Instructor Notes:

Add instructor notes here.

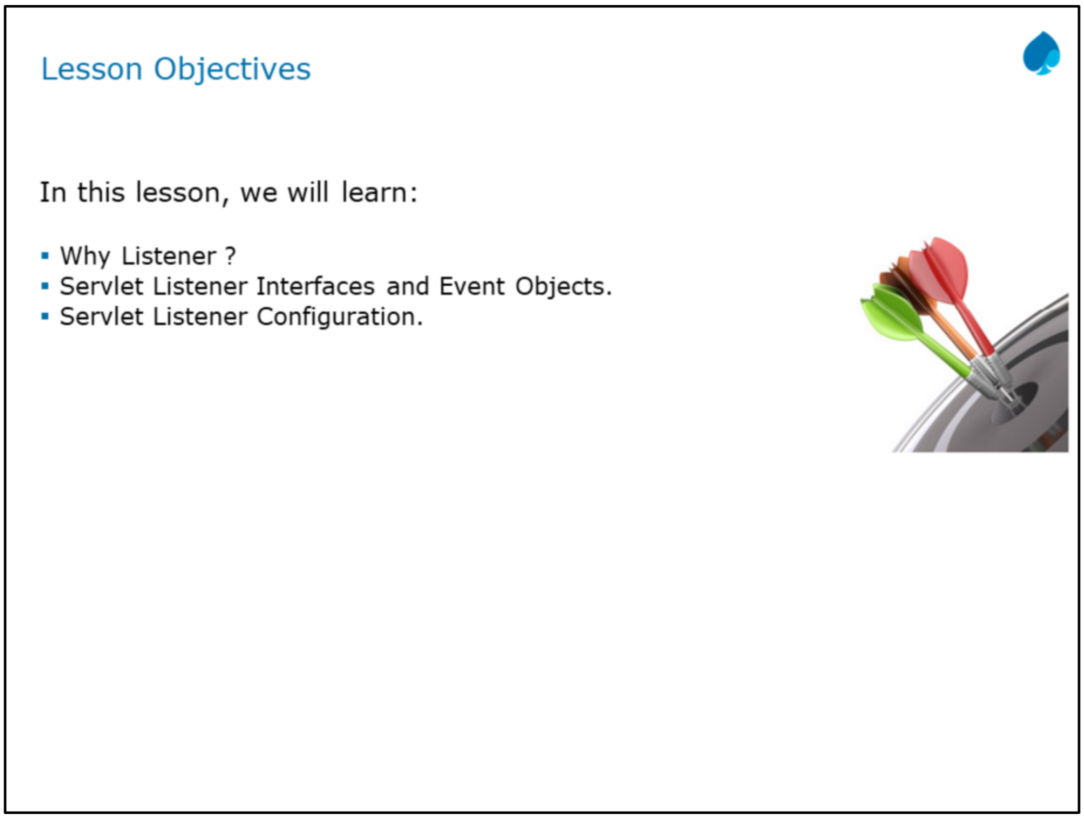
Java Servlets 3.0

Lesson 05: Listener

Capgemini

Instructor Notes:

