Web Component Development with Servlet & JSP Technologies (EE 6)

Module-4: The Servlet's Environment

Team Emertxe

Objectives

Upon completion of this module, you should be able to:

- Describe the environment in which the servlet runs
- Describe HTTP headers and their function
- Use HTML forms to collect data from users and send it to a servlet
- Understand how the web container transfers a request to the servlet
- Understand and use HttpSession Object

Relevance

Discussion – The following questions are relevant to understanding what technologies are available for developing web applications and the limitations of those technologies:

- What environment does the servlet run in?
- What are HTTP headers, and what are they for?
- How can you get data from a browser to a server?
- How can a server recognize a browser that is making successive calls?

Http Get Method

One of the two most common HTTP methods is the GET request. A GET method is used whenever the user clicks a hyperlink in the HTML page currently being viewed. A GET method is also used when the user enters a URL into the Location field (for Netscape NavigatorTM and FireFox) or the Address field (for Microsoft Internet Explorer). While processing a web page, the browser also issues GET requests for images, applets, style sheet files, and other linked media.

HTTP Request

The request stream acts as an envelope to the request URL and message body of the HTTP client request. The first line of the request stream is called the request line. It includes the HTTP method (usually either GET or POST), followed by a space character, followed by the requested URL (usually a path to a static file), followed by a space, and finally followed by the HTTP version number. The request line is followed by any number of request header lines.

