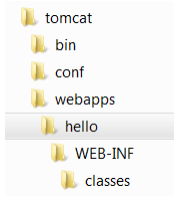
76

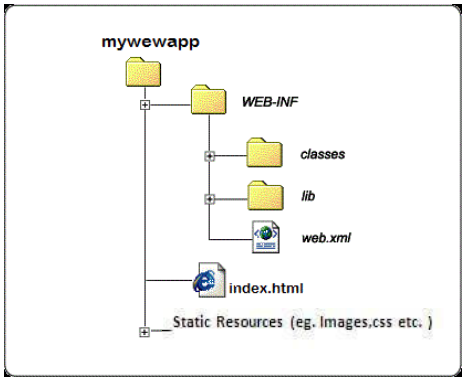


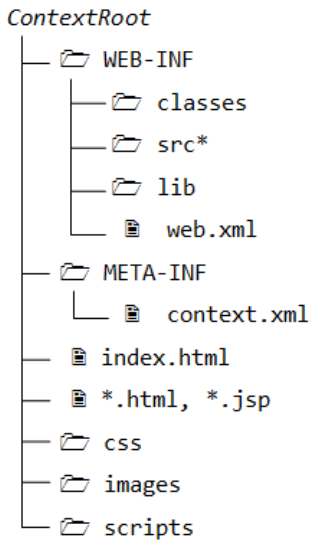
e 2-5 HttpServlet Life Cycle

77

77







// Compile with servlet API library
javac -cp .;c:\tomcat\lib\servlet-api.jar HelloServlet.java
javac -cp .:$HOME/tomcat/lib/servlet-api.jar HelloServlet.java

Need to restart server

78

78

Application Deployment

Deployment Descriptor *web.xml*

- Conveys configuration information of a web application
- The primary elements of a deployment descriptor file
  - Servlet definitions & mappings
  - Servlet context initialization parameters
  - Welcome pages
  - Error pages
  - File based security
- Rules for the deployment descriptor file
  - Resides at the top level of the WEB-INF directory
  - Must be a well formed XML file called web.xml
  - Must conform to the dtd (located at http://java.sun.com/dtd/web-app-2-3.dtd)

81 of 99

81

Application Deployment

Deployment Descriptors - Header

- Header denotes the version of XML
  - <?xml version="1.0" encoding="ISO-8859-1"?>
- Describes the the DTD for the application

  <!DOCTYPE web-app

  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

  “http://java.sun.com/dtd/web-app_2_3.dtd">

- Description of the application enclosed in web-app tags

```
<web-app>
  Contents of the file
</web-app>
```

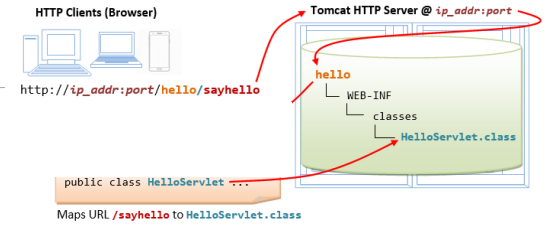
82

82

Example 1

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN“
   "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

  <servlet>
    <servlet-name>HelloS</servlet-name>
    <servlet-class>HelloServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloS</servlet-name>
    <url-pattern>/sayHello</url-pattern>
  </servlet-mapping>

</web-app>
```



83

83

HelloWorld

```
<servlet>
    <servlet-name>HelloS</servlet-name>
    <servlet-class>HelloServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloS</servlet-name>
    <url-pattern>/sayHello</url-pattern>
  </servlet-mapping>
```

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

@WebServlet("/sayhello")
public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                 throws IOException, ServletException
    {
    }
```

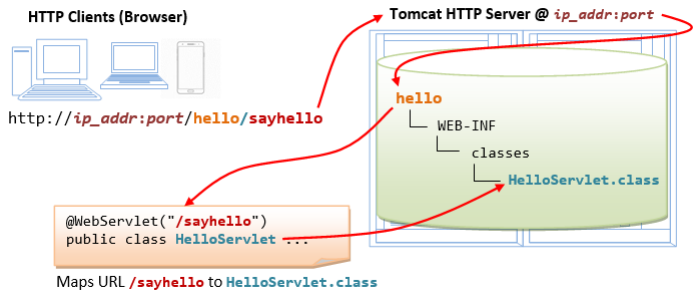
- In newer version (by Servlet 3), can use annotations `@WebServlet` to replace the configuration
- But we still need **`web.xml`** for other configurations (later)

85

HelloWorld

```
<servlet>
    <servlet-name>HelloS</servlet-name>
    <servlet-class>HelloServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloS</servlet-name>
    <url-pattern>/sayHello</url-pattern>
  </servlet-mapping>
```