Explain the lesson coverage



Lesson Objectives

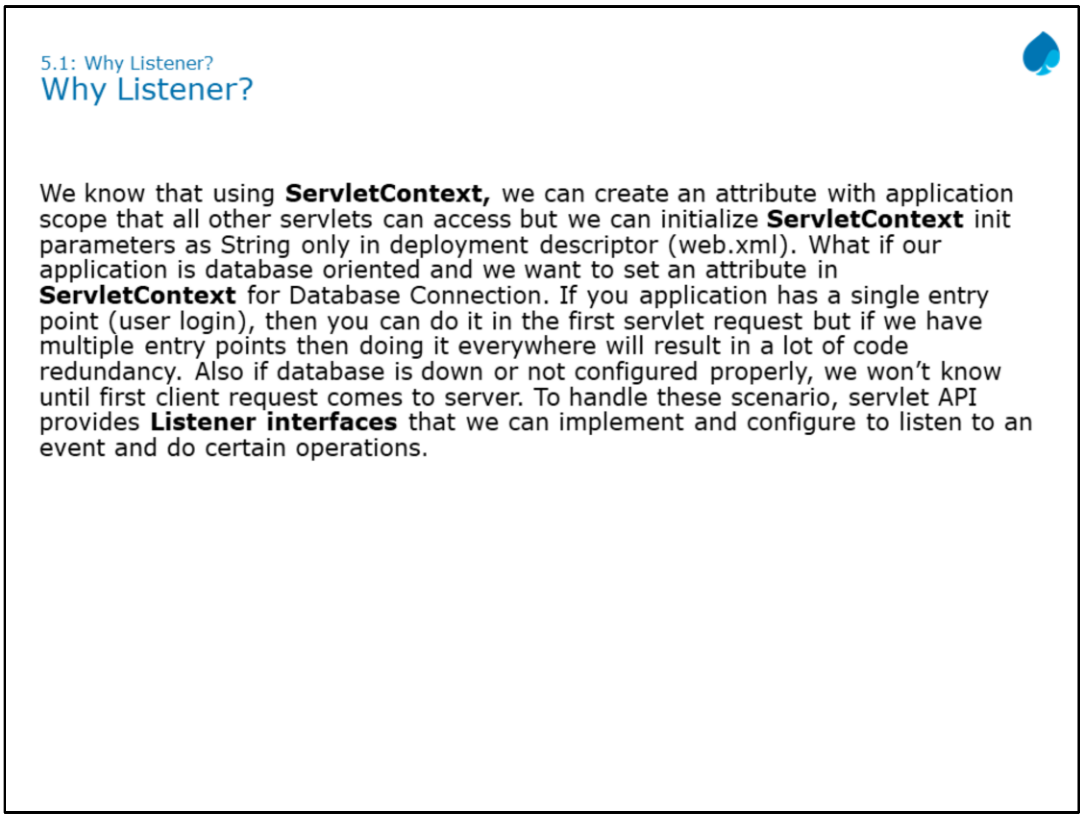
In this lesson, we will learn:

- Why Listener ?
- Servlet Listener Interfaces and Event Objects.
- Servlet Listener Configuration.

Lesson Objectives:
This lesson introduces Session Management. The lesson contents are:
Lesson 08: Session Management
    8.1: Introduction and Need for Session Management
    8.2: Different Techniques of Session Tracking
    8.3: Best Practices

Instructor Notes:

Ask this question to the class.
Most of the participants will answer “Duration from the time user logs in till he logs off” which is not completely true.
Do not display the bulleted point which is the correct answer.
To justify the answer, explain sites like e-newspapers, booksonline which do not require log in but still user can see the same long alphanumeric string in address bar across all the pages on that site.



5.1: Why Listener?

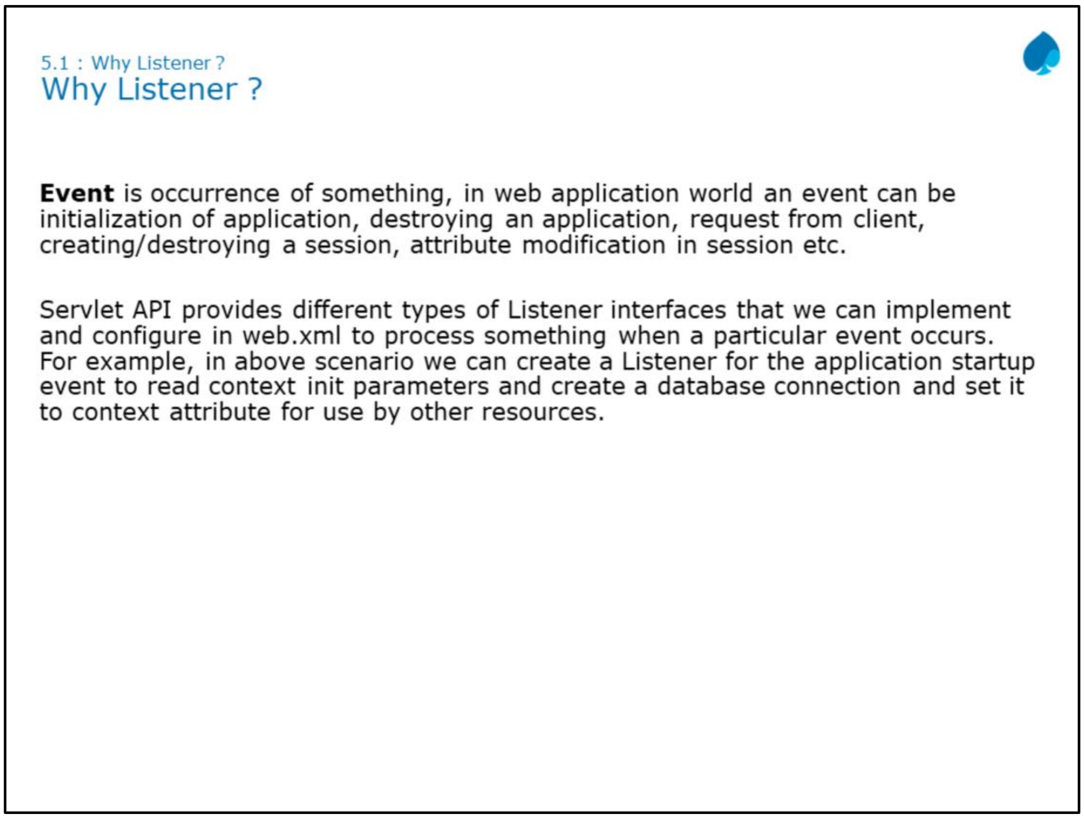
Why Listener?