EMERTXE

Http Request Stream Example

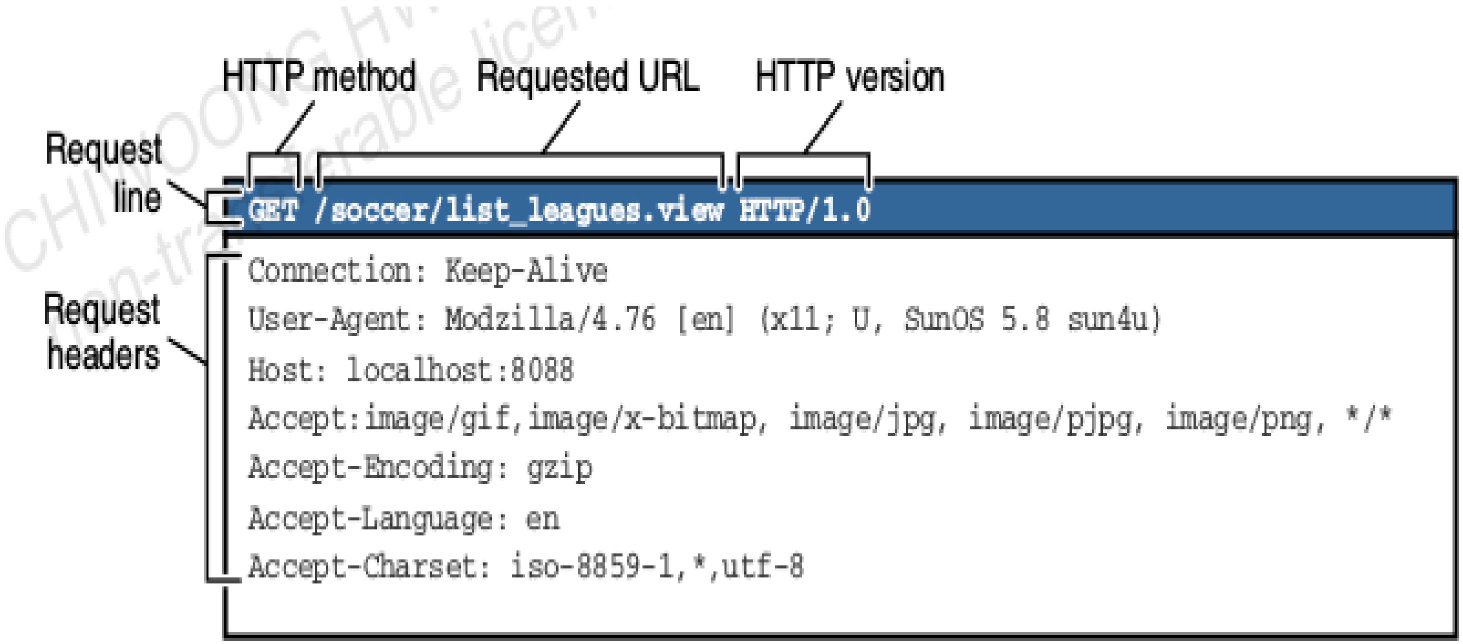


Figure 4-2 HTTP Request Stream Example

HTTP Request Headers

Table illustrates some of the HTTP headers that the browser can place in the request. The headers could then be used to determine how the server processes the request.

Header	Use
Accept	The MIME types the client can receive
Host	The Internet host and port number of the resource being requested
Referer	The address from which the Request-URI was obtained
User-Agent	The information about the client originating the request

HTTP Response

The response stream acts as an envelope to the message body of the HTTP server response. The first line of the response stream is called the status line. The status line includes the HTTP version number, followed by a space, followed by the numeric status code of the response, followed by a space, and finally followed by a short text message represented by the status code.

HTTP Response Headers

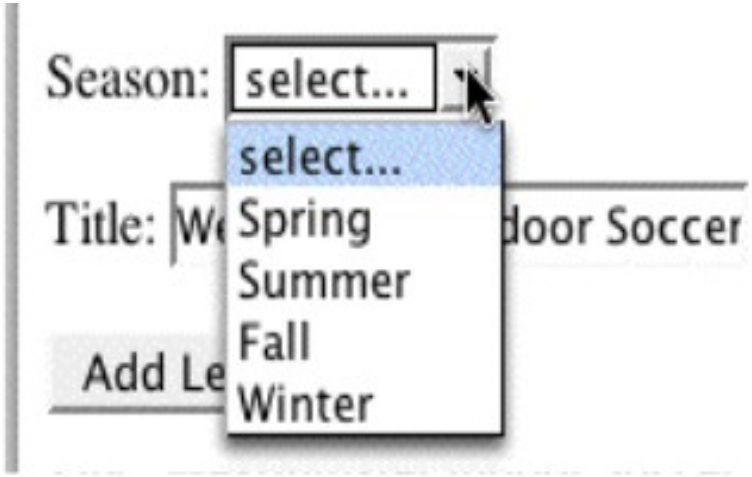
Table illustrates some of the HTTP headers that the server can place in the response. The headers could then be used to determine how the browser processes the response.

Header	Use
Content-Type	A MIME type (such as `text/html`), which classifies the type of data in the response
Content-Length	The length (in bytes) of the payload of the response
Server	An informational string about the server that responded to this HTTP request
Cache-Control	A directive for the web browser (or proxies) to indicate whether the content of the response should be cached

Input Types


```
<p>Year: <input type='text' name='theYear'/></p>
```

Drop-Down List Component



Drop-Down List Component

```
<select name='season'>
Season:
<option value='UNKNOWN'>select...</option>
<option value='Spring'>Spring</option>
<option value='Summer'>Summer</option>
<option value='Fall'>Fall</option>
<option value='Winter'>Winter</option>
</select> <br/><br/>
```

EMERTXE

An Example HTML Form

```
 <form>
First name:<br>
<input type="text" name="firstname">
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

How Form Data Are Sent in an HTTP Request

Syntax:

fieldName1=fieldValue1&fieldName2=fieldValue2

Example:

Username= Fred&password=C1r5z

HTTP GET Method Request

Form data are carried in the URL of the HTTP GET request:

```
GET /admin/add_league.do
year=2003&season=Winter&title=Westminster+Indoor
```

HTTP POST Method Request

Form data is contained in the body of the HTTP request:

```
POST /admin/add_league.do HTTP/1.1
```

HTTP GET Methods

The HTTP GET method is used in these conditions:

- The processing of the request is idempotent.

  This means that the request does not have side-effects on the server or that multiple invocations produce no more changes than the first invocation.

- The amount of form data is small.

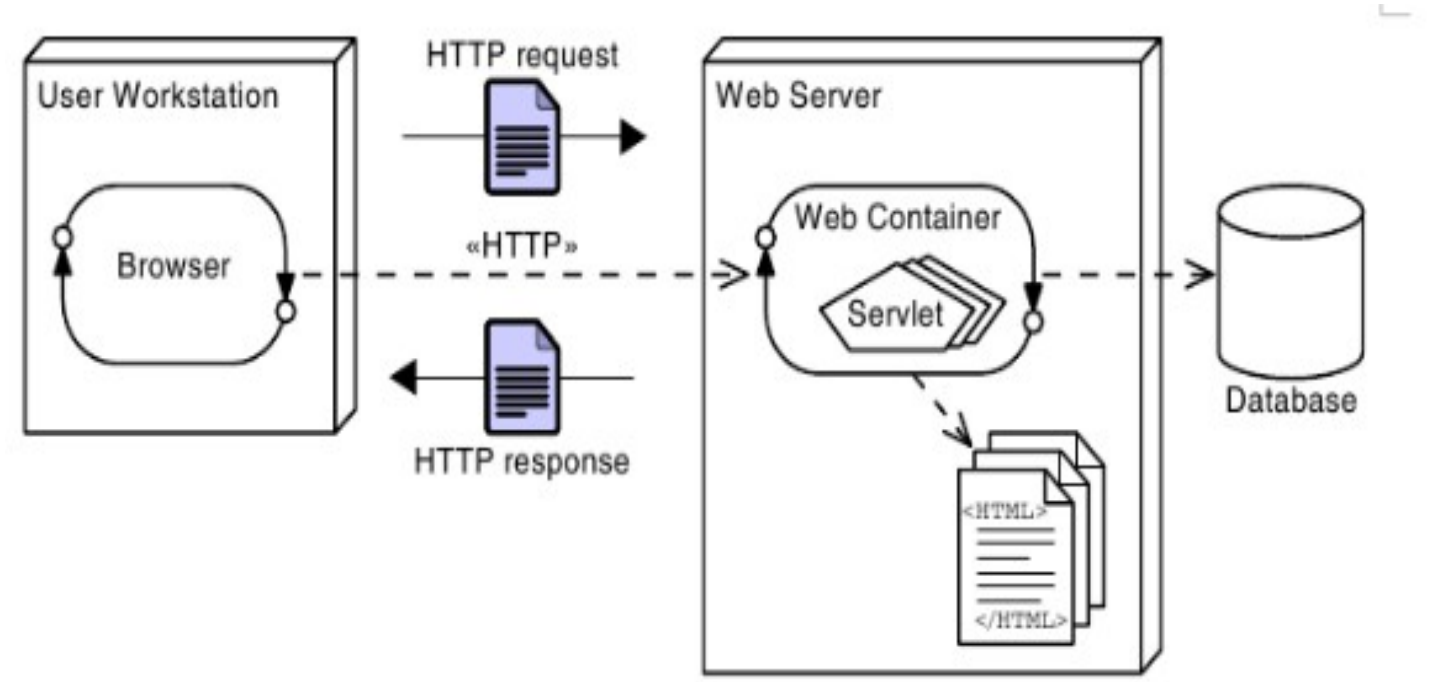
- You want to allow the request to be bookmarked along with its data.

HTTP Post Method

The HTTP POST method is used in these conditions:

- The processing of the request changes the state of the server, such as storing data in a database.
- The amount of form data is large.
- The contents of the data should not be visible in the URL (for example, passwords)

Web Container Architecture



Request and Response Process

Browser Connects to the Web Container:

The first step in this process is the web browser sending an HTTP request to the web container. To process a request, the browser makes a TCP socket connection to the web container.

Web Container Objectifies the Input/Output Streams:

Next, the web container creates an object that encapsulates the data in therequest and response streams. These two objects represent all of the information available in the HTTP request and response streams.

Request and Response Process

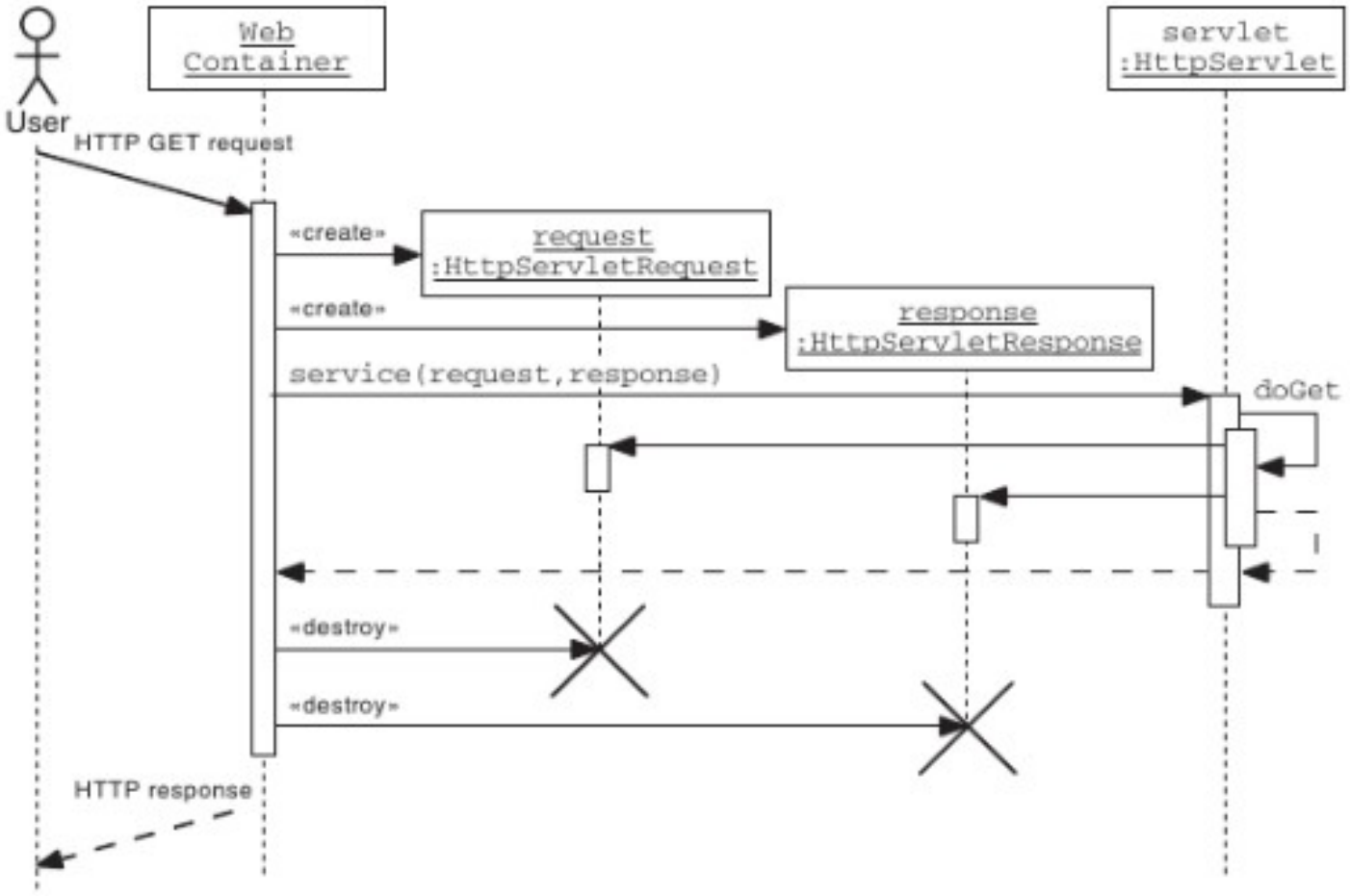
Web Container Executes the Servlet:

The web container then executes the service method on the requested servlet. The request and response objects are passed as arguments to this method. The execution of the service method occurs in a separate Thread.

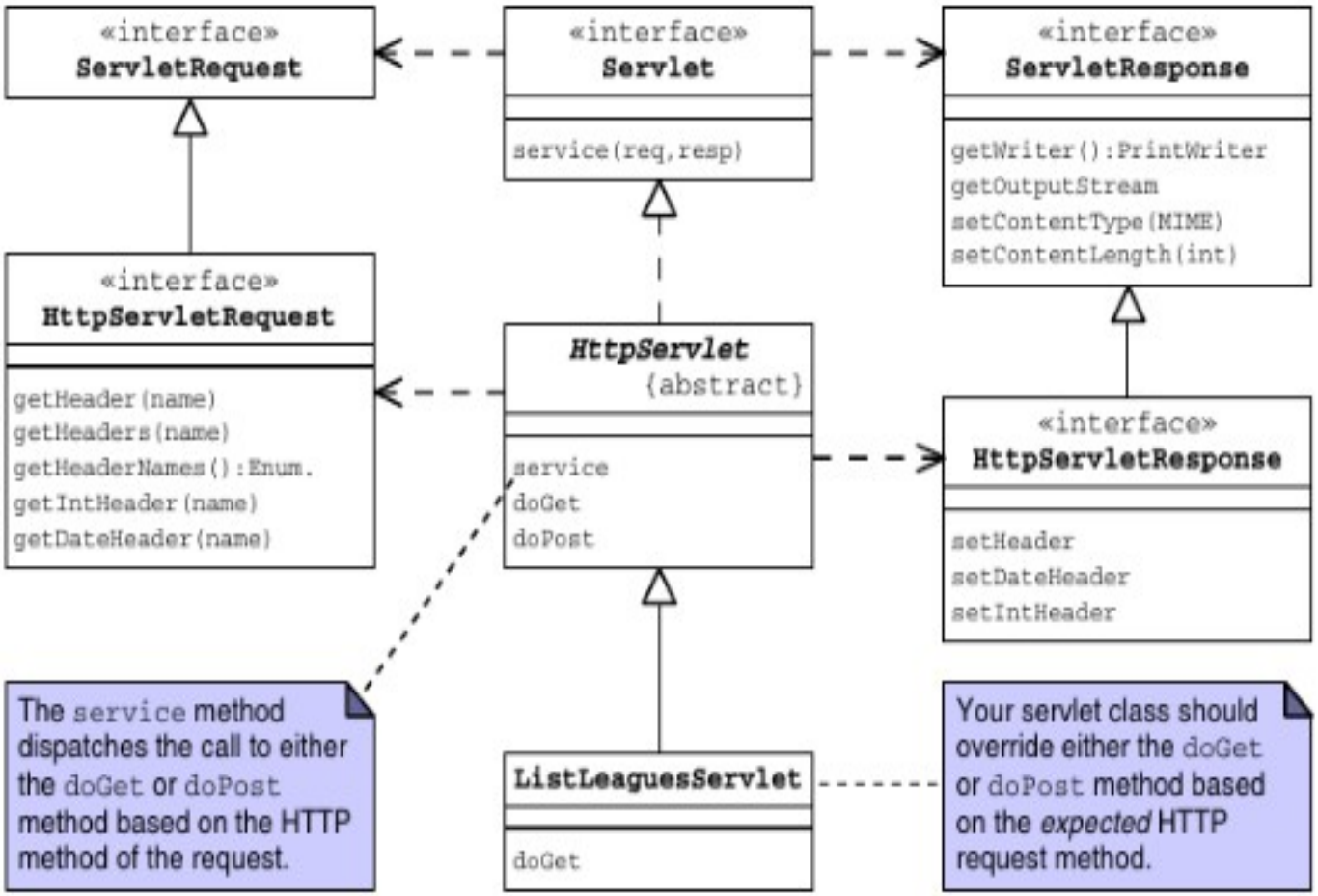
Servlet Uses the Output Stream to Generate the Response:

Finally, the text of the response generated by the servlet is packaged into an HTTP response stream, which is returned to the web browser.

Sequence Diagram of an HTTP GET Request



The HttpServlet API



HttpServletRequest API

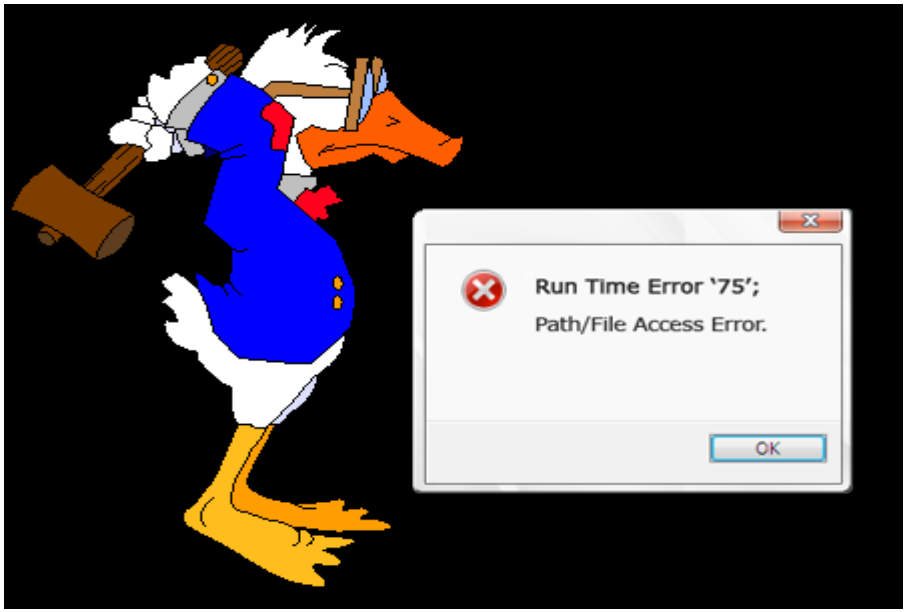
The HTTP request information is encapsulated by the HttpServletRequest interface. The getHeaderNames method returns an enumeration of strings composed of the names of each