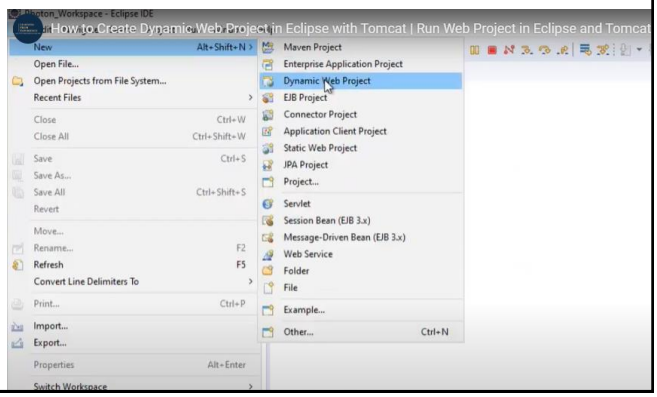
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet("/sayhello") //or @webServlet(urlPatterns={"/sayhello"})
public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                 throws IOException, ServletException
```



86

Eclipse + tomcat

- Add server
- Create Dynamic Web Project
- Create New Servlet
- Run on Server



87

87

Other ways to connect

```
File  Edit  Format  View  Help
yu2031@RUSH-MO8A48ME44:~$ telnet localhost 8080
Trying ::1...
Connected to localhost.
Escape character is '^]'.
GET /HelloDemoM/Hello721 HTTP/1.1
Host: localhost

HTTP/1.1 200
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 132
Date: Thu, 05 Oct 2023 20:24:02 GMT

<html>
<title>Hello 4413!</title>
</head>
<h1>Hello 4413! xxzxsdd</h1>
<h2>Thu Oct 05 16:24:02 EDT 2023</h2>
</body>
</html>
Connection closed by foreign host.
```

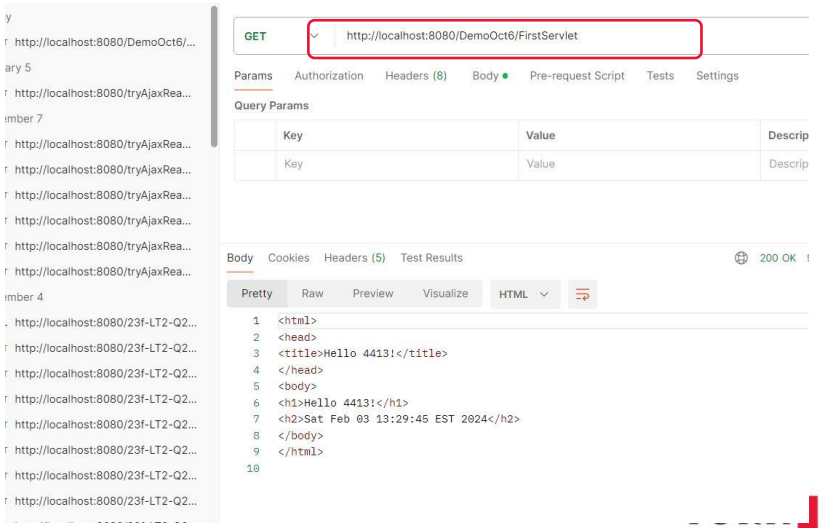
88

88

```
yuyu2031@DESKTOP-MAGTB02:~$ telnet 10.0.0.29 8080
Trying 10.0.0.29...
Connected to 10.0.0.29.
Escape character is '^]'.
GET /DemoOct6/FirstServlet HTTP/1.1
Host: 10.0.0.29

HTTP/1.1 200
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 140
Date: Sat, 03 Feb 2024 18:24:42 GMT

<html>
<head>
<title>Hello 4413!</title>
</head>
<body>
<h1>Hello 4413!</h1>
<h2>Sat Feb 03 13:24:42 EST 2024</h2>
</body>
</html>
Connection closed by foreign host.
yuyu2031@DESKTOP-MAGTB02:~$ curl http://10.0.0.29:8080/DemoOct6/FirstServlet
<html>
<head>
<title>Hello 4413!</title>
</head>
<body>
<h1>Hello 4413!</h1>
<h2>Sat Feb 03 13:29:05 EST 2024</h2>
</body>
</html>
yuyu2031@DESKTOP-MAGTB02:~$
```

89

89



90

90

Reading Parameters

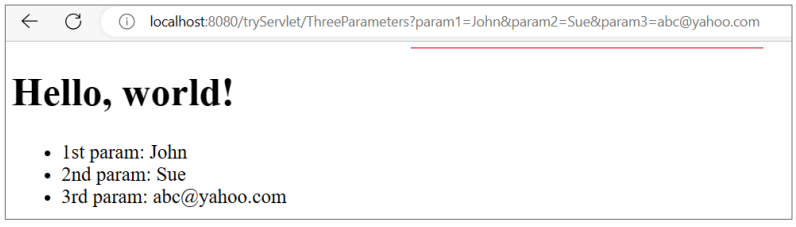
```
@WebServlet("/ThreeParameters")
public class ThreeParams extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    out.println("<html>");
    out.println("<body>");