We know that using **ServletContext,** we can create an attribute with application scope that all other servlets can access but we can initialize **ServletContext** init parameters as String only in deployment descriptor (web.xml). What if our application is database oriented and we want to set an attribute in **ServletContext** for Database Connection. If you application has a single entry point (user login), then you can do it in the first servlet request but if we have multiple entry points then doing it everywhere will result in a lot of code redundancy. Also if database is down or not configured properly, we won’t know until first client request comes to server. To handle these scenario, servlet API provides **Listener interfaces** that we can implement and configure to listen to an event and do certain operations.

Introducing Session Tracking:
A session refers to all the connections that a single client might make to a server in the course of viewing any pages associated with a given application.
Sessions are specific to both the individual user and the application. Therefore every user of an application has a separate session. The user has access to a separate set of session variables.
Logically a session begins with the first connection to an application by a client and ends after that client’s last connection.
Session can be an authenticated session also where a user has to log-in. In this case, sessions begin when the user logs in and ends when the user log-out.
Session also ends after a time-out period of inactivity.

Instructor Notes:

Explain the scenario of online shopping, what would happen if the client's state was not maintained on the server ? What would be the content of the cart while order is processed.
Banking example also can be given with respect to the multiple transactions by the same account holder.



5.1 : Why Listener ?

Why Listener ?

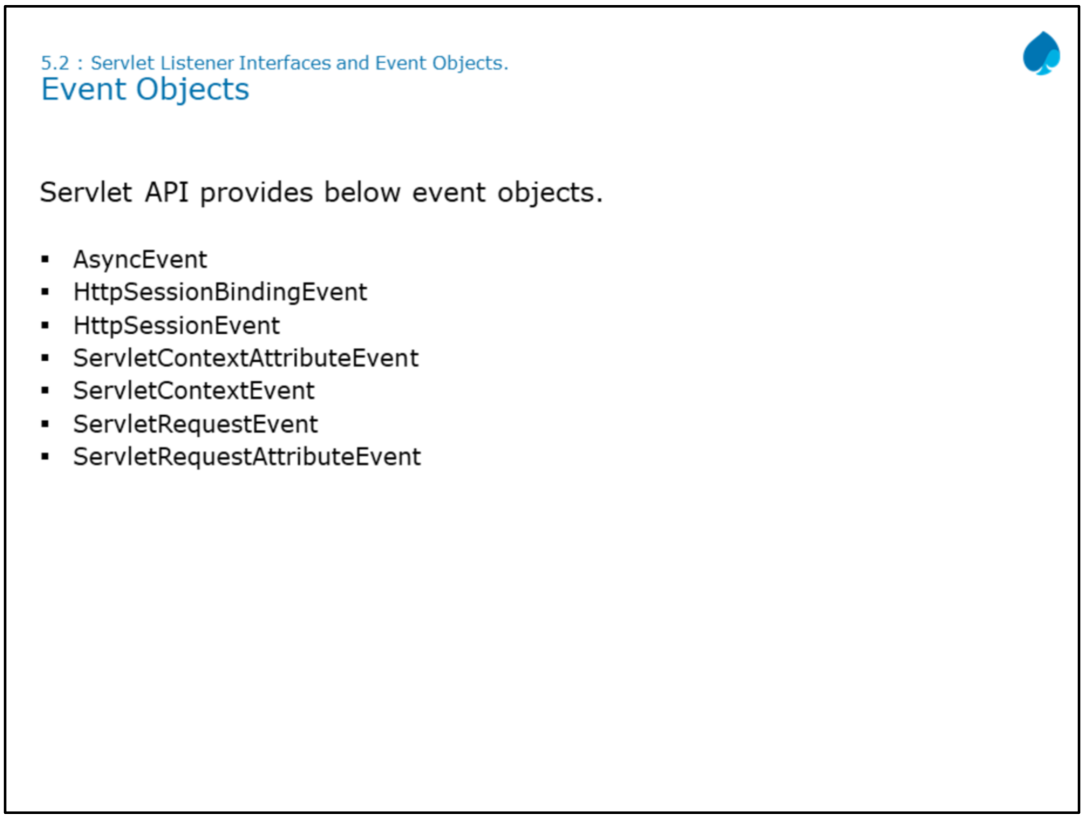
Event is occurrence of something, in web application world an event can be initialization of application, destroying an application, request from client, creating/destroying a session, attribute modification in session etc.

