LISTENER
and
SERVLET FILTER

Instructor:

Table Content

Learning Goals

After the course, attendees will be able to:

- Understand Event and Listener, Web filter in Servlet
- Can use to build Project

Section 1

EVENT AND LISTENER

Event and Listener in Servlet

- Events are basically **occurrence of something**. Changing the state of an object is known as an event.
- We can perform some **important tasks** at the occurrence of these exceptions, such as counting total and current logged-in users, creating tables of the database at time of deploying the project, creating database connection object etc.
- There are many **Event** classes and **Listener** interfaces in the **javax.servlet** and **javax.servlet.http** packages.

Event and Listener in Servlet

- **Event classes**
  - ✓ ServletRequestEvent
  - ✓ ServletContextEvent
  - ✓ ServletRequestAttributeEvent
  - ✓ ServletContextAttributeEvent
  - ✓ **HttpSessionEvent**
  - ✓ **HttpSessionBindingEvent**
- **Event interfaces**
  - ✓ ServletRequestListener
  - ✓ ServletRequestAttributeListener
  - ✓ ServletContextListener
  - ✓ ServletContextAttributeListener
  - ✓ **HttpSessionListener**
  - ✓ HttpSessionAttributeListener
  - ✓ **HttpSessionBindingListener**
  - ✓ HttpSessionActivationListener

HttpSessionEvent and HttpSessionListen

- The **HttpSessionEvent** is notified when session ***object is changed***.
- The corresponding Listener interface for this event is **HttpSessionListener**.
- We can perform some operations at this event such as **counting total** and **current logged-in users**, maintaing a log of user details such as **login time**, **logout time** etc.
- **Methods of HttpSessionListener interface:**
  - ✓ public void **sessionCreated(HttpSessionEvent e):** is invoked when session object is created.
  - ✓ public void **sessionDestroyed(ServletContextEvent e):** is invoked when session is invalidated.

HttpSessionEvent and HttpSessionListen

❖ Example: to count total and current logged-in users

- ✓ **login.html**: to get input from the user.
- ✓ **CountUserListener.java**: A listener class that counts total and current logged-in users and stores this information in ServletContext object as an attribute.
- ✓ **LoginServlet.java**: A Servlet class that creates session and prints the total and current logged-in users.
- ✓ **LogoutServlet.java**: A Servlet class that invalidates session.

HttpSessionEvent and HttpSessionL

- **Login.html**

```
<head>
      <title>Login</title>
</head>
<form action="Login">
      Name:<input type="text" name="username"><br>
      Password:<input type="password" name="userpass"><br>
      <input type="submit" value="Login" />
</form>
```

- **LoginServlet.java**

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
                                        throws ServletException, IOException {

      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String userName =request.getParameter("username");
      out.print("Welcome " + userName);
      HttpSession session=request.getSession();
      session.setAttribute("uname", userName);
      ServletContext ctx=getServletContext();
      int t=(Integer)ctx.getAttribute("totalusers");
      int c=(Integer)ctx.getAttribute("currentusers");
      out.print("<br>Total users= "+t);
      out.print("<br>Current users= "+c);

      out.print("<br><a href='logout'>logout</a>");

      out.close();
}
}
```

HttpSessionEvent and HttpSessionL

- **LogoutServlet.java**

```java
@WebServlet("/logout")
public class LogoutServlet extends HttpServlet {
    /**
     *
     */
    private static final long serialVersionUID = 1L;

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

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

        HttpSession session=request.getSession(false);
        session.invalidate();

        out.print("You are successfully logged out");


        out.close();
    }

}
```

HttpSessionEvent and HttpSessionL

- **CountUserListener.java**

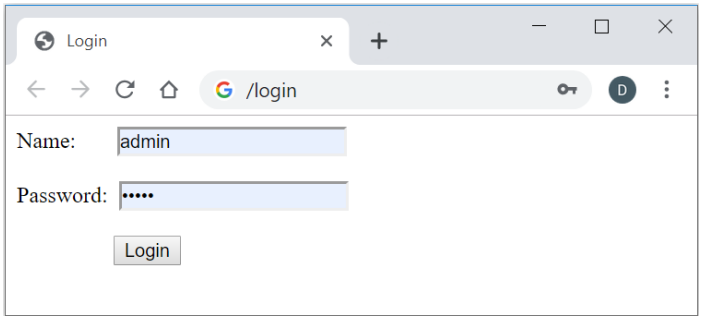
```java
public class CountUserListener implements HttpSessionListener {
    ServletContext ctx = null;
    static int total = 0, current = 0;

    public void sessionCreated(HttpSessionEvent e) {
        total++;
        current++;

        ctx = e.getSession().getServletContext();
        ctx.setAttribute("totalusers", total);
        ctx.setAttribute("currentusers", current);