    out.println("<UL>" +
     "<LI>param1: " + request.getParameter("param1") + "<\LI>" +
     "<LI>param2: " + request.getParameter("param2") + "<\LI>" +
     "<LI>param3: " + request.getParameter("param3") + "<\LI>" +
     "</UL>");

     out.println("</body>");
     out.println("</html>");
     out.close();

 }

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

91

91

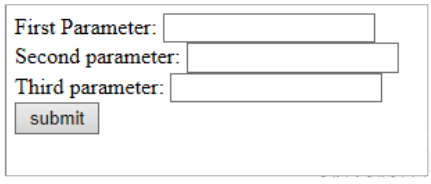
Process HTML Post Form

- From URL or curl



- From FORM in project

```
<form action="ThreeParameters" method= "get">
  First Parameter:  <input type="text" name="param1"><br>
  Second parameter: <input type="text" name="param2"><br>
  Third parameter:  <input type="text" name="param3"><br>

 <input type="submit">

</form>
```



93

93

Process HTML Post Form, with checkbox

```
<form action="HelloForm" method="post">  // default method GET

 Name: <input type="text" name="name" /><br/><br/>
 Email address: <input type="text" name="email_add" /><br/><br/>

 Year:
<select name="year">
   <option value="1st" selected> first
   <option value="2nd">second
   <option value="3rd" > third
   <option value="4th"> fourth

</select><br><br>

Select one:
<input type="checkbox" value='car' name="ice" > car
<input type="checkbox" value='bike' name="ice" > bike
<input type="checkbox" value='boat' name="ice" > boat
<input type="checkbox" value='plane' name="ice" > plane
<br><br>

<input type="submit" value="Submit" />

</form>
```

Name:

Email address:

Year: first

Select one: car bike boat plane

Submit

94

94

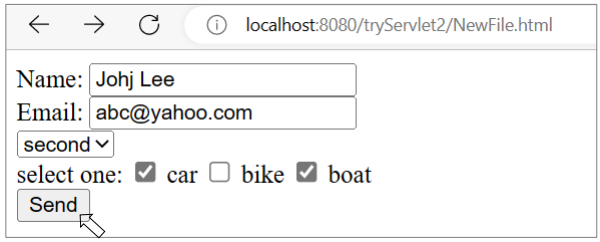
Reading Parameters

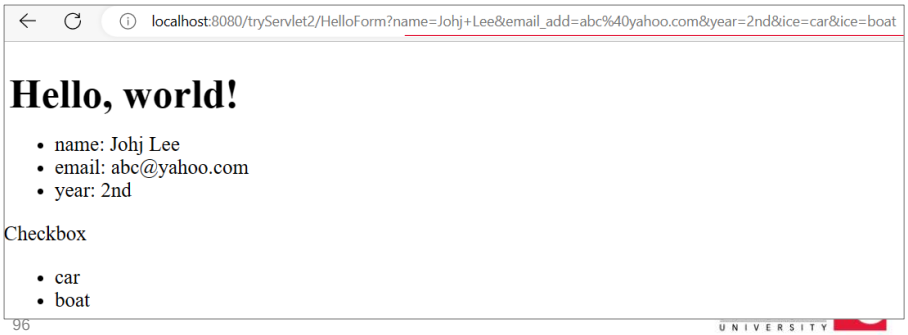
```
public class HelloForm extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

   response.setContentType("text/html");
   PrintWriter out = response.getWriter();

   out.println("<html>");
   out.println("<body>");

   out.println("<h1> Hello Word </h1>" );
   out.println("<ul>" + "<li> name: " + request.getParameter("name") + "</li>");
   out.println("<li> email: " + request.getParameter("email_add") + "</li>");
   out.println("<li> year: " + request.getParameter("year") + "</li>");
   out.println("</ul>");

   out.println(request.getParameter("ice"));
   String a[] = request.getParameterValues("ice");   // an array of values

   out.println("Checkbox input<br> <ul>");
   for(int i=0; i<a.length;i++)
          out.println( "<li> year: " + a[i]+ "</li>");
   out.println("</ul>");

   out.println("</body>");
   out.println("</html>");


}

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

Post?

95

95

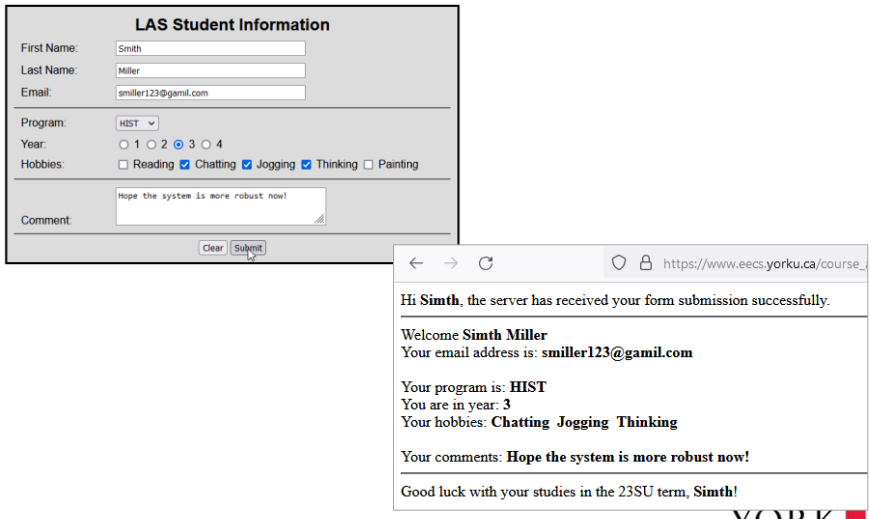




96

96

lab3



97

97

Generating HTML forms

So far, generated list