Servlet API provides different types of Listener interfaces that we can implement and configure in web.xml to process something when a particular event occurs. For example, in above scenario we can create a Listener for the application startup event to read context init parameters and create a database connection and set it to context attribute for use by other resources.

Introducing Session Tracking:
What is Session Tracking?
In Session tracking, client first makes a request for any servlet or any page, the container receives the request.
The container then generates a unique identification, called Session ID, for that client and gives it back to the client along with the response. This ID gets stores on the client machine. Thereafter when the client sends a request again to the server, it also sends the session Id with the request. The container sees the Id and sends back the response. While seeing the ID the container recognizes that it is the same previous client making a request.
Thus container identifies the client.

Instructor Notes:

Explain reason behind session tracking. Exhibit that Http is a stateless protocol and thus the need for session tracking


5.2 : Servlet Listener Interfaces and Event Objects.

Event Objects

Servlet API provides below event objects.

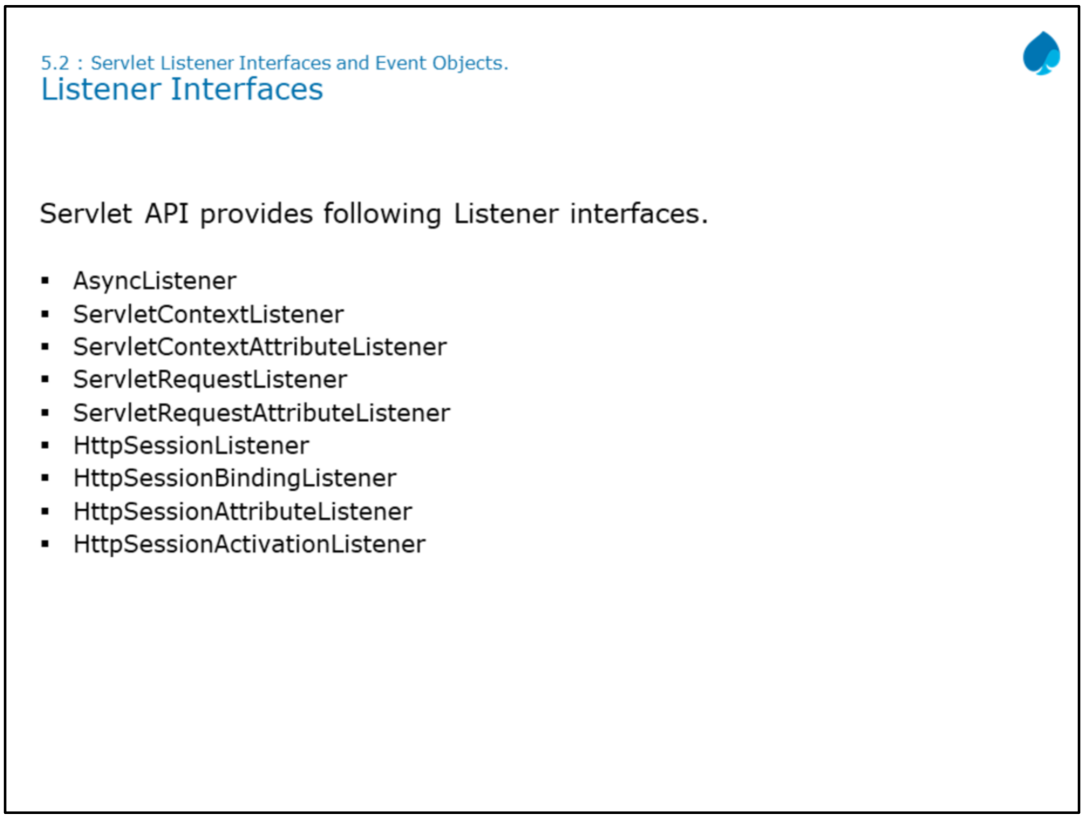
- AsyncEvent
- HttpSessionBindingEvent
- HttpSessionEvent
- ServletContextAttributeEvent
- ServletContextEvent
- ServletRequestEvent
- ServletRequestAttributeEvent

