Coding Bootcamp

simplilearn

Servlets

Life Cycle and Session Tracking

Learning Objectives

By the end of this lesson, you will be able to:

- Analyze the concept of session tracking in Servlets to make the client-server relationship stateful
- Identify the techniques of session tracking to maintain state information about a user across multiple requests
- Illustrate the concept of filters in servlets to build more robust and maintainable web applications
- Utilize cookies in servlets to develop more interactive, stateful, and user-friendly web experiences

TECHNOLOGY

Maintaining User Sessions

simplilearn

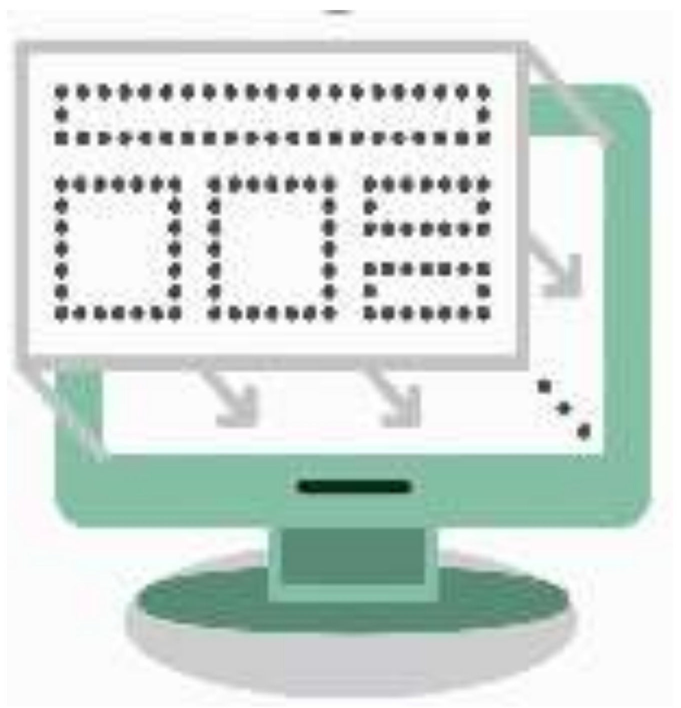
User Sessions

The HTTP protocol and web servers are stateless.

Because the HTTP protocol is stateless, session tracking is required to make the client-server relationship stateful.

Session Tracking

Session tracking is the process of remembering and documenting customer conversations over time.



Session Tracking: Techniques

It employs four different techniques:

1. Cookies
2. Hidden Form Field
3. URL Rewriting
4. HttpSession

HTML Hidden Field

The information is inserted into the web pages via the hidden form field, which is then transferred to the server. These fields are hidden from the user's view.

Syntax

```
<input type = hidden'  name =
'session' value = '12345' >
```

HttpSession

A user session is represented by the HttpSession object. A session is established between an HTTP client and an HTTP server using the HttpSession interface.

Syntax

```
HttpSession session =
request.getSession( );
Session.setAttribute("username",
"password");
```

Session Tracking in Servlets

Problem Statement:

You are given a project to demonstrate the implementation of session tracking in Servlets using cookies.

Outcome:

By completing this project, you will gain a comprehensive understanding of session tracking in Servlets using cookies. You will learn to effectively manage user sessions, track user activities, and enhance the functionality of web applications through practical implementation.

Note: Refer to the demo document for detailed steps:
01_Session_Tracking_in_Servlets

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Set up and authenticate the user
2. Implement session tracking with cookies

TECHNOLOGY

Working with Cookies

Working with Cookies

Web applications use cookies to track sessions and personalize responses.

To do so, start by creating a dynamic web application as shown below:

```
<?xml version="......" encoding="......."?>
<web-app xmlns:xsi="..." xmlns="..." xsi:schemaLocation="...."
id="..." version="...">
<display-name>The use of cookies</display-name>
 <welcome-file-list>
  <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>
  </web-app>
```

Working with Cookies

The home page of the application is **login.jsp**, where the authentication details from the user are logged as shown below:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="US-ASCII">
<title>Login Page</title>
</head>
<body>

