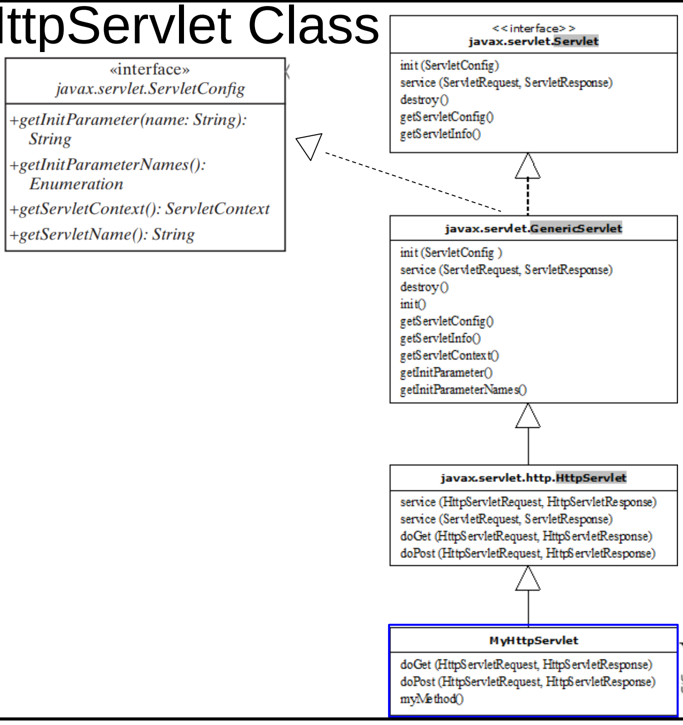
More on Servlet

- **Http related header info**
- Initialization -- context vs config scope
- Request Dispatcher
- Session tracking
- More on annotations and web.xml

97

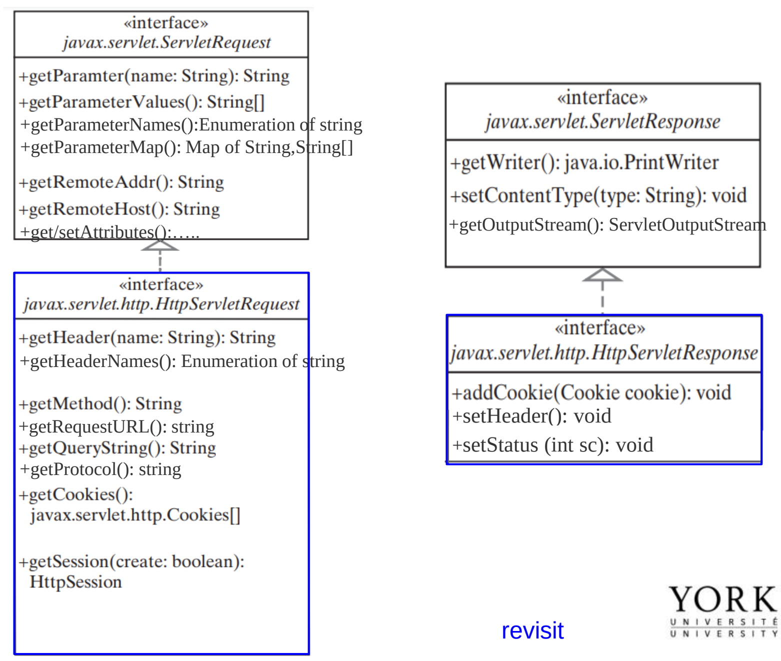
97

HttpServlet Class


«interface»
javax.servlet.ServletConfig
+getInitParameter(name: String): String
+getInitParameterNames(): Enumeration
+getServletContext(): ServletContext
+getServletName(): String
<<interface>>
javax.servlet.Servlet
init (ServletConfig)
service (ServletRequest, ServletResponse)
destroy()
getServletConfig()
getServletInfo()
javax.servlet.GenericServlet
init (ServletConfig)
service (ServletRequest, ServletResponse)
destroy()
init()
getServletConfig()
getServletInfo()
getServletContext()
getInitParameter()
getInitParameterNames()
javax.servlet.http.HttpServlet
service (HttpServletRequest, HttpServletResponse)
service (ServletRequest, ServletResponse)
doGet (HttpServletRequest, HttpServletResponse)
doPost (HttpServletRequest, HttpServletResponse)
MyHttpServlet
doGet (HttpServletRequest, HttpServletResponse)
doPost (HttpServletRequest, HttpServletResponse)
myMethod()

98

98


«interface»
javax.servlet.ServletRequest
+getParamter(name: String): String
+getParameterValues(): String[]
+getParameterNames():Enumeration of string
+getParameterMap(): Map of String,String[]
+getRemoteAddr(): String
+getRemoteHost(): String
+get/setAttributes():.....
«interface»
javax.servlet.http.HttpServletRequest
+getHeader(name: String): String
+getHeaderNames(): Enumeration of string
+getMethod(): String
+getRequestURL(): string
+getQueryString(): String
+getProtocol(): string
+getCookies(): javax.servlet.http.Cookies[]
+getSession(create: boolean): HttpSession
«interface»
javax.servlet.ServletResponse
+getWriter(): java.io.PrintWriter
+setContentType(type: String): void
+getOutputStream(): ServletOutputStream
«interface»
javax.servlet.http.HttpServletResponse
+addCookie(Cookie cookie): void
+setHeader(): void
+setStatus (int sc): void

revisit

101

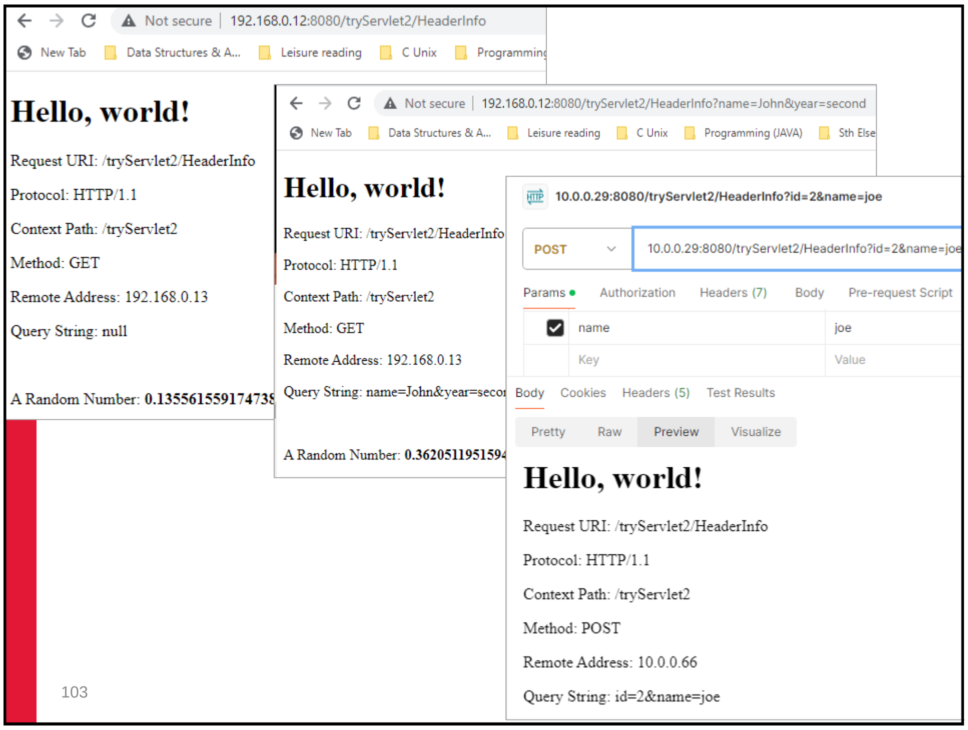
101

Get header info

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

      // Set the response MIME type of the response message
      response.setContentType("text/html");
    // Allocate a output writer to write response message into the network socket
      PrintWriter out = response.getWriter();

      // Write the response message, in an HTML page
      out.println("<!DOCTYPE html>");
      out.println("<html>");
      out.println("<head><title>Hello, World</title></head>");
      out.println("<body>");
      out.println("<h1>Hello, world!</h1>");  // says Hello
      // Echo client's request information
      out.println("<p>Request URI: " + request.getRequestURI() + "</p>");
      out.println("<p>Protocol: " + request.getProtocol() + "</p>");
      out.println("<p>Context Path: " + request.getContextPath() + "</p>");
      out.println("<p>Method: " + request.getMethod() + "</p>");
      out.println("<p>Remote Address: " + request.getRemoteAddr() + "</p>");
      out.println("<p>Query String: " + request.getQueryString() + "</p>  <br>");
      // Generate a random number upon each request
      out.println(<p>A Random Number: <strong>" + Math.random() + "</strong></p>");

      out.println("</body></html>");
      out.close();  // Always close the output writer
```

102

102



Not secure | 192.168.0.12:8080/tryServlet2/HeaderInfo

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

Hello, world!

Request URI: /tryServlet2/HeaderInfo

Protocol: HTTP/1.1

Context Path: /tryServlet2

Method: GET

Remote Address: 192.168.0.13

Query String: null

A Random Number: **0.135561559174738**

Not secure | 192.168.0.12:8080/tryServlet2/HeaderInfo?name=John&year=second

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

Hello, world!

Request URI: /tryServlet2/HeaderInfo

Protocol: HTTP/1.1

Context Path: /tryServlet2

Method: GET

Remote Address: 192.168.0.13

Query String: name=John&year=secon

A Random Number: **0.3620511951594**

10.0.0.29:8080/tryServlet2/HeaderInfo?id=2&name=joe

POST | 10.0.0.29:8080/tryServlet2/HeaderInfo?id=2&name=joe

Params ● | Authorization | Headers (7) | Body | Pre-request Script

	Key	Value
✓	name	joe
	Key	Value

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize

Hello, world!

Request URI: /tryServlet2/HeaderInfo

Protocol: HTTP/1.1

Context Path: /tryServlet2

Method: POST

Remote Address: 10.0.0.66

Query String: id=2&name=joe

103

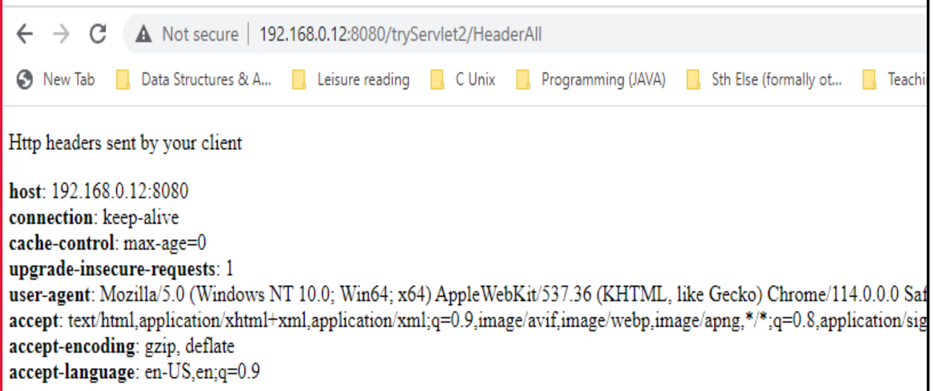
103

Get all header info