Servlet API provides different kind of listeners for different types of Events. Listener interfaces declare methods to work with a group of similar events, for example we have ServletContext Listener to listen to startup and shutdown event of context. Every method in listener interface takes Event object as input. Event object works as a wrapper to provide specific object to the listeners.
Servlet API provides following event objects.

javax.servlet.AsyncEvent – Event that gets fired when the asynchronous operation initiated on a ServletRequest (via a call to ServletRequest#startAsync or ServletRequest#startAsync(ServletRequest , ServletResponse)) has completed, timed out, or produced an error.
javax.servlet.http.HttpSessionBindingEvent – Events of this type are either sent to an object that implements HttpSessionBindingListener when it is bound or unbound from a session, or to a HttpSessionAttributeListener that has been configured in the web.xml when any attribute is bound, unbound or replaced in a session.
The session binds the object by a call to HttpSession.setAttribute and unbinds the object by a call to HttpSession.removeAttribute.
We can use this event for cleanup activities when object is removed from session.
javax.servlet.http.HttpSessionEvent – This is the class representing event notifications for changes to sessions within a web application.
javax.servlet.ServletContextAttributeEvent – Event class for notifications about changes to the attributes of the ServletContext of a web application.
javax.servlet.ServletContextEvent – This is the event class for notifications about changes to the servlet context of a web application.
javax.servlet.ServletRequestEvent – Events of this kind indicate lifecycle events for a ServletRequest. The source of the event is the ServletContext of this web application.
javax.servlet.ServletRequestAttributeEvent – This is the event class for notifications of changes to the attributes of the servlet request in an application.

Instructor Notes:

List the Session tracking techniques with a brief explanation. Tell participants that detailed explanation is in the following slides and we shall be seeing some of these in demos later.


5.2 : Servlet Listener Interfaces and Event Objects.

Listener Interfaces

Servlet API provides following Listener interfaces.

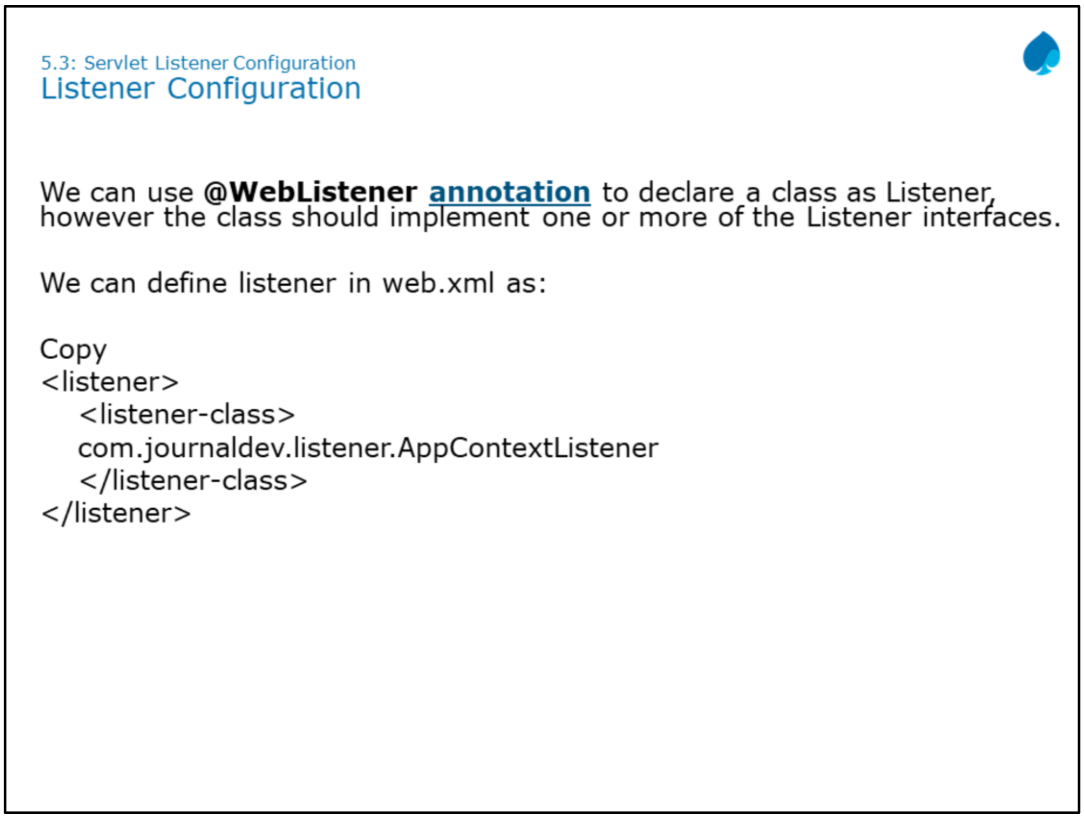
- AsyncListener
- ServletContextListener
- ServletContextAttributeListener
- ServletRequestListener
- ServletRequestAttributeListener
- HttpSessionListener
- HttpSessionBindingListener
- HttpSessionAttributeListener
- HttpSessionActivationListener