```
out.println("<h1> Hello Word </h1>" );
   out.println("<ul>" + "<li> name: " + request.getParameter("name") + "</li>");
   out.println("<li> email: " + request.getParameter("email_add") + "</li>");
   out.println("<li> year: " + request.getParameter("year") + "</li>");
   out.println("</ul>");
```

98

98

```
String name = "smith";
String lastN = "Miller";

String[] Program = {"EECS", "CHEM", "CIVL","HIST", "MATH", "ECON"}
String [] Hobbies = {"Reading", "Chatting", "Jogging", "Thinking",

out.println("<body>");
out.println("<h1>Hello, world!</h1>");  // says Hello

out.println("<form method='get' action='order'>");

out.println("Name: ");
out.println("<input type='text' name='name' value='" +name + "' />
out.println("Last Name: <input type='text' name='Lname' value='" +
out.println("Email: ");
out.println("<input type='text' name='cust_phone' /> <br>");
out.println("<hr>");

out.println("Program: ");
out.println("<select name='cust_city' > ");
for (String p: Program) {
    out.println("<option value='" + p+  "' >" + p +"</option>");
}
out.println("</select> <br><br>");

out.println("Hobbies: ");
for (String h: Hobbies) {
    if (h.equals("Thinking"))
      out.println("<input type='checkbox' name='ice' value='" + h+  "' checked='checked' />" + h);
    else out.println("<input type='checkbox' name='ice' value='" + h+  "'  />" + h);
}
out.println("<hr>");

// Submit and reset buttons
out.println("<input type='reset' value='Clear' />");
out.println("<input type='submit' value='Submit' />");
out.println("</form>");
```

```
First Name:
  <input type="text"  name="firstName">
</p>

  <!-- Last Name input -->
Last Name:
<input  type="text" name="lastName">

Email:
<input  type="text" name="email">

<hr>

Program:
<select id="provinceList" name="program">
  <option value="CHEM" >CHEM</option>
  <option value="CIVL" >CIVL</option>
  <option value="EECS" selected>EECS</option>
  <option value="ECON" >ECON</option>
  <option value="HIST" >HIST</option>
  <option value="MATH"> MATH</option>
</select>

Year:
<input type="radio" name="year" value="1" checked="checked" /> 1
<input type="radio" name="year" value="2" /> 2
<input type="radio" name="year" value="3" /> 3
<input type="radio" name="year" value="4" /> 4

Hobbies:
<input type="checkbox" name="hobby" value="Reading"/> Reading
<input type="checkbox" name="hobby" value="Chatting"  /> Chatting
<input type="checkbox" name="hobby" value="Jogging"/>  Jogging
<input type="checkbox" name="hobby" value="Thinking" checked="checked"/
<input type="checkbox" name="hobby" value="Painting"/>  Painting

<hr>

  <input type="reset" value="Clear"  >
  <input type="submit" value="Submit" >

</form>
```

99

```
String name = "smith";
String lastN = "Miller";