```
public class ShowHeaders extends HttpServlet {

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws IOException, ServletException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Request's HTTP Headers</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<p>HTTP headers sent by your client:</p>");
      Enumeration enum = request.getHeaderNames();
      while (enum.hasMoreElements()) {
        String headerName = (String) enum.nextElement();
        String headerValue = request.getHeader(headerName);
        out.print("<b>"+headerName + "</b>: ");
        out.println(headerValue + "<br>");
      }
      out.println("</body>");
      out.println("</html>");
  }
}
```

104

104



```
192.168.0.12:8080/tryServlet2/HeaderAll

Http headers sent by your client

host: 192.168.0.12:8080
connection: keep-alive
cache-control: max-age=0
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Saf
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
accept-encoding: gzip, deflate
accept-language: en-US,en;q=0.9
```

105

105

More on Servlet

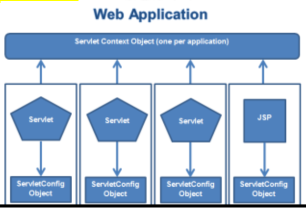
- Http related header info
- **Initialization -- context vs config scope**
- Request Dispatcher
- Session tracking
- More on annotations and web.xml

107

107

`ServletContext` and `ServletConfig`

- `ServletContext` and `ServletConfig` these two are the important interfaces of Servlet API which are used during web application development
- Both `ServletContext` and `ServletConfig` are basically configuration objects which are used by the servlet container to initialize various parameters of web applications.
- But they have some the difference in terms of **scope** and **availability**
- The most important difference is that `ServletContext` is per web application while `ServletConfig` is per servlet basis.



108

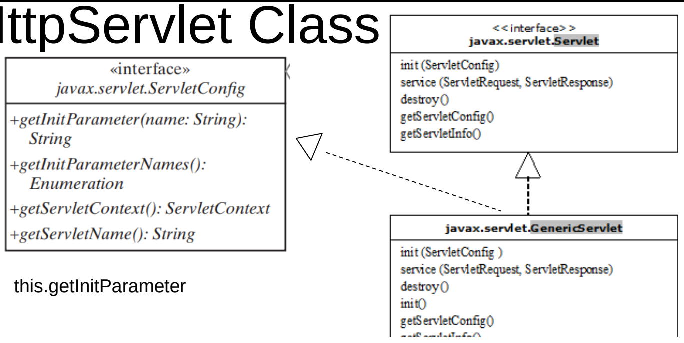
Servlet Config

- `ServletConfig` is an interface in Servlet API and `ServletConfig` object represents or is used to initialize a single servlet in web application by the servlet container.
- When the Web Container initializes a servlet, it creates a `ServletConfig` object for the servlet. `ServletConfig` object is used to pass information to a servlet during initialization by getting configuration information from **web.xml** (Deployment Descriptor).
- Frequently used methods of `ServletConfig` interface.
  - **public String getInitParameter(String param)**: It returns the value of given parameter or null if the parameter doesn't exist.
  - **public Enumeration getInitParameterNames()**: Returns an enumeration of the servlet's parameters as String, Object. Or an empty Enumeration if the servlet has no initialized parameters.
  - **public void getServletContext()**: returns a reference to the ServletContext.

YORK UNIVERSITÉ UNIVERSITY

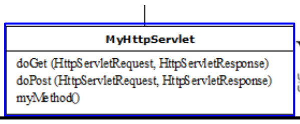
109

HttpServlet Class


«interface»
javax.servlet.ServletConfig
+getInitParameter(name: String): String
+getInitParameterNames(): Enumeration
+getServletContext(): ServletContext
+getServletName(): String
<<interface>>
javax.servlet.Servlet
init (ServletConfig)
service (ServletRequest, ServletResponse)
destroy()
getServletConfig()
getServletInfo()
javax.servlet.GenericServlet
init (ServletConfig)
service (ServletRequest, ServletResponse)
destroy()
init()
getServletConfig()

this.getInitParameter

- Frequently used methods of `ServletConfig` interface.
  - **public String getInitParameter(String param)**: It returns the value of given parameter or null if the parameter doesn't exist.
  - **public Enumeration getInitParameterNames()**: Returns an enumeration of the servlet's parameters as String, Object. Or an empty Enumeration if the servlet has no initialized parameters.
  - **public void getServletContext()**: returns a reference to the ServletContext.


MyHttpServlet
doGet (HttpServletRequest, HttpServletResponse)
doPost (HttpServletRequest, HttpServletResponse)
myMethod()

YORK U UNIVERSITÉ UNIVERSITY

110

110

```
<servlet>
 <init-param>
   <param-name>name</param-name>
   <param-value>value</param-value>
 </init-param>
</servlet>
….
```

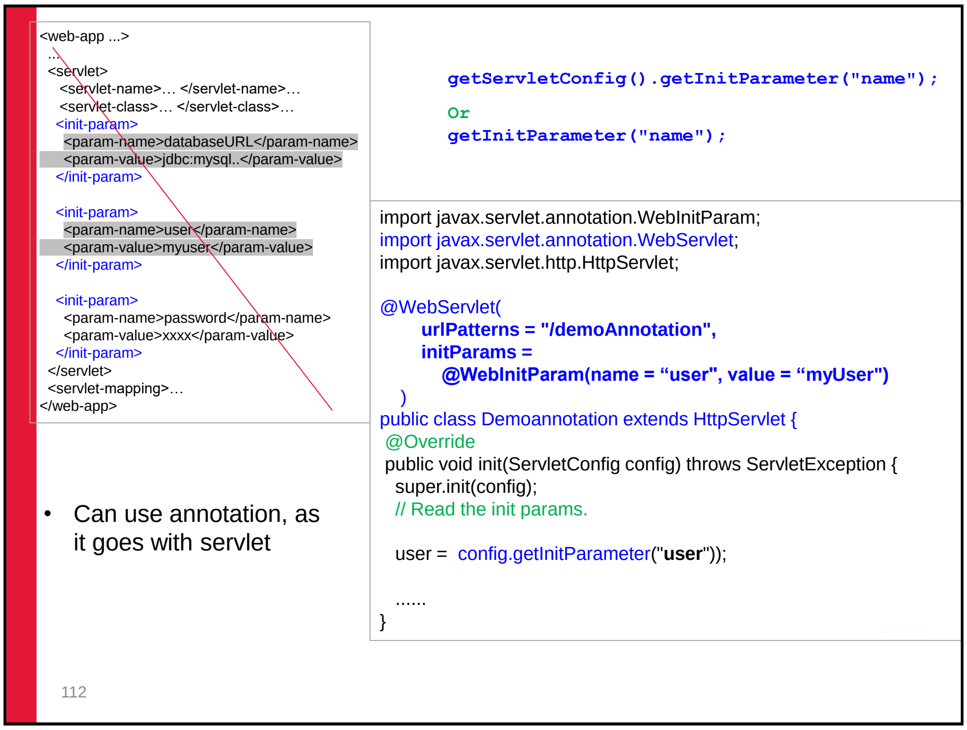
```
getServletConfig().getInitParameter("name");
or
getInitParameter("name");
```

```
<web-app ...>
  ...
  <servlet>
      <servlet-name>… </servlet-name>…
      <servlet-class>… </servlet-class>…
       <init-param>
         <param-name>databaseURL</param-name>
         <param-value>jdbc:mysql://localhost:3306/ae</param-value>
       </init-param>

       <init-param>
         <param-name>user</param-name>
         <param-value>myuser</param-value>
       </init-param>

       <init-param>
         <param-name>password</param-n
         <param-value>xxxx</param-value>
       </init-param>
 </servlet>
  <servlet-mapping>…
</web-app>
```

```
@Override
public void init(ServletConfig config) throws ServletException {
   super.init(config);
   // Read the init params.
   dbURL = config.getInitParameter("databaseURL");
   user =     config.getInitParameter("user");
   passw=  config.getInitParameter("password");
   ……
}
```

111

111



```
<web-app ...>
 ...
 <servlet>
    <servlet-name>… </servlet-name>…
    <servlet-class>… </servlet-class>…
   <init-param>
     <param-name>databaseURL</param-name>
     <param-value>jdbc:mysql..</param-value>
   </init-param>

   <init-param>
     <param-name>user</param-name>
     <param-value>myuser</param-value>
   </init-param>

   <init-param>
     <param-name>password</param-name>
     <param-value>xxxx</param-value>
   </init-param>
 </servlet>
 <servlet-mapping>…
</web-app>
```

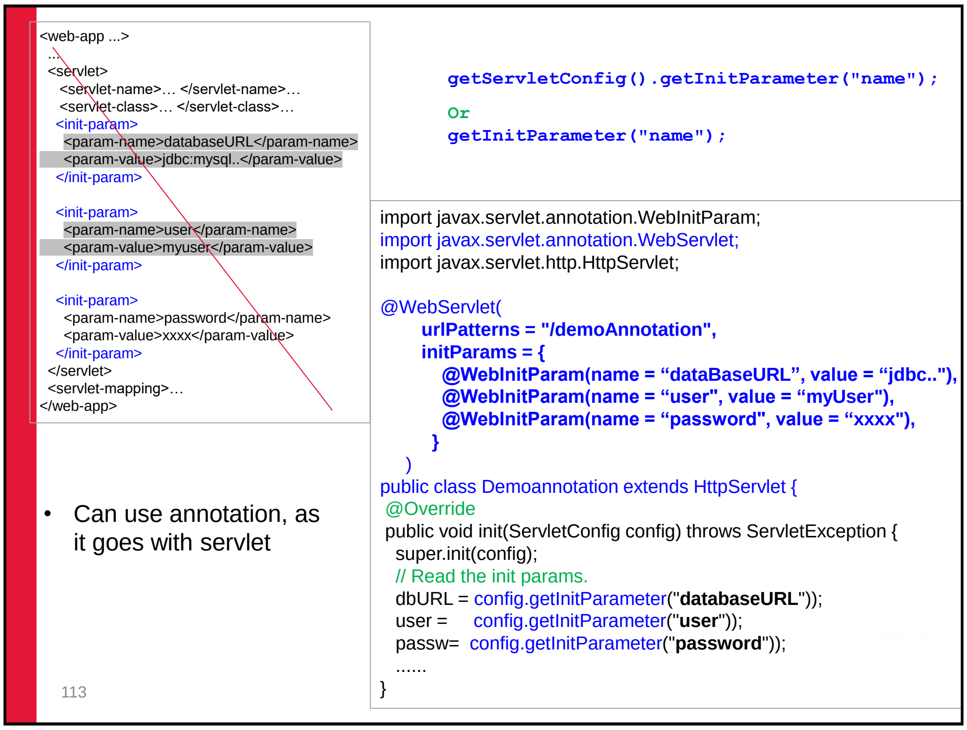
```
getServletConfig().getInitParameter("name");
Or
getInitParameter("name");
```