Servlet API provides following Listener interfaces.

javax.servlet.AsyncListener – Listener that will be notified in the event that an asynchronous operation initiated on a ServletRequest to which the listener had been added has completed, timed out, or resulted in an error.
javax.servlet.ServletContextListener – Interface for receiving notification events about ServletContext lifecycle changes.
javax.servlet.ServletContextAttributeListener – Interface for receiving notification events about ServletContext attribute changes.
javax.servlet.ServletRequestListener – Interface for receiving notification events about requests coming into and going out of scope of a web application.
javax.servlet.ServletRequestAttributeListener – Interface for receiving notification events about ServletRequest attribute changes.
javax.servlet.http.HttpSessionListener – Interface for receiving notification events about HttpSession lifecycle changes.
javax.servlet.http.HttpSessionBindingListener – Causes an object to be notified when it is bound to or unbound from a session.
javax.servlet.http.HttpSessionAttributeListener – Interface for receiving notification events about HttpSession attribute changes.
javax.servlet.http.HttpSessionActivationListener – Objects that are bound to a session may listen to container events notifying them that sessions will be passivated and that session will be activated. A container that migrates session between VMs or persists sessions is required to notify all attributes bound to sessions implementing HttpSessionActivationListener.

Instructor Notes:

Convey this is oldest and the simplest way of session tracking. Point out the disadvantage. Briefly explain small html fragment from the notes



5.3: Servlet Listener Configuration

Listener Configuration

We can use **@WebListener** annotation to declare a class as Listener, however the class should implement one or more of the Listener interfaces.

We can define listener in web.xml as:

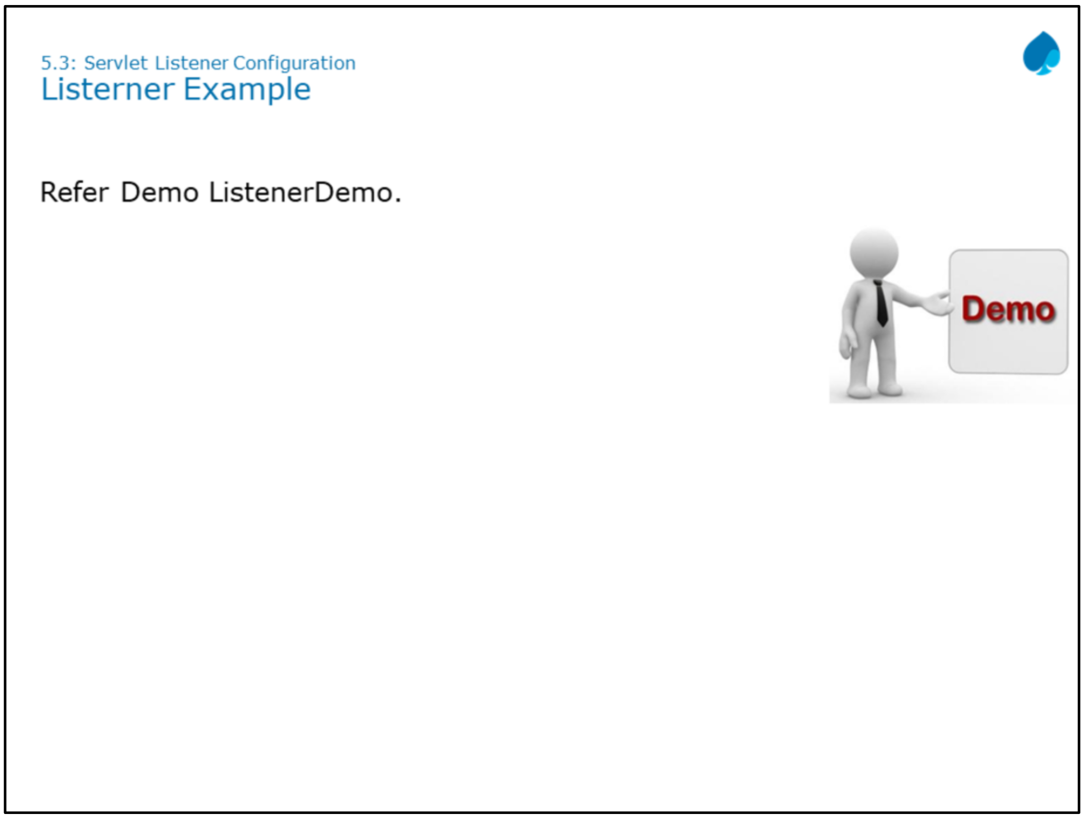
Copy

```
<listener>
   <listener-class>
   com.journaldev.listener.AppContextListener
   </listener-class>
</listener>
```

Ways of Session Tracking in JEE:
Hidden Form Fields:
In this technique the fields are added to an HTML form which are not displayed in the client's request. The hidden form fields are sent back to the server when the form is submitted. In hidden form fields, the HTML entry will be as shown below:

This means that when form is submitted ,the specified name and value will be get included in get or post method.

Instructor Notes:



5.3: Servlet Listener Configuration

Listerner Example

Refer Demo ListenerDemo.

Execute the following servlets:
http://localhost:9090/SessionManagement/login.html

This would lead to servlet:

http://localhost:9090/SessionManagement/HiddenFormServlet

In HiddenFormServlet, username is added as Hidden Field in html form as shown below in partial listing.

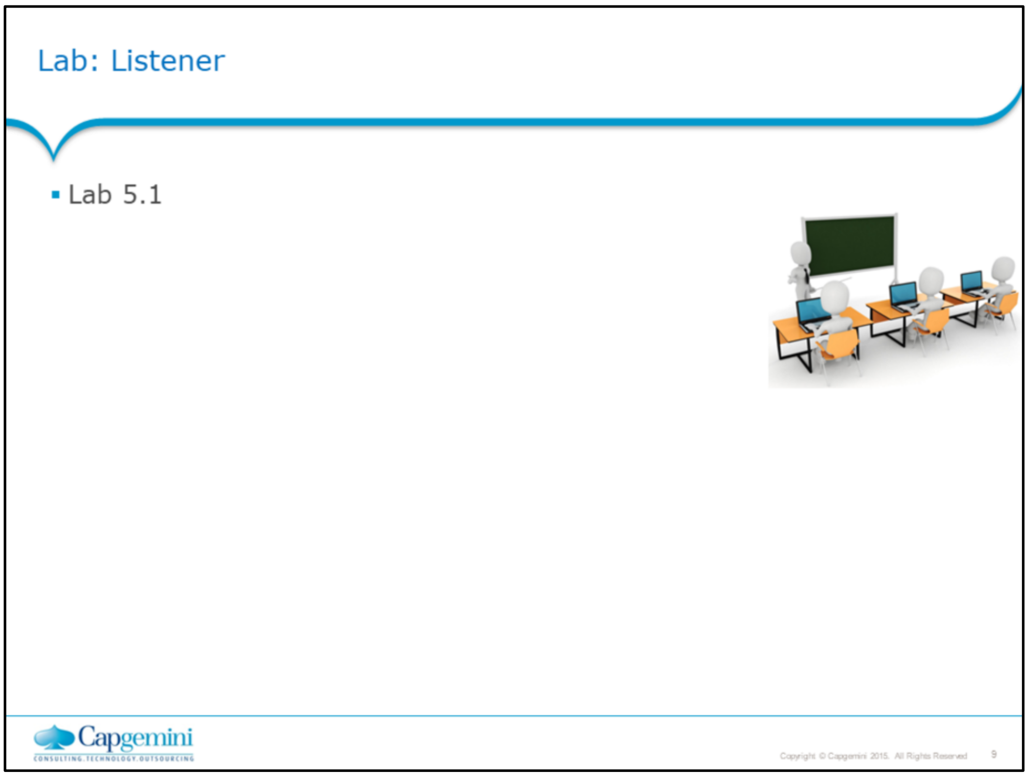
```
out.println("<form action='ShowServlet'>");
out.println("<input type='hidden' name='user' value='" + user + "'>");
out.println("<input type='submit' value='submit' >");
```