String[] Program = {"EECS", "CHEM", "CIVL","HIST", "MATH", "ECO
String [] Hobbies = {"Reading", "Chatting", "Jogging", "Thinkin

out.println("<body>");
out.println("<h1>Hello, world!</h1>");  // says Hello

out.println("<form method='get' action='order'>");

out.println("Name: ");
out.println("<input type='text' name='name' value='" +name + "' /> <br>");
out.println("Last Name: <input type='text' name='Lname' value='" +lastN + "' /> <br>");
out.println("Email: ");
out.println("<input type='text' name='cust_phone' /> <br>");
out.println("<hr>");

out.println("Program: ");
out.println("<select name='cust_city' > ");
for (String p: Program) {
    out.println("<option value='" + p+  "' >" + p +"</option>");
}
out.println("</select> <br><br>");

out.println("Hobbies: ");
for (String h: Hobbies) {
    if (h.equals("Thinking"))
      out.println("<input type='checkbox' name='ice' value='" + h+  "' checked='checked' />" + h);
    else out.println("<input type='checkbox' name='ice' value='" + h+  "'  />" + h);
}
out.println("<hr>");

// Submit and reset buttons
out.println("<input type='reset' value='Clear' />");
out.println("<input type='submit' value='Submit' />");
out.println("</form>");
```

Not secure | 192.168.0.12:8080/tryServlet2/GenerateForm

New Tab | Data Structures & A... | Leisure reading | C Unix | Programmin

Hello, world!

Name: smith
Last Name: Miller
Email:

Program: EECS

Hobbies: ☐ Reading ☐ Chatting ☐ Jogging ☑ Thinking ☐ Painting

Clear | Submit

100

lab3

Not secure | 192.168.0.12:8080/lab3ntu-Eshop0/

New Tab | Data Structures & A... | Leisure reading | C Unix

Yet An e-Bookshop

Choose one or more authors:

☑	Ah Teck
☐	Sue Lee
☑	Joe Suh

Search Books

Not secure | 192.168.0.12:8080/lab3ntu-Eshop0/QueryServlet?auth

New Tab | Data Structures & A... | Leisure reading | C Unix | Programming (J

Query Results

	Book ID	AUTHOR	TITLE	PRICE
☑	3	Tan Ah Teck	Java for Dummy	$22.0
☐	4	Tan Ah Teck	More Java for Dummies	$42.0
☑	5	Joe Suh	Good Java Style	$12.0

Enter your Name: Joseph Su
Enter your Phone Number: 7256300
Choose your City: Montreal

ORDER | CLEAR

Back to Select Menu

Not secure | 192.168.0.12:8080/lab3ntu-Eshop0/order?cust_name=Jo

New Tab | Data Structures & A... | Leisure reading | C Unix | Programming (JAVA

Thank you for ordering.

We cannot show items in this lab. Will do later..

- Customer Name: Joseph Su
- Customer Phone Number: 7256300
- Customer City: Montreal

Back to Select Menu

101

101

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {


  // get author selections from index.html, and search for the books, get a arraylist of books
   .....

  // now generating dynamic page -- form and a table in form
  response.setContentType("text/html;charset=UTF-8");
  PrintWriter out = response.getWriter();

  out.println("<html><head><title>Query Results</title></head><body>");
  out.println("<h2> Query Results</h2>");


  // Print the result in an HTML form inside a table
  out.println("<form method='get' action='...your second servlet url ...'>");

  // table inside form, table header
  out.println("<table border='1' cellpadding='6'>");
  out.println("<tr>");
  out.println("<th> </th>");
  out.println("<th>Book ID</th>");
  out.println("<th>AUTHOR</th>");
  out.println("<th>TITLE</th>");
  out.println("<th>PRICE</th>");
  out.println("</tr>");

  // print each row of table
  for(Book e: resulist) {
    // Print each row with a checkbox identified by book's id
    out.println("<tr>");

    out.println("<td><input type='checkbox'   /> </td>");  // checkbox, not used in this lab but later
    out.println("<td>" + e.getId() + "</td>");
   // print author name, title, price .....

    out.println("</tr>");
  }
  out.println("</table><br />"); // end table
  // print name phone textbox,
  out.println("Enter your Name: ");
  out.println("<input .... name='cust_name' <br>");

      ....

  out.println("Choose your City: ");
  // print dropdown list for Toronto, Montreal, Vancouver
  out.println("<select name='cust_city' > ");
  ....
  out.println("</select> <br><br>");


  // Submit and reset buttons
      ....

  out.println("</form>");  // end form

  // Hyperlink to go back to search menu
```

102

102