```
import javax.servlet.annotation.WebInitParam;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;

@WebServlet(
      urlPatterns = "/demoAnnotation",
      initParams =
         @WebInitParam(name = “user", value = “myUser")
  )
public class Demoannotation extends HttpServlet {
 @Override
 public void init(ServletConfig config) throws ServletException {
   super.init(config);
   // Read the init params.

   user =  config.getInitParameter("user"));

   ......
}
```

- Can use annotation, as it goes with servlet

112

112



```
<web-app ...>
 ...
 <servlet>
    <servlet-name>… </servlet-name>…
    <servlet-class>… </servlet-class>…
   <init-param>
     <param-name>databaseURL</param-name>
     <param-value>jdbc:mysql..</param-value>
   </init-param>

   <init-param>
     <param-name>user</param-name>
     <param-value>myuser</param-value>
   </init-param>

   <init-param>
     <param-name>password</param-name>
     <param-value>xxxx</param-value>
   </init-param>
 </servlet>
 <servlet-mapping>…
</web-app>
```

```
getServletConfig().getInitParameter("name");
Or
getInitParameter("name");
```

```
import javax.servlet.annotation.WebInitParam;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;

@WebServlet(
      urlPatterns = "/demoAnnotation",
      initParams = {
         @WebInitParam(name = “dataBaseURL”, value = “jdbc..”),
         @WebInitParam(name = “user", value = “myUser"),
         @WebInitParam(name = “password", value = “xxxx"),
       }
   )
public class Demoannotation extends HttpServlet {
 @Override
 public void init(ServletConfig config) throws ServletException {
   super.init(config);
   // Read the init params.
   dbURL = config.getInitParameter("databaseURL"));
   user =     config.getInitParameter("user"));
   passw=  config.getInitParameter("password"));
   ......
}
```

- Can use annotation, as it goes with servlet

113

113

ServletContext

- Each webapp is represented in a single context within the servlet container (such as Tomcat). In Servlet API, this context is defined in `javax.servlet.ServletContext` interface). A webapp may use many servlets. This object is common for all the servlets in this webapp. Servlets deployed in the same webapp can share information between them using the shared ServletContext object.
- There is one `ServletContext` per webapp (or web context). It can be retrieved via `ServletConfig.getServletContext()`. A servlet can use it to communicate with its servlet container, other Servlets and JSP in the application.
- `ServletContext` has an "application" scope, and can also be used to pass information between servlets and JSPs within the same application,
  - via methods `setAttribute`("name", object) and `getAttribute`("name").
- Frequently used methods of **ServletContext** interface.
  - **public String getInitParameter(String param)**: It returns the value of given parameter or null if the parameter doesn't exist.
  - **public Enumeration getInitParameterNames()**: Returns an enumeration of context parameters names.
  - **public void setAttribute(String name,Object object)**: Sets the attribute value for the given attribute name.
  - **public Object getAttribute(String name)**:Returns the attribute value for the given name or null if the attribute doesn't exist.

114

114

```
<context-param>
   <param-name>name</param-name>
   <param-value>value</param-value>
</context-param>
```

```
getServletContext().getInitParameter("name");
```
Or,
```
getServeltConfig().getServletContext().
                   getInitParameter("name");
```

```
<web-app ...>
   ...
  <servlet>
    ...
  </servlet>
   ...
 <context-param>
       <param-name> user  </param-name>
       <param-vlaue> admin  </param-value>
 </context-param>

 <context-param>
       <param-name> adminEmail </param-name>
       <param-vlaue> admin@wrox.com  </param-value>
 </context-param>

 ….
</web-app>
```

```
@Override
public void init(ServletConfig config) throws ServletException {
   super.init(config);
   // Read the init params.
   usr = getServletContext().getInitParameter("user");
   email = getServletContext().getInitParameter("adminEmail");
     ......
}
```
Or in doGet()

- Cannot use annotation (in servlet class), as it does not go with servlet
- Must use **web.xml**

115

115

```
@Override
public void init(ServletConfig config) throws ServletException {
   super.init(config);
   // Read the init params for this servlet.
   dbURL = "databaseURL";
   user =     "user";

   // set in web context for use by servlets JSPs within this web app
   ServletContext sContext = config.getServletContext();
   sContext.setAttribute("universal_dbURL", dbURL);
   sContext.setAttribute("universal_User ,user);



   ......
}
```

set/get attributes
Context level example

Share with other servlets in the application

Other ways, revisit

116

116

Servlet Config	Servlet Context
Servlet config object represent single servlet	Servlet context object represents the whole web application running on a particular JVM and common for all the servlet
It's like a local parameter associated with a particular servlet	It's like a global parameter associated with the whole application
It's a name-value pair defined inside the servlet section of `web.xml` file so it has servlet wide scope	`ServletContext` has application-wide scope so define outside of servlet tag in web.xml file.
`getServletConfig()` method is used to get the config object	`getServletContext()` method is used to get the context object. or `getServletConfig().getServletcontext()`
Object of `ServletConfig` will be created during the initialization process of the servlet.	Object of `ServletContext` will be created at the time of web application deployment
Scope: As long as a servlet is executing, the `ServletConfig` object will be available, it will be destroyed once the servlet execution is completed	Scope: As long as a web application is executing, the `ServletContext` object will be available, and it will be destroyed once the application is removed from the server

117

117

More on Servlet

- Http related header info
- Initialization -- context vs config scope
- **Request Dispatcher**
- Session tracking
- More on annotations and web.xml

119

119

What is the purpose

- Forward a request from one servlet to another servlet (or JSP).
- Have first servlet do some of the work and then pass on to another.
- Can even forward on to a static source like html

120

120

The Request Dispatcher

- The **`RequestDispatcher`** object is used to send a client request to any resource on the server
- Such a resource may be dynamic (e.g. a Servlet or a JSP file) or static (e.g. a HTML document)
- **`RequestDispatcher`** object can be obtained either
  - by **`request`** object (means the dispatch is relative to the current URL)

```
RequestDispatcher rd = request.getRequestDispatcher("xyz")
```

  - by **`ServletContext`** Object – (means the dispatch is relative to the root of the ServletContext). / is needed

```
RequestDispatcher rd =
        getServletContext().getRequestDispatcher("/xyz")
```

- Dispatcher methods
  - **`forward(request,response)`** forwards the request from one servlet to another resource (such as servlet, JSP, HTML file).
  - **`include(request,response)`** includes the content of the resource(such as servlet, JSP, HTML file) in the response.

122

122

- Dispatcher methods

`void forward(ServletRequest request, ServletResponse response)`

Forwards a request from a Servlet to another resource (such as servlet, JSP, HTML file)

`void include(ServletRequest request, ServletResponse response)`

Includes the content of a resource(such as servlet, JSP, HTML file) in the current response

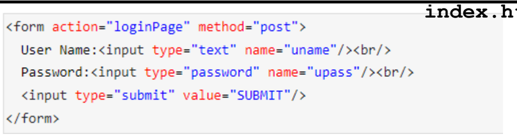
```
RequestDispatcher rd = request.getRequestDispatcher("xyz");
rd.forward(request, response);
// or
request.getRequestDispatcher("xyz").forward(request, response);

// or
RequestDispatcher rd= getServletContext().getRequestDispatcher("/xyz");
rd.forward(request, response);

getServletContext().getRequestDispatcher("/xyz").forward(req, res);
```

Usually set attributes before forwarding

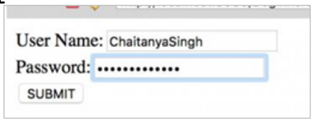
124

124

index.html



```
<form action="loginPage" method="post">
  User Name:<input type="text" name="uname"/><br/>
  Password:<input type="password" name="upass"/><br/>
  <input type="submit" value="SUBMIT"/>
</form>
```



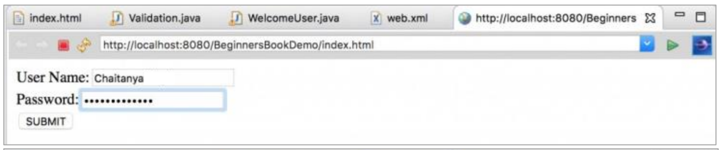
User Name: ChaitanyaSingh
Password: •••••••••••••
SUBMIT

```
@WebServlet("/loginPage")
public class Validation extends HttpServlet
{
   public void doPost(HttpServletRequest request,
     HttpServletResponse response)
       throws ServletException, IOException
   {
       response.setContentType("text/html");
       PrintWriter pwriter = response.getWriter();
       String name=request.getParameter("uname");
       String pass=request.getParameter("upass");
       if(name.equals("Chaitanya") && pass.equals("beginnersbook"))
       {
          RequestDispatcher dis=request.getRequestDispatcher("welcome");
          dis.forward(request, response);
       }
       else
       {
          pwriter.print("User name or password is incorrect!");

          RequestDispatcher dis=request.getRequestDispatcher("index.html");
          dis.include(request, response);
       }
   }
```

126

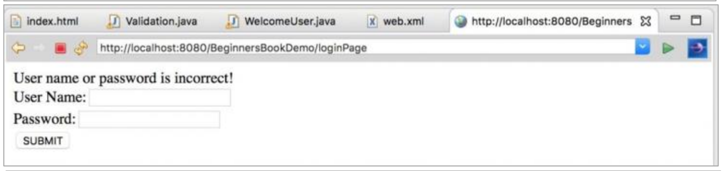
126

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

 public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
 {
    response.setContentType("text/html");
    PrintWriter pwriter = response.getWriter();
    writer.println("<html><body>");

    String name= request.getParameter("uname"); // get from forwarded request
    pwriter.print("Hello " + name + "!");
    pwriter.print(" Welcome to Beginnersbook.com");
    ......
  }
}
```



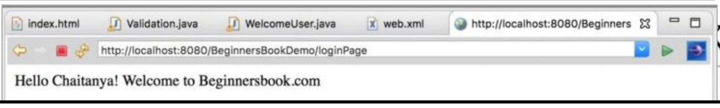
http://localhost:8080/BeginnersBookDemo/index.html

User Name: Chaitanya
Password: •••••••••••••
SUBMIT



http://localhost:8080/BeginnersBookDemo/loginPage

User name or password is incorrect!
User Name:
Password:
SUBMIT



http://localhost:8080/BeginnersBookDemo/loginPage

Hello Chaitanya! Welcome to Beginnersbook.com

127

127

Passing on Data

- Usually need to manipulate data and then forward.