Then username is retrieved in ShowServlet.

http://localhost:9090/SessionManagement/ShowServlet?user=John.

Query parameters get appended due to the GET method.

Instructor Notes:



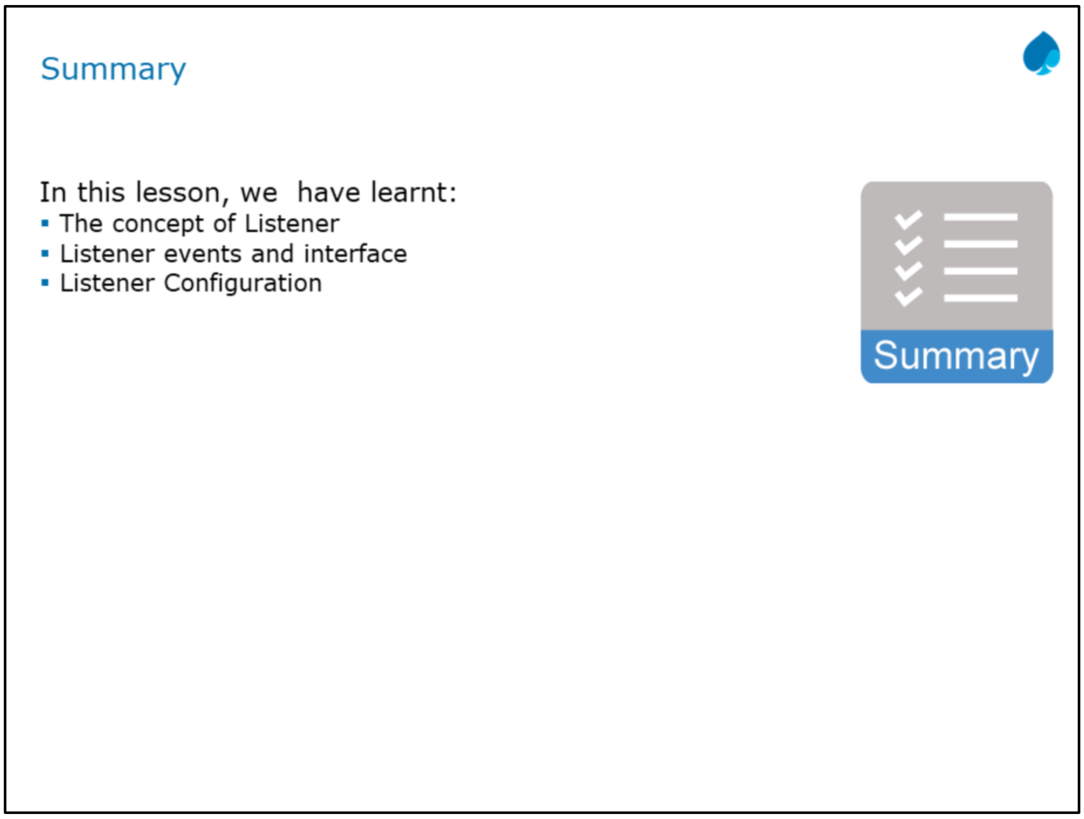
Lab: Listener

- Lab 5.1

9

Instructor Notes:

Summarize the chapter by briefly mentioning what all is learnt.



Summary

In this lesson, we have learnt:

- The concept of Listener
- Listener events and interface
- Listener Configuration

Summary