    }

    public void sessionDestroyed(HttpSessionEvent e) {
        current--;
        ctx.setAttribute("currentusers", current);
    }

}
```

- **Web.xml**

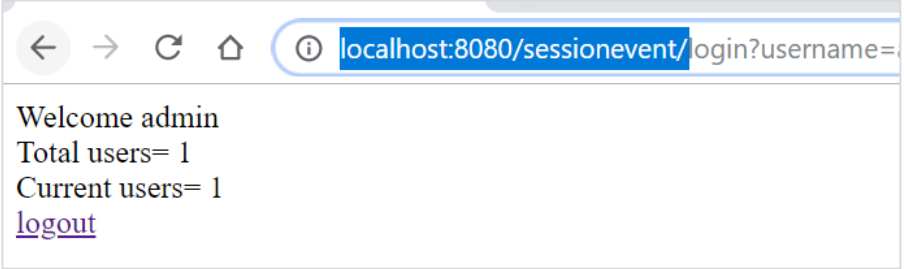
```xml
<listener>
    <listener-class>CountUserListener</listener-class>
</listener>
```

HttpSessionEvent and HttpSessionL

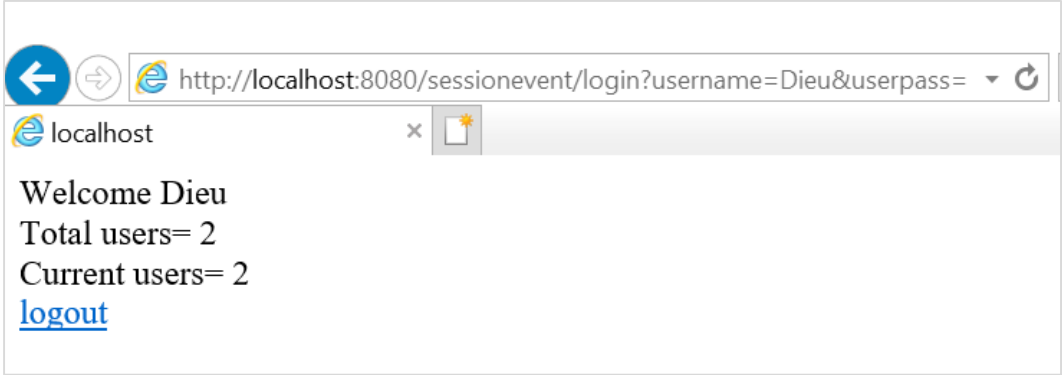
- **Login.html**



- **After an user login:**



- **After two user login:**



HttpSessionBindingListener

- Can be implemented by a class to get notified when its instance is added to the session or when it is removed from the session.
  - ✓ HttpSessionBindingListener#**valueBound**() is invoked when this object is added to the session by the use of HttpSession.setAttribute().
  - ✓ HttpSessionBindingListener#**valueUnbound**() is invoked when this object is removed from the session. That happens when
    - HttpSession.**removeAttribute()** is used for this object.
    - Or HttpSession.**invalidate()** is used.
    - Or after some time of session timeout.

HttpSessionBindingListener

- **Example**
- Step1: Implementing **HttpSessionBindingListener**

```java
@WebListener
public class UserData implements HttpSessionBindingListener {

  @Override
  public void valueBound(HttpSessionBindingEvent event) {
      System.out.println("-- HttpSessionBindingListener#valueBound() --");
      System.out.printf("added attribute name: %s, value:%s %n",
              event.getName(), event.getValue());
  }

  @Override
  public void valueUnbound(HttpSessionBindingEvent event) {
      System.out.println("-- HttpSessionBindingEvent#valueUnbound() --");
      System.out.printf("removed attribute name: %s, value:%s %n",
              event.getName(), event.getValue());
  }
}
```

HttpSessionBindingListener

- Implementing **HttpSessionListener**
- *We are also implementing HttpSessionListener to set a session timeout value and also to print messages to see the relative lifecycle notification of the session itself.*

```
@WebListener
public class MySessionListener implements HttpSessionListener {
  @Override
  public void sessionCreated(HttpSessionEvent se) {
      System.out.println("-- HttpSessionListener#sessionCreated invoked --");
      HttpSession session = se.getSession();
      System.out.println("session id: " + session.getId());
      session.setMaxInactiveInterval(60);//in seconds
  }