- 3 different ways/levels to pass data for the forwarded Servlet or JSP `setAttribute(String, Obj), getAttribute(String)`

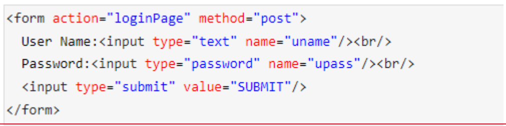
  - Data that will be used only for this request:
    ```
    request.setAttribute("key", value);
    request.getAttribute("key");
    ```

  - Data will be used for this client (also for future requests):
    ```
    session.setAttribute("key", value);
    session.getAttribute("key");
    ```

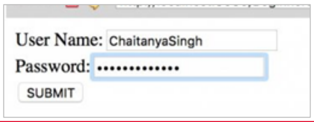
  - Data that will be used in the future for every client
    ```
    context.setAttribute("key", value);
    context.getAttribute("key");
    ```

128

128



```
<form action="loginPage" method="post">
  User Name:<input type="text" name="uname"/><br/>
  Password:<input type="password" name="upass"/><br/>
  <input type="submit" value="SUBMIT"/>
</form>
```



User Name: ChaitanyaSingh

Password: •••••••••••••

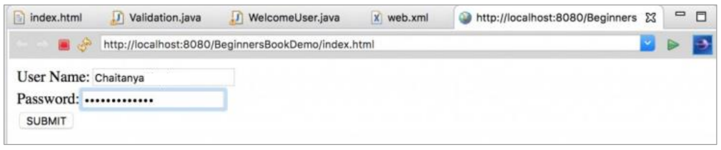
```
@WebServlet("/loginPage")
public class Validation extends HttpServlet
{
   public void doPost(HttpServletRequest request,
     HttpServletResponse response)
        throws ServletException, IOException
   {
        response.setContentType("text/html");
        PrintWriter pwriter = response.getWriter();
        String name=request.getParameter("uname");
        String pass=request.getParameter("upass");
        if(name.equals("Chaitanya") && pass.equals("beginnersbook"))
        {
           request.setAttribute("upperName", name.toUpperCase());

           RequestDispatcher dis=request.getRequestDispatcher("welcome");
           dis.forward(request, response);
        }
        else
        {
           pwriter.print("User name or password is incorrect!");

           RequestDispatcher dis=request.getRequestDispatcher("index.html");
           dis.include(request, response);
```

Revisit the forward program

135

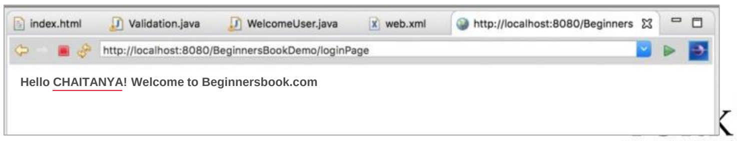
135

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

 public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
 {
    response.setContentType("text/html");
    PrintWriter pwriter = response.getWriter();
    writer.println("<html><body>");

    String name= (String)request.getAttribute("upperName");

    pwriter.print("<p>Hello " + name + "!");
    pwriter.print(" Welcome to Beginnersbook.com  </p>");
    pwriter.println("<html><body>");
  }
}
```



http://localhost:8080/BeginnersBookDemo/index.html

User Name: Chaitanya
Password: ••••••••••••••



http://localhost:8080/BeginnersBookDemo/loginPage

Hello CHAITANYA! Welcome to Beginnersbook.com

136

For entered name, can still setAttribute, getAttribute, but not needed here

136

```
<servlet>
 <init-param>
   <param-name>name</param-name>
   <param-value>value</param-value>
</init-param>
</servlet>
….
```

```
getServletConfig().getInitParameter("name");

Or
getInitParameter("name");
```

```
<web-app ...>
  ...
  <servlet>
     <servlet-name>… </servlet-name>…
     <servlet-class>… </servlet-class>…
  <init-param>
    <param-name>databaseURL</param-name>
    <param-value>jdbc:mysql://localhost:3306/ae</param-value>
  </init-param>

  <init-param>
    <param-name>user</param-name>
    <param-value>myuser</param-value>
  </init-param>

  <init-param>
    <param-name>password</param-name
    <param-value>xxxx</param-value>
  </init-param>
 </servlet>
  <servlet-mapping>…
</web-app>
```

Revisit the config parameter example

set/get attributes
Context level example

```
@Override
public void init(ServletConfig config) throws ServletException {
   super.init(config);
   // Read the init params for this servlet.
   dbURL = config.getInitParameter("databaseURL"));
   user =    config.getInitParameter("user"));
   passw=  config.getInitParameter("password"));
   // set in web context for use by servlets JSPs within this web ap
   ServletContext sContex = config.getContext();
   sContext.setAttribute("universal_dbURL", dbURL);
   sContext.setAttribute("universal_User ,user);
   sContext.setAttribute("universal_passwd" , passwr);
```

137

Forward or not (e.g. hyperlink, etc)

137

Forwarding versus Redirection

- The **sendRedirect()** method of **HttpServletResponse** interface can (also) be used to redirect response to another resource, it may be servlet, jsp or html file.
- It accepts relative as well as absolute URL.
- It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.

```
response.sendRedirect("anotherServlet");
```

- By default, **SendRedirect** does not preserve parameters of the request
- **SendRedirect** ends up with a different URL on the client

138

138

More on Servlet

- Http related header info
- Initialization -- context vs config scope
- Request Dispatcher
- **Session tracking**
- More on annotations and web.xml

139

139

HTTP - Stateless protocol (1)

- The Web server can’t remember previous transactions since HTTP is a “stateless” protocol.
- Cannot associate requests from a client, treating each request independently.
- A **session** can be defined as a series of related interactions between a single client and the Web server over a period of time.
- Tracking data among requests in a session is known as session tracking.

- E.g., a shopping cart:
  - it contains all items that have been selected from an online store's catalog by a customer;
  - the customer can check the content of the cart at any time during the session;
  - thus, the server must be able to maintain the cart of the user across several Web page requests – same ‘browser’

HTTP - Stateless protocol (2)

There are four ways to maintain the state:

- Hidden fields in forms
  - `<input type="hidden" value="<valore>" />`

- URL rewriting
  - e.g., `http://host/serveltPages/ShowSession;jsessionid=E4D371710`
  - String `encodeURL(String url)` of the class `HttpServletResponse`
  - where: url is the original l’URL and the output is the rewritten one

- Cookies
  - `javax.servlet.http.Cookie(String name, String val)`
  - methods: `setName(String cookieName), setValue(String cookieValue)`, etc.

- **Servlets (HttpSession API)**

Hidden fields (brief)

- Can track a session by passing data from the servlet to the client as hidden values in a dynamically generated HTML form
- With a field like this

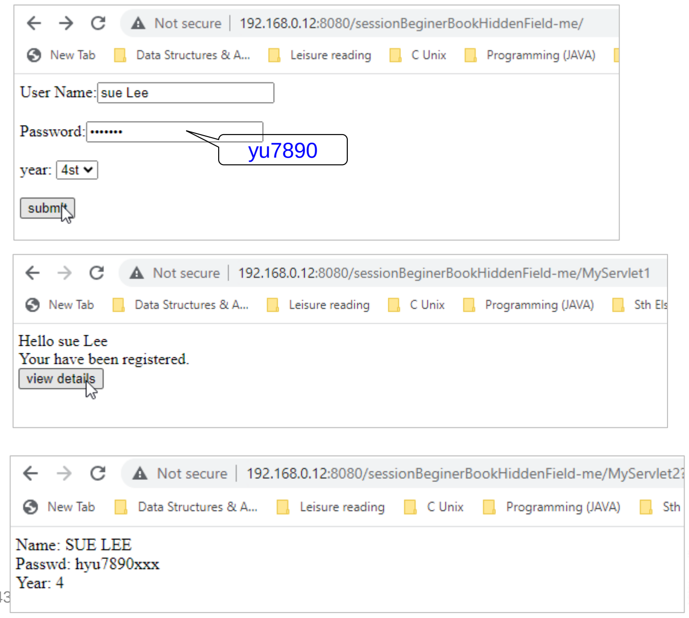
  `<input type="hidden" name= "lastname" value="smith">`

- When the form is submitted, the servlet receives the hidden value just like a regular parameter value,
  - using **`getParameter("lastname")`**

142

142

An example



Not secure | 192.168.0.12:8080/sessionBeginerBookHiddenField-me/

User Name: sue Lee

Password: •••••••

yu7890

year: 4st

submit

Not secure | 192.168.0.12:8080/sessionBeginerBookHiddenField-me/MyServlet1

Hello sue Lee
Your have been registered.

view details

Not secure | 192.168.0.12:8080/sessionBeginerBookHiddenField-me/MyServlet2

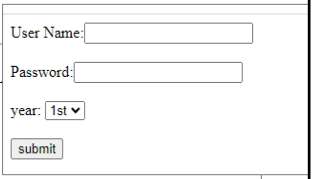
Name: SUE LEE
Passwd: hyu7890xxx
Year: 4

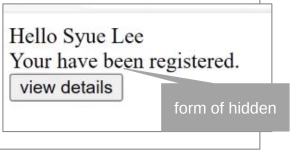
143

143

An example