header in the request stream. To retrieve the value of a specific header, you can use the getHeader method. This method returns a String. Some header values might represent integer or date information. There are two convenience methods, getIntHeader and getDateHeader, that perform the conversion for you.

HttpServletResponse API

The HTTP response information is encapsulated by the HttpServletResponse interface. You can set a response header using the setHeader method. If the header value you want to set is either an integer or a date, then you can use the convenience methods setIntHeader or setDateHeader.

Handling Errors in Servlet Processing

You throw an exception from a doXxx() servlet method, you should wrap the exception in either a ServletException, or an UnavailableException. The ServletException is used to indicate that this particular request has failed. UnavailableException indicates that the problem is environmental, rather than specific to this request.



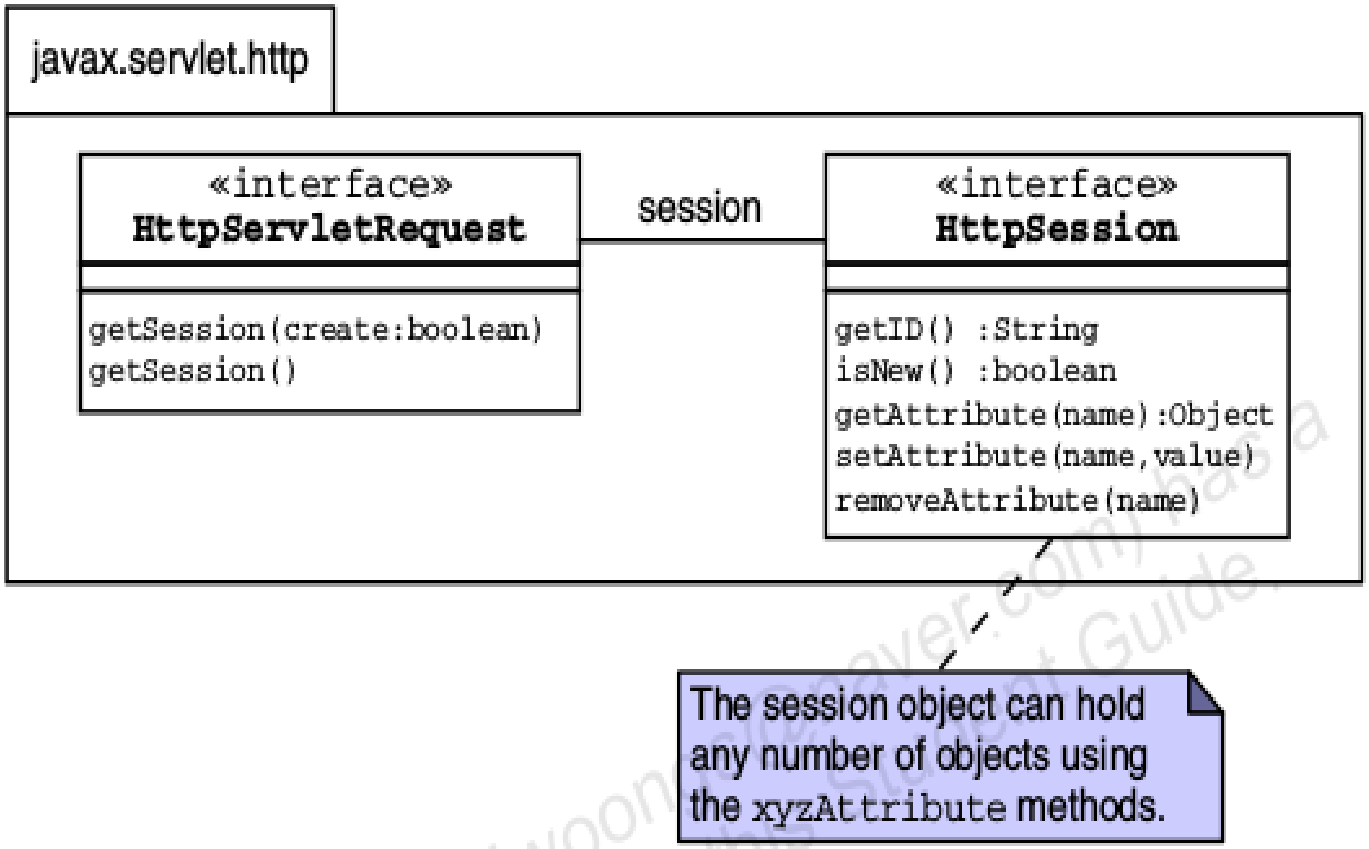
EMERTXE

The HTTP Protocol and Client Sessions