  @Override
  public void sessionDestroyed(HttpSessionEvent se) {
      System.out.println("-- HttpSessionListener#sessionDestroyed invoked --");
  }
}
```

HttpSessionBindingListener

- Step2: Create a **MyServlet** class

```java
@WebServlet(name = "myServlet", urlPatterns = {"/"})
public class MyServlet extends HttpServlet {

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
          throws ServletException, IOException {

      HttpSession session = req.getSession(false);
      if (session == null) {
          System.out.println("-- creating new session in the servlet --");
          session = req.getSession(true);
          System.out.println("-- session created in the servlet --");
      }

      UserData userData = (UserData) session.getAttribute("userData");
      if (userData == null) {
          userData = new UserData();
          session.setAttribute("userData", userData);
      }

      resp.setContentType("text/html");
      PrintWriter w = resp.getWriter();
      w.write("Hello !!");
  }
}
```

HttpSessionBindingListener

- Step3: Create a **LogoutServlet** class to removes 'UserData' attribute from the session.

```
@WebServlet(name = "logoutServlet", urlPatterns = {"/clean"})
public class LogoutServlet extends HttpServlet {

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
          throws IOException {

      HttpSession session = req.getSession(false);
      if (session != null && session.getAttribute("userData") != null) {
          System.out.println("-- removing userData attribute from session --");
          session.removeAttribute("userData");
      }

      resp.setContentType("text/html");
      PrintWriter w = resp.getWriter();
      w.write("attribute removed !!");
  }
}
```

HttpSessionBindingListener

- **Output**
- Accessing http://localhost:8080/ from browser, prints the following on the console:

```
-- creating new session in the servlet --
-- HttpSessionListener#sessionCreated invoked --
session id: 56A239430222B1CBE8A8A5DDFB764AFD
-- session created in the servlet --
-- HttpSessionBindingListener#valueBound() --
added attribute name: userData, value:com.logicbig.example.UserData@4d7c522
```

- http://localhost:8080/clean prints the following:

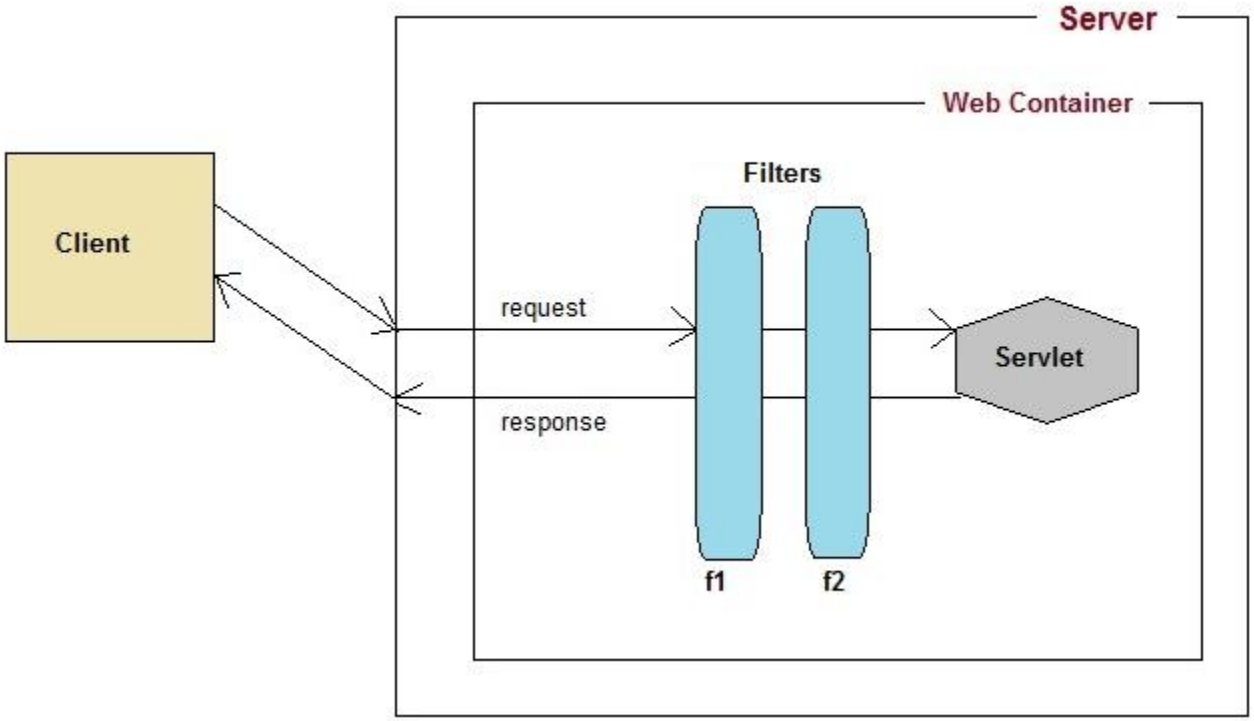
```
-- removing userData attribute from session --
-- HttpSessionBindingEvent#valueUnbound() --
removed attribute name: userData, value:com.logicbig.example.UserData@604a95c7
```

Section 2

SERVLET FILTER

Introduction to Filter API

- **Filters** are compontents that you can use and configure to perform some filtering tasks.
  - ✓ Filter is used for pre-processing of requests and post-processing of responses.



Introduction to Filter API (2/2)

- For creating a filter, we must implement **Filter** interface. Filter interface gives the following life cycle methods for a filter:
  - ✓ void **init**(FilterConfig filterConfig): invoked by the web container to indicate to a filter that it is being placed into service.
  - ✓ void **doFilter**(ServletRequest request, ServletResponse response, FilterChain chain): invoked by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain.
  - ✓ void **destroy**(): invoked by the web container to indicate to a filter that it is being taken out of service.

Servlet Filter

Usage of Filter

- ✓ recording all incoming requests
- ✓ logs the IP addresses of the computers from which the requests originate
- ✓ conversion
- ✓ data compression
- ✓ encryption and decryption
- ✓ input validation etc.

Advantage of Filter

- ✓ Filter is pluggable.
- ✓ One filter don't have dependency onto another resource.
- ✓ Less Maintenance

Deployment Descriptor