```
<form action="MyServlet1" method="post">
  User Name:<input type="text" name="userName"/><br/>
  Password:<input type="password">
  name="userPassword"/><br/><br>
  year: <select name="year">
          <option value="1"> 1st
          <option value="2"> 2st
          <option value="3"> 3st
          <option value="4"> 4st
         </select>
  <br><br>
  <input type="submit" value="submit"/>
</form>
```




Hello Syue Lee
Your have been registered.

MyServlet1

```
      response.setContentType("text/html");
      PrintWriter pwriter = response.getWriter();
      String name = request.getParameter("userName").toUpperCase();
      String password = request.getParameter("userPassword")+"XXXX";
      String year = request.getParameter("year");

      pwriter.println("Hello "+name +"<br>");
      pwriter.println("You have been registered.");
      // hidden field  //must be form?!
      pwriter.println("<form method='get' action='MyServlet2' />");
      pwriter.println("<input type='hidden' name='unameH' value='" + name+ "' />"
      pwriter.println("<input type='hidden' name='upassH' value='" + password+"
..
      pwriter.println("<INPUT TYPE='SUBMIT'  VALUE='view details' '/> ");
      pwriter.println("</form>"
```

144

MyServlet2

- Just retrieve as regular form parameters

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

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

      String myName = request.getParameter("uname");
      String myPass= request.getParameter("upass");
      String myYear= request.getParameter("year");

      pwriter.println("Name: "+myName+"<br>");
      pwriter.println("Passwd: "+ myPass+"<br>");
      pwriter.println("Year: "+ myYear+"<br>");
      pwriter.close();
}
```

145

Cookies (brief)

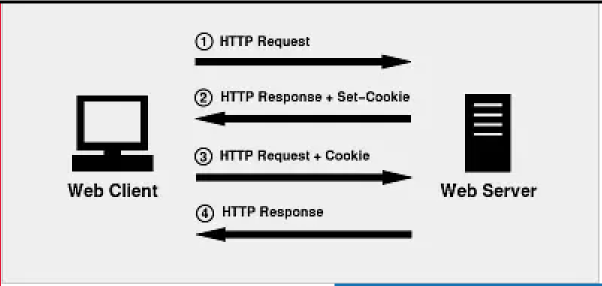
- Cookies are the textual information sent by server and that is stored in key-value pair format to the client's browser (disk) during multiple requests.
- It is one of the state management techniques in session tracking
- When the client generates a request, the server gives the response with cookies having an id which are then stored in the client's browser.
- Thus if the client generates a second request, a cookie with the matched id is also sent to the server. The server will fetch the cookie id, if found it will treat it as an old request otherwise the request is considered new.

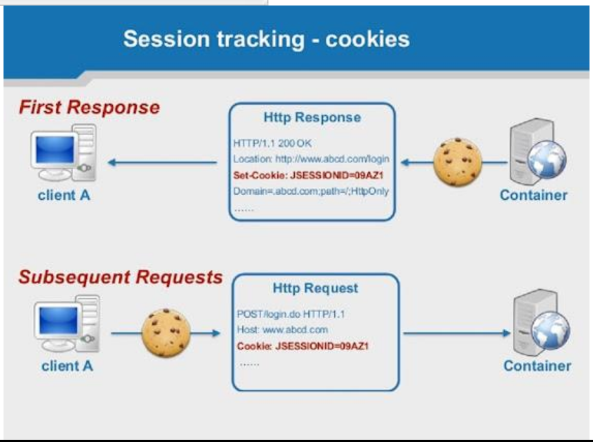
Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.
   - For object, store field by field, multiple cookies

146

146





```
HTTP/1.1 200 OK
Location: http://www.abcd.com/login
Set-Cookie: JSESSIONID=09AZ1
Domain=.abcd.com;path=/;HttpOnly
......
```

```
POST /login.do HTTP/1.1
Host: www.abcd.com
Cookie: JSESSIONID=09AZ1
......
```

147

- In order to use cookies in java, use a Cookie class that is present in **javax.servlet.http** package.

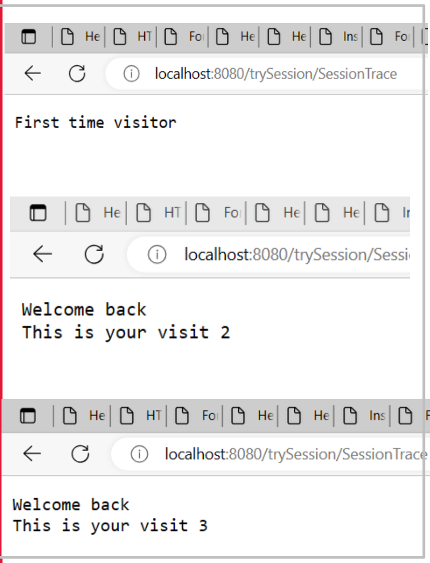
- Steps for sending cookie to the client:
  1) Create a Cookie object:
  ```
  Cookie c = new Cookie("userName","Chaitanya"); // string only
  ```
  2) Set the maximum Age:
  By using setMaxAge () method we can set the maximum age for the particular cookie in seconds. Default 30 minutes
  ```
  c.setMaxAge(1800);
  ```
  3) Place the Cookie in HTTP response header:
  We can send the cookie to the client browser through **response.addCookie()** method.
  ```
  response.addCookie(c);
  ```

- How to read cookies // no way to get specific cookie (by name)
  ```
  Cookie c[] = request.getCookies();
  for(int i=0; i<c.length; i++){
   out.print("Name: "+c[i].getName()+" & Value: "+c[i].getValue());
  }
  ```

148

148

Example: cookie to detect first time visitor


localhost:8080/trySession/SessionTrace

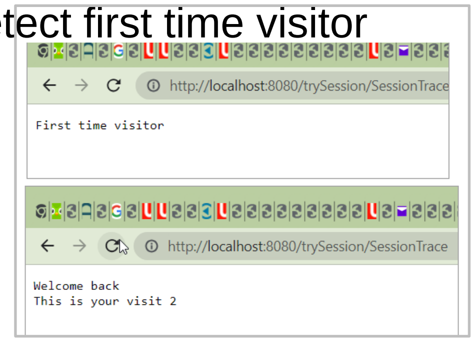
```
First time visitor
```

localhost:8080/trySession/Sessi

```
Welcome back
This is your visit 2
```

localhost:8080/trySession/SessionTrace

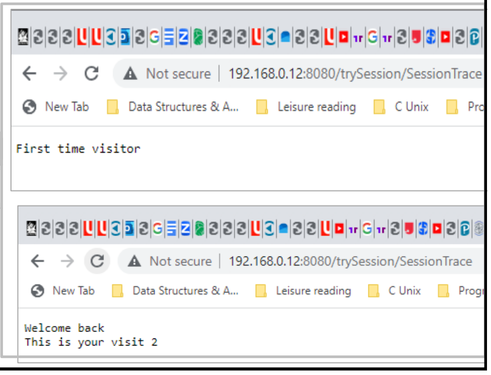
```
Welcome back
This is your visit 3
```


http://localhost:8080/trySession/SessionTrace

```
First time visitor
```

http://localhost:8080/trySession/SessionTrace

```
Welcome back
This is your visit 2
```


Not secure | 192.168.0.12:8080/trySession/SessionTrace

```
First time visitor
```

Not secure | 192.168.0.12:8080/trySession/SessionTrace

```
Welcome back
This is your visit 2
```

149

149

Example: cookie to detect first time visitor

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

     Cookie cook[] = request.getCookies();
     boolean newbie = true;
     int visit = 0;
     if (cook !=null) {
        for (Cookie c: cook) {
          if (c.getName().equals ("count") )
            visit = Integer.parseInt(c.getValue());
        }
     }
     if (visit ==0) {
          response.getWriter().println("First time visitor");
          Cookie c = new Cookie ("count", "1");
          c.setMaxAge(300); // 5 minutes   default 30 min
          response.addCookie(c);

     }
     else {
          response.getWriter().println("Welcome back");
          response.getWriter().println("This is your visit "+ (visit+1));

          Cookie c = new Cookie ("count", visit+1+""); // replace old
          response.addCookie(c);
      }
```

150

150

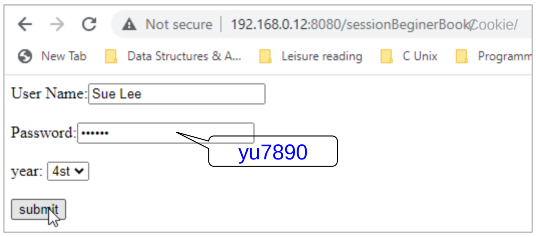
Example: cookie to detect first time visitor

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

     Cookie cook[] = request.getCookies();
     boolean newbie = true;
     int visit = 0;
     if (cook !=null) {
        for (Cookie c: cook) {
          if (c.getName().equals ("count") )
            visit = Integer.parseInt(c.getValue());
        }
     }
     if (visit ==0) {
          response.getWriter().println("First time visitor");
          Cookie c = new Cookie ("count", "1");
          c.setMaxAge(300); // 5 minutes   default 30 min
          response.addCookie(c);

     }
     else {
          response.getWriter().println("Welcome back");
          response.getWriter().println("This is your visit "+ (visit+1));

          for (Cookie c: cook) {
            if (c.getName().equals ("count") ) {
                    c.setValue(visit+1+"");  // or create new
                    response.addCookie(c);  // still needed
```

151

151

Another example (revisit)


Not secure | 192.168.0.12:8080/sessionBeginerBook/Cookie/
New Tab | Data Structures & A... | Leisure reading | C Unix | Programm
submit

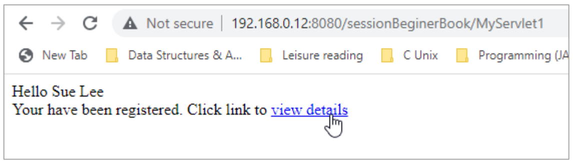
User Name: Sue Lee

Password: ••••••

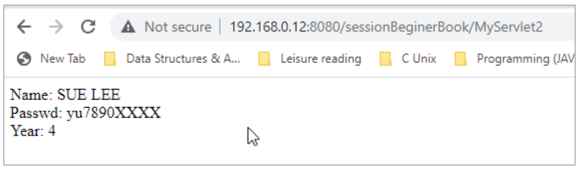
yu7890

year: 4st

No space


Not secure | 192.168.0.12:8080/sessionBeginerBook/MyServlet1
New Tab | Data Structures & A... | Leisure reading | C Unix | Programming (JA