<form action="LoginServlet" method="post">
EmailAddress: <input type="email" name="eml">
<br>
Password: <input type="password" name="pwd">
<br>
<input type="submit" value="Login">
</form>
</body>
</html>
```

Working with Cookies

Cookies are set in the response and then forwarded to **success.jsp** to track the session.

The cookie timeout is set to 20 minutes.

Creating a FrontController Design Pattern

Problem Statement:

You are given a project to create a FrontController design pattern and send a request to the FrontController servlet.

Outcome:

By completing this project and implementing the FrontController design pattern to send requests to the FrontController servlet, learners will gain a comprehensive understanding of how to centralize request handling, enhance modularity, and improve maintainability in web applications.

Note: Refer to the demo document for detailed steps:
02_Creating_a_FrontController_Design_Pattern

Assisted Practice: Guidelines

Steps to be followed are:

1. Create a FrontController Servlet
2. Send a request to the FrontController Servlet
3. Create a method in the FrontController Servlet
4. Update responses in the FrontController Servlet

TECHNOLOGY

Filters in Servlets

Filters: Overview

A filter is an object used throughout the pre- and post-processing stages of a request.

Filters: Overview

By default, filters apply pre- or post-processing to requests and responses for a group of servlets, as the user desires.

Filters: Benefits

01 It can be plugged in.

02 The filter is not reliant on a third-party resource.

03 It requires little maintenance.

Filters: Methods

Every filter must implement three methods inside the javax.servlet.filter interface.

init()

destroy()

doFilter()

Extend a class that implements them.

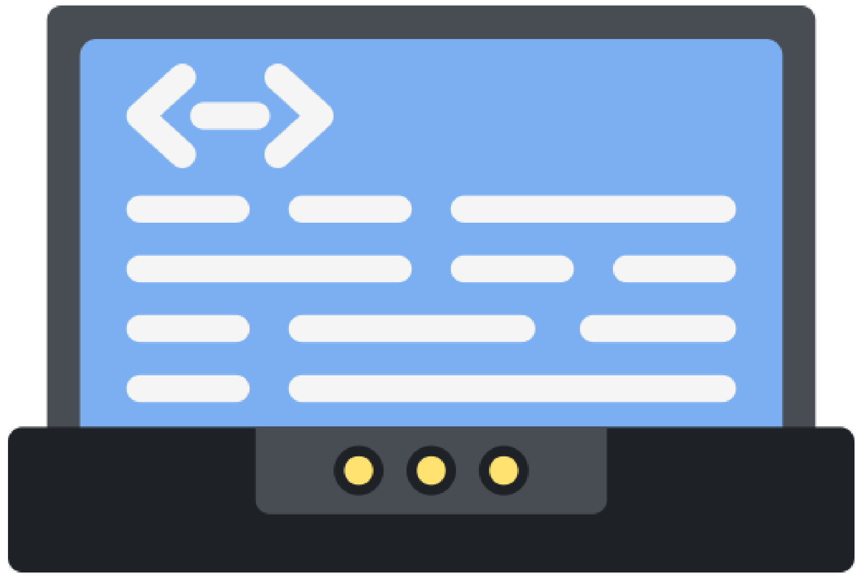
Filters: Example

The generic example for the filter is shown below:

```
MyFilter Class:
Package com.example.filter;
import javax.servlet.*;
public class MyFilter implements javax.servlet.Filter {
public FilterConfig filterConfig;

  public void doFilter(final ServletRequest request,
final ServletResponse response, FilterChain chain)
      throws java.io.IOException, javax.servlet.ServletException {
    chain.doFilter(request,response);
  }

  public void init(final FilterConfig filterConfig) {
    this.filterConfig = filterConfig;
  }

  public void destroy() {
  }
}
```

Filters: Example

Save this code in a file known as MyGenericFilter.java in the package directory.



How to Implement a Filter?

Users can implement a filter by writing the filter logic in the doFilter() method.

```
The ProductFilter class will extend the MyFilter class as shown:

import javax.servlet.*;

public class ProductFilter extends MyFilter {
  private FilterConfig Product Filter;

  public void doFilter(final ServletRequest request, final
ServletResponse response, FilterChain chain)
       throws java.io.IOException, javax.servlet.ServletException
{
    out.println("Filter - doFilter Start");
    request.setAttribute("productTitle","iPhone 13 Pro");
    chain.doFilter(request,response);
    out.println("Filter - doFilter End");
  }
}
```

simplilearn

Filters: XML Configuration

The XML configuration for the filter in the web.xml file is shown below:

```
 <filter>
    <filter-name>MyFilter</filter-name>
    <filter-
class>com.example.estore.ProductFilter</filter
-class>
  </filter>
  <filter-mapping>
    <filter-name>MyFilter</filter-name>
    <url-pattern>/filter.jsp</url-pattern>
  </filter-mapping>
```

Filters

Problem Statement:

You are given a project to explore filters by creating a new filter and working on the login filter's response, pre-processing, and post-processing.

Outcome:

By completing this project, you will gain a comprehensive understanding of filter creation and manipulation within a login system. You will learn how to modify filter responses, implement pre-processing techniques, and execute post-processing operations effectively.

Note: Refer to the demo document for detailed steps: 03_Filters

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Create a new filter
2. Work with mandatory methods
3. Work with a filter mechanism
4. Work on the login filter’s response
5. Make a filter for login parameters

Key Takeaways

- Session tracking is the process of remembering and documenting customer conversations over time.
- A user session is represented by the HttpSession object.
- A filter is an object used throughout the pre- and post-processing stages of a request.
- Web applications use cookies to track sessions and personalize responses.

TECHNOLOGY

Thank You