```
<web-app ...>
<filter>
<filter-name>MyFilter</filter-name>
<filter-class>MyFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>MyFilter</filter-name>
<url-pattern>..</url-pattern> or <servlet-name>.</servlet-name>
</filter-mapping>
</web-app>
```

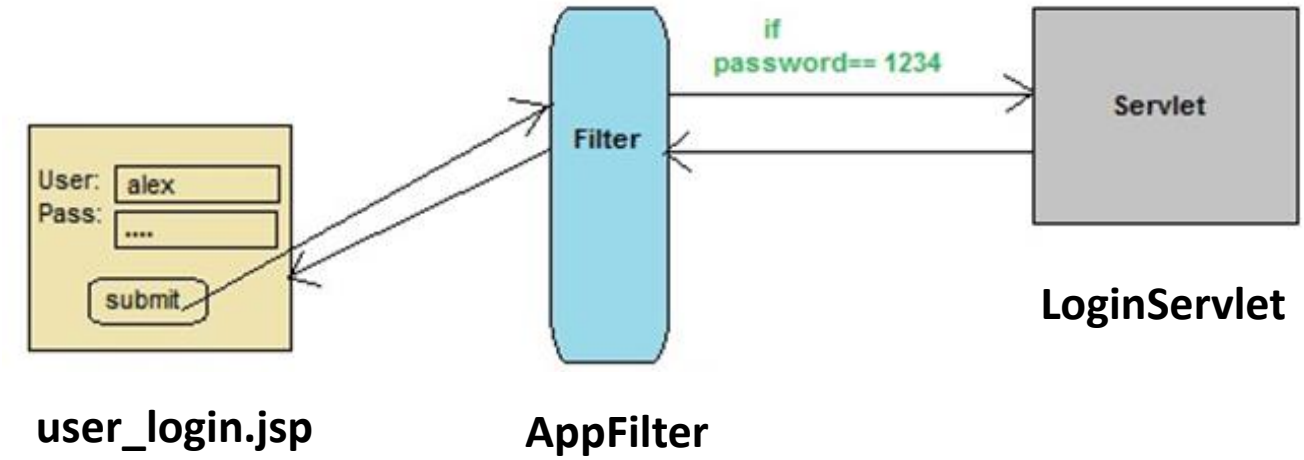
<filter-name> tag is used to give a internal name to your filter

<filter-class> declares the filter that you have created

Either the <url-pattern> or the <servlet-name> element is mandatory which web app resource will use this filter

Practical time

- In this example we are using Filter to authenticate:



Summary

- EVENT AND LISTENER
- SERVLET FILTER
- Q&A

Thank you