Hello Sue Lee
Your have been registered. Click link to view details


Not secure | 192.168.0.12:8080/sessionBeginerBook/MyServlet2
New Tab | Data Structures & A... | Leisure reading | C Unix | Programming (JAV

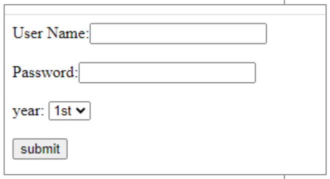
Name: SUE LEE
Passwd: yu7890XXXX
Year: 4

152

152

Another example

```
<form action="MyServlet1" method="post">
  User Name:<input type="text" name="userName"/><br/><br>
  Password:<input type="password">
  name="userPassword"/><br/><br>
  year: <select name="year">
          <option value="1"> 1st
          <option value="2"> 2st
          <option value="3"> 3st
          <option value="4"> 4st
         </select>
  <br><br>
  <input type="submit" value="submit"/>
</form>
```



User Name:

Password:

year: 1st

MyServlet1

```
      response.setContentType("text/html");
      PrintWriter pwriter = response.getWriter();
      String name = request.getParameter("userName").toUpperCase();
      String password = request.getParameter("userPassword")+ "XXXX";
      String year = request.getParameter("year");

      Cookie c1 = new Cookie ("uname",name);
      Cookie c2 = new Cookie ("upass", password);
      Cookie c3 = new Cookie ("year",year);
      response.addCookie(c1);
      response.addCookie(c2);
      response.addCookie(c3);

      pwriter.println("Hello "+name +"<br>");
      pwriter.println("Your have been registered. Click link to ");
      pwriter.print("<a href='MyServlet2'>view details</a>");
```

153

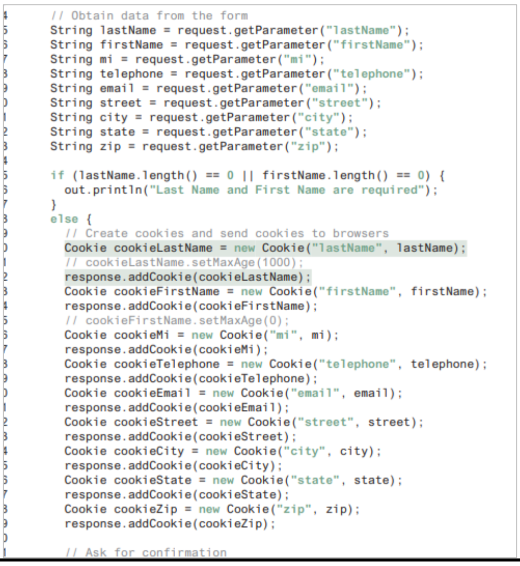
153

MyServlet2

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

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

      Cookie cks[] = request.getCookies();
      for (Cookie c : cks) {
        if (c.getName().equals("uname"))
           pwriter.println("Name:  "+ c.getValue()+"<br>");
        else if (c.getName().equals("upass"))
           pwriter.println("Passwd:  "+ c.getValue()+"<br>")
        else if (c.getName().equals("year"))
           pwriter.println("Year:  "+  c.getValue() + "<br>");

      }



      pwriter.close();
}
```

Disadvantage of Cookies

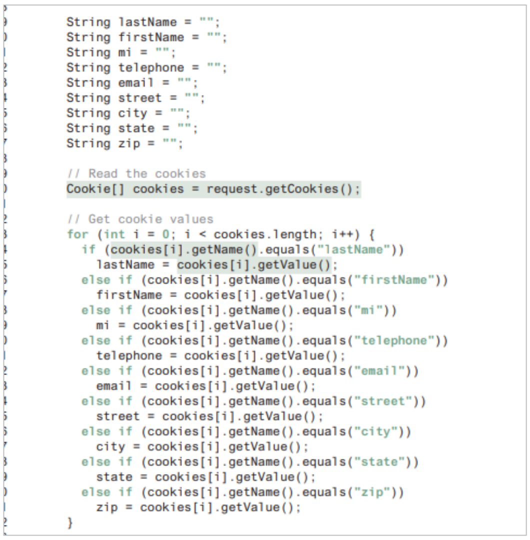
1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.
   - For object, store field by field, multiple cookies



```
    // Obtain data from the form
    String lastName = request.getParameter("lastName");
    String firstName = request.getParameter("firstName");
    String mi = request.getParameter("mi");
    String telephone = request.getParameter("telephone");
    String email = request.getParameter("email");
    String street = request.getParameter("street");
    String city = request.getParameter("city");
    String state = request.getParameter("state");
    String zip = request.getParameter("zip");

    if (lastName.length() == 0 || firstName.length() == 0) {
      out.println("Last Name and First Name are required");
    }
    else {
      // Create cookies and send cookies to browsers
      Cookie cookieLastName = new Cookie("lastName", lastName);
      // cookieLastName.setMaxAge(1000);
      response.addCookie(cookieLastName);
      Cookie cookieFirstName = new Cookie("firstName", firstName);
      response.addCookie(cookieFirstName);
      // cookieFirstName.setMaxAge(0);
      Cookie cookieMi = new Cookie("mi", mi);
      response.addCookie(cookieMi);
      Cookie cookieTelephone = new Cookie("telephone", telephone);
      response.addCookie(cookieTelephone);
      Cookie cookieEmail = new Cookie("email", email);
      response.addCookie(cookieEmail);
      Cookie cookieStreet = new Cookie("street", street);
      response.addCookie(cookieStreet);
      Cookie cookieCity = new Cookie("city", city);
      response.addCookie(cookieCity);
      Cookie cookieState = new Cookie("state", state);
      response.addCookie(cookieState);
      Cookie cookieZip = new Cookie("zip", zip);
      response.addCookie(cookieZip);

      // Ask for confirmation
```

Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.
   - For object, store field by field, multiple cookies



```
        String lastName = "";
        String firstName = "";
        String mi = "";
        String telephone = "";
        String email = "";
        String street = "";
        String city = "";
        String state = "";
        String zip = "";

        // Read the cookies
        Cookie[] cookies = request.getCookies();

        // Get cookie values
        for (int i = 0; i < cookies.length; i++) {
          if (cookies[i].getName().equals("lastName"))
            lastName = cookies[i].getValue();
          else if (cookies[i].getName().equals("firstName"))
            firstName = cookies[i].getValue();
          else if (cookies[i].getName().equals("mi"))
            mi = cookies[i].getValue();
          else if (cookies[i].getName().equals("telephone"))
            telephone = cookies[i].getValue();
          else if (cookies[i].getName().equals("email"))
            email = cookies[i].getValue();
          else if (cookies[i].getName().equals("street"))
            street = cookies[i].getValue();
          else if (cookies[i].getName().equals("city"))
            city = cookies[i].getValue();
          else if (cookies[i].getName().equals("state"))
            state = cookies[i].getValue();
          else if (cookies[i].getName().equals("zip"))
            zip = cookies[i].getValue();
        }
```

157

157

HTTP - Stateless protocol (2)

There are four ways to maintain the state:

- Hidden fields in forms
  - `<input type="hidden" value="<valore>" />`
- URL rewriting
  - e.g., `http://host/servletPages/ShowSession;jsessionid=E4D371710`
  - String `encodeURL(String url)` of the class `HttpServletResponse`
  - where: url is the original l'URL and the output is the rewritten one
- Cookies
  - `javax.servlet.http.Cookie(String name, String val)`
  - methods: `setName(String cookieName)`, `setValue(String cookieValue)`, etc.
- **Servlets (HttpSession API)**

159

159

Servlet HTTP sessions

- The hidden data are in HTML form can be viewed from the browser. Also, it need dynamic form. **href will not work.**
- Cookies are stored in the Cache directory of the browser. Because of security concerns, some browsers do not accept cookies. The client can turn the cookies off and limit their number.
- Another problem is that hidden data and cookies pass data as strings. You cannot pass objects using these two methods.

- Programming your own session tracking (using the above approaches Cookies and Hidden fields) is thus tedious and cumbersome.

- Java servlet API provides the **javax.servlet.http.HttpSession** interface, which provides a way to identify a user across more than one page request or visit to a website and to store information about that user.
- The servlet container uses this interface to create a session between an HTTP client and an HTTP server.

- High level interface built on top of Cookies or URL rewriting under the hood. Done by the container for you.
  - Authors do not need to explicitly manipulate cookies or URL appending.

160

160

- To use the Java servlet API for session tracking, first create a session object using the **getSession**() method in the **HttpServletRequest** interface:

```
HttpSession session = request.getSession();  // (true)
```

This obtains the session or creates a new session if the client does not have a session on the server

```
HttpSession session = request.getSession(false);//does not create
```

- Methods of HttpSession

public void setAttribute(String name, Object value): Binds the object with a name and stores the name/value pair as an attribute of the HttpSession object. If an attribute already exists, then this method replaces the existing attributes.

public Object getAttribute(String name): Returns the String object specified in the parameter, from the session object. If no object is found for the specified attribute, then the getAttribute() method returns null.

public Enumeration getAttributeNames(): Returns an Enumeration that contains the name of all the objects that are bound as attributes to the session object.

public void removeAttribute(String name): Removes the given attribute from session.

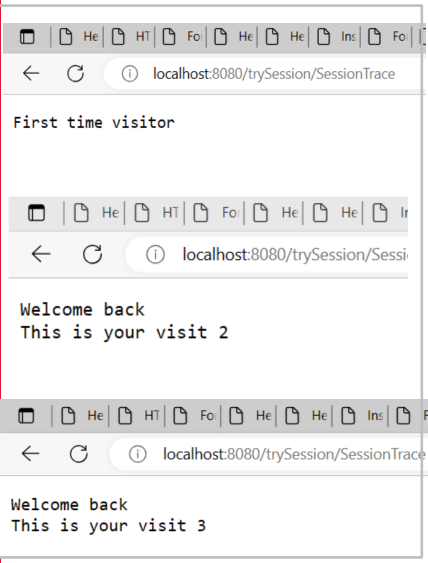
public Boolean isNew(): Returns true if the client does not yet know about the session

public void setMaxInactiveInterval(int interval): Sets the session inactivity time in seconds. This is the time in seconds that specifies how long a sessions remains active since last request received from client. Default is 30 min.

161 **public void invalidate ()** invalidate a session manually, All objects bound to the session are removed.

161

Example: cookie to detect first time visitor



localhost:8080/trySession/SessionTrace

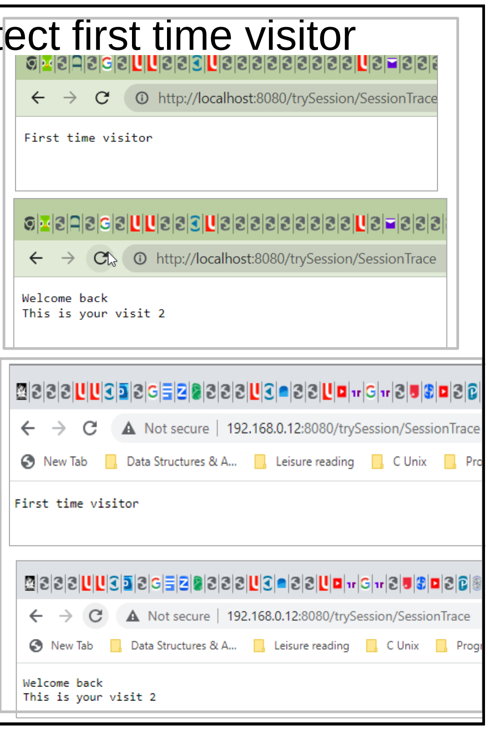
```
First time visitor
```

localhost:8080/trySession/Sessi

```
Welcome back
This is your visit 2
```

localhost:8080/trySession/SessionTrace

```
Welcome back
This is your visit 3
```



http://localhost:8080/trySession/SessionTrace

```
First time visitor
```

http://localhost:8080/trySession/SessionTrace

```
Welcome back
This is your visit 2
```

Not secure | 192.168.0.12:8080/trySession/SessionTrace

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

```
First time visitor
```

Not secure | 192.168.0.12:8080/trySession/SessionTrace

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

```
Welcome back
This is your visit 2
```

163

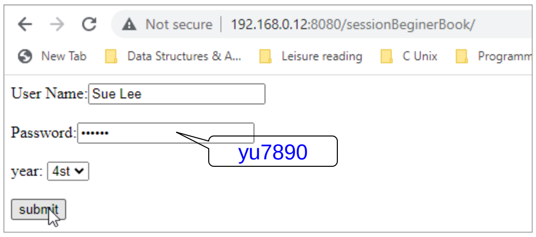
163

```
@WebServlet("/SessionTrace")
public class sessionTrace extends HttpServlet
{
   public void doPost(HttpServletRequest request,
     HttpServletResponse response)
       throws ServletException, IOException
   {
       response.setContentType("text/html");
       PrintWriter pwriter = response.getWriter();

       HttpSession s = request.getSession(); // create one if no
       if (s.isNew()){
          pwriter.println("First time visitor");
          s.setAttribute("count", 1);    // new Integer(1));
       }
       else{ // session is not new
          pwriter.println("Welcome back");
          int c = (Integer)s.getAttribute("count");
          pwriter.println("This is your visit: " + (c+1));
          s.setAttribute("count", c+1); // new Integer(c+1));

       }
   }
```

164

164

Another example


192.168.0.12:8080/sessionBeginerBook/

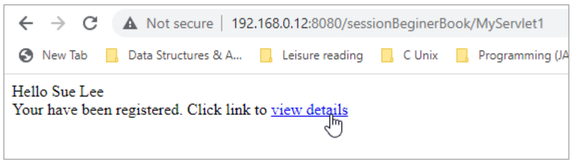
User Name: Sue Lee

Password: ••••••

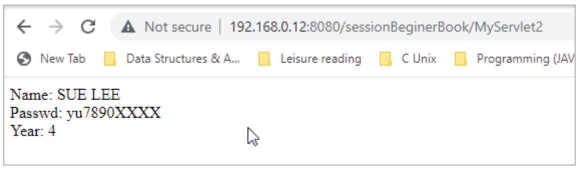
yu7890

year: 4st

submit


192.168.0.12:8080/sessionBeginerBook/MyServlet1

Hello Sue Lee
Your have been registered. Click link to view details


192.168.0.12:8080/sessionBeginerBook/MyServlet2

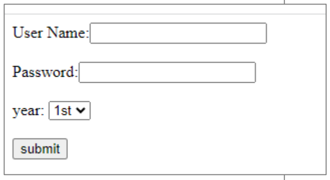
Name: SUE LEE
Passwd: yu7890XXXX
Year: 4

165

165

Another example

```
<form action="MyServlet1" method="post">
  User Name:<input type="text" name="userName"/><br/><br>
  Password:<input type="password">
  name="userPassword"/><br/><br>
  year: <select name="year">
          <option value="1"> 1st
          <option value="2"> 2st
          <option value="3"> 3st
          <option value="4"> 4st
         </select>
  <br><br>
  <input type="submit" value="submit"/>
</form>
```


User Name:

Password:

year: 1st

submit

MyServlet1

```
      response.setContentType("text/html");
      PrintWriter pwriter = response.getWriter();
      String name = request.getParameter("userName").toUpperCase();
      String password = request.getParameter("userPassword")+"XXXX";
      String year = request.getParameter("year");

      pwriter.println("Hello "+name +"<br>");

      HttpSession session=request.getSession();
      session.setAttribute("uname",name);
      session.setAttribute("upass",password);
      session.setAttribute("year", year);

      pwriter.println("Your have been registered. Click link to ");
      pwriter.print("<a href='MyServlet2'>view details</a>");
```

166

166

MyServlet2

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

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

      HttpSession session=request.getSession(false); // don't create
      String myName=(String)session.getAttribute("uname");
      String myPass=(String)session.getAttribute("upass");
      String myYear=(String)session.getAttribute("year");

      pwriter.println("Name:  "+myName+"<br>");
      pwriter.println("Passwd:  "+ myPass+"<br>");
      pwriter.println("Year:  "+ myYear+"<br>");
      pwriter.close();
}
```

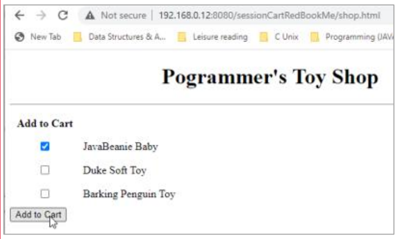
Can encapsulate into object and set/get object

YORK U UNIVERSITÉ UNIVERSITY

167

167

Another example: simple shopping cart



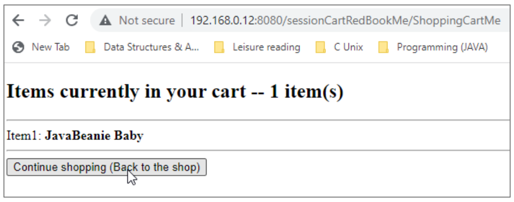
Not secure | 192.168.0.12:8080/sessionCartRedBookMe/shop.html

Pogrammer's Toy Shop

Add to Cart

- [✓] JavaBeanie Baby
- [ ] Duke Soft Toy
- [ ] Barking Penguin Toy

Add to Cart

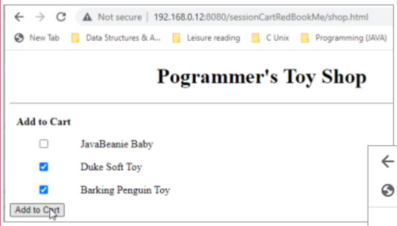


Not secure | 192.168.0.12:8080/sessionCartRedBookMe/ShoppingCartMe

Items currently in your cart -- 1 item(s)

Item1: **JavaBeanie Baby**

Continue shopping (Back to the shop)



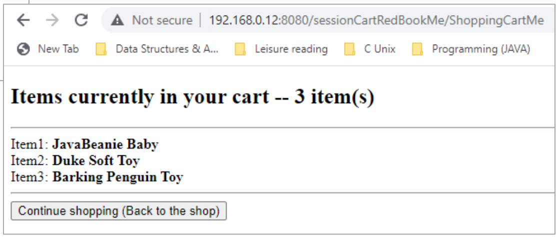
Not secure | 192.168.0.12:8080/sessionCartRedBookMe/shop.html

Pogrammer's Toy Shop

Add to Cart

- [ ] JavaBeanie Baby
- [✓] Duke Soft Toy
- [✓] Barking Penguin Toy

Add to Cart



Not secure | 192.168.0.12:8080/sessionCartRedBookMe/ShoppingCartMe

Items currently in your cart -- 3 item(s)

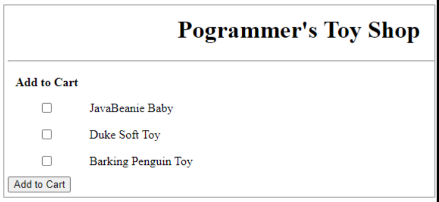
Item1: **JavaBeanie Baby**
Item2: **Duke Soft Toy**
Item3: **Barking Penguin Toy**

Continue shopping (Back to the shop)

169

169

```
<body>
<center><h1>Pogrammer's Toy Shop</h1></center>
<hr>
<form action="ShoppingCartMe" method="post" >

 <table cellspacing="5" cellpadding="5">
 <tr>
    <td align="center"> <B>Add to Cart</B> </td>
    <td align="center"> </td>
  </tr>

  <tr>
    <td align="center"> <input type="checkbox" name="item" value="JavaBeanie Baby"> </td>
    <td align="left"> JavaBeanie Baby </td>
  </tr>
  <tr>
    <td align="center"> <input type="checkbox" name="item" value="Duke Soft Toy"> </td>
    <td align="left"> Duke Soft Toy</td>
  </tr>
  <tr>
    <td align="center"> <input type="checkbox" name="item" value=" Barking Penguin Toy ">
   </td>
    <td align="left">Barking Penguin Toy</td>
  </tr>
  </table>
  <input type="submit" name=bun_submit" value="Add to Cart" >
  </form>
 </html>
```



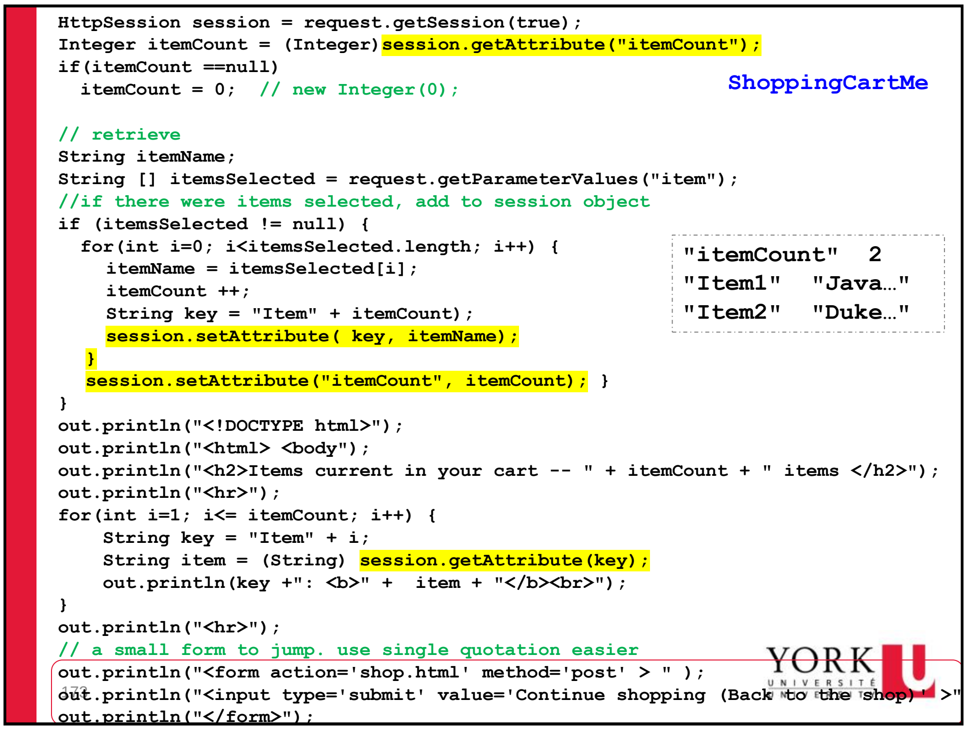
170

ShoppingCartMeNoSession

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

// retrieve
String itemsSelected []= request.getParameterValues("item");
int itemCount = itemsSelected.length;

out.println("<!DOCTYPE html>");
out.println("<html><head>");
out.println("<body>");
out.println("<h2>Items current in your cart -- " + itemCount + " items </h2>");
out.println("<hr>");

for(int i=0; i<= itemCount; i++) {
    out.println("Item"+(i+1)+ ": <b>" itemsSelected[i] + "</b><br>");
}
out.println("<hr>");
// a small form to jump. use single quotation easier
out.println("<form action='shop.html' method='post' > " );
out.println("<input type='submit' value='Continue shopping (Back to the shop)' >")
out.println("</form>");

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

171

171



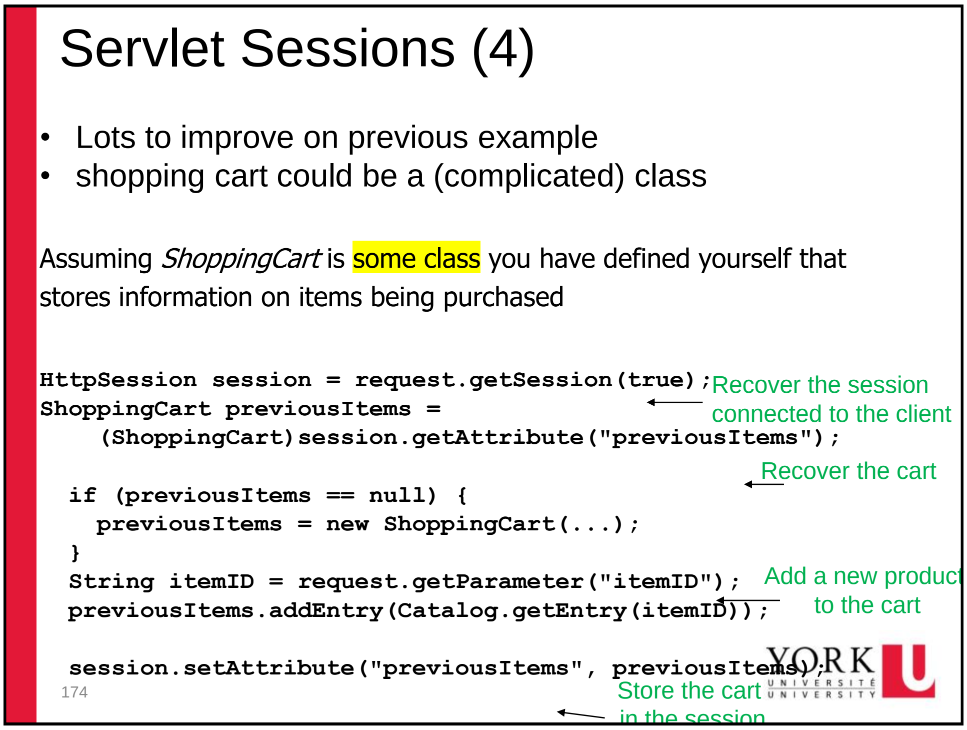
ShoppingCartMe

```
HttpSession session = request.getSession(true);
Integer itemCount = (Integer)session.getAttribute("itemCount");
if(itemCount ==null)
  itemCount = 0;  // new Integer(0);

// retrieve
String itemName;
String [] itemsSelected = request.getParameterValues("item");
//if there were items selected, add to session object
if (itemsSelected != null) {
  for(int i=0; i<itemsSelected.length; i++) {
    itemName = itemsSelected[i];
    itemCount ++;
    String key = "Item" + itemCount);
    session.setAttribute( key, itemName);
  }
  session.setAttribute("itemCount", itemCount); }
}
out.println("<!DOCTYPE html>");
out.println("<html> <body");
out.println("<h2>Items current in your cart -- " + itemCount + " items </h2>");
out.println("<hr>");
for(int i=1; i<= itemCount; i++) {
    String key = "Item" + i;
    String item = (String) session.getAttribute(key);
    out.println(key +": <b>" +  item + "</b><br>");
}
out.println("<hr>");
// a small form to jump. use single quotation easier
out.println("<form action='shop.html' method='post' > " );
out.println("<input type='submit' value='Continue shopping (Back to the shop)' >"
out.println("</form>");
```

"itemCount"	2
"Item1"	"Java…"
"Item2"	"Duke…"

173

173



Servlet Sessions (4)

- Lots to improve on previous example
- shopping cart could be a (complicated) class

Assuming *ShoppingCart* is some class you have defined yourself that stores information on items being purchased

```
HttpSession session = request.getSession(true);
ShoppingCart previousItems =
    (ShoppingCart)session.getAttribute("previousItems");

  if (previousItems == null) {
    previousItems = new ShoppingCart(...);
  }
  String itemID = request.getParameter("itemID");
  previousItems.addEntry(Catalog.getEntry(itemID));

  session.setAttribute("previousItems", previousItems);
```

Recover the session connected to the client

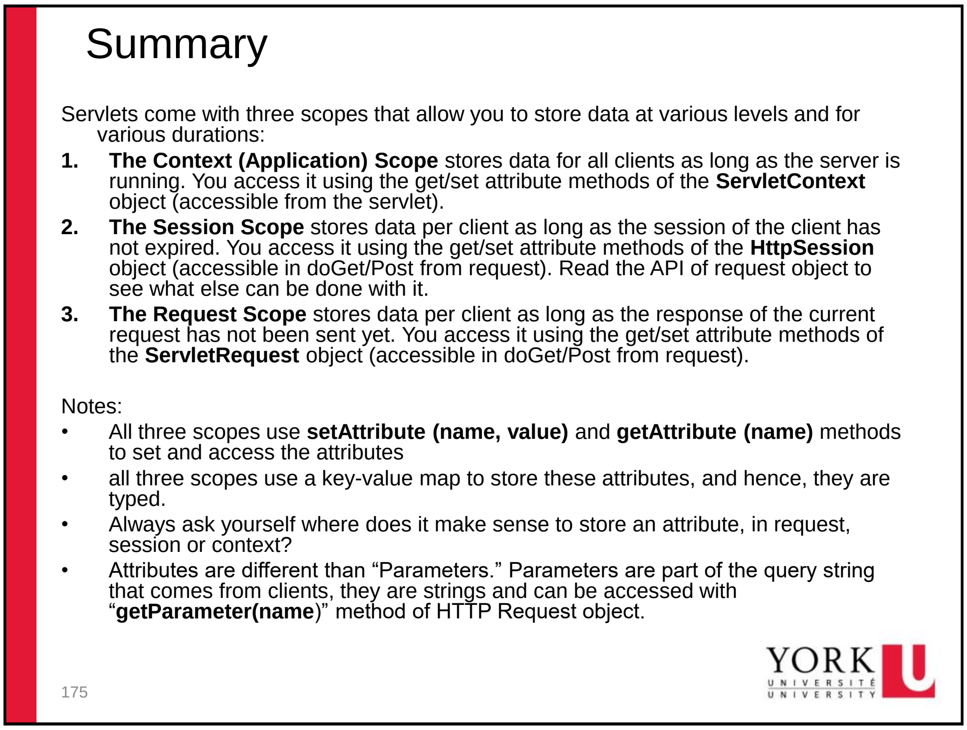
Recover the cart

Add a new product to the cart

Store the cart in the session

174

174



Summary

Servlets come with three scopes that allow you to store data at various levels and for various durations:

1. **The Context (Application) Scope** stores data for all clients as long as the server is running. You access it using the get/set attribute methods of the **ServletContext** object (accessible from the servlet).
2. **The Session Scope** stores data per client as long as the session of the client has not expired. You access it using the get/set attribute methods of the **HttpSession** object (accessible in doGet/Post from request). Read the API of request object to see what else can be done with it.
3. **The Request Scope** stores data per client as long as the response of the current request has not been sent yet. You access it using the get/set attribute methods of the **ServletRequest** object (accessible in doGet/Post from request).

Notes:

- All three scopes use **setAttribute (name, value)** and **getAttribute (name)** methods to set and access the attributes
- all three scopes use a key-value map to store these attributes, and hence, they are typed.
- Always ask yourself where does it make sense to store an attribute, in request, session or context?
- Attributes are different than "Parameters." Parameters are part of the query string that comes from clients, they are strings and can be accessed with "**getParameter(name**)" method of HTTP Request object.

175

YORK UNIVERSITÉ UNIVERSITY

175