- HTTP is a stateless protocol. That means that every request is, from the perspective of HTTP, entirely separate from every other. There is no concept of an ongoing conversation. This provides great potential for scalability and redundancy in servers, because any request can go to any one of a collection of identically equipped servers.

- Web servers in a Java EE environment are required to provide mechanisms for identifying clients and associating each client with a map that can be used to store client specific data. That map is implemented using the HttpSession class.

The HttpSession API



Storing Session Attributes

To store a value in a session object, use the setAttribute method. For example, if a servlet has prepared an object referred to by a variable called league, this could be placed in the session using the following code:

```
HttpSession session = request.getSession();
session.setAttribute("league", league);
```

Retrieving Session Attributes

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

League league = (League)session.getAttribute("league");
```

Closing the Session

When the web application has finished with a session, such as if the user logs out, you should call the invalidate method on the HTTPSession object. If you fail to invalidate the session, you might clutter up the memory of the server with lots of unnecessary objects. Such memory clutter is detrimental to performance and scalability.

Note – In a secured environment, the invalidate method is likely to delete authentication information forcing, the user to login again.

Additional Session Methods

SetMaxInactiveInterval method

Session Configuration

```
<web-app ...>
<session-config>
<session-timeout>30</session-timeout>
<tracking-mode>SSL</tracking-mode>
</session-config>
</web-app>
```

Using Cookies for Client-Specific Storage

Cookies allow a web server to store information on the client machine. Data are stored as key-value pairs in the browser and are specific to the individual client.

Using Cookies for Client-Specific Storage

- Cookies are sent in a response from the web server.
- Cookies are stored on the client's computer.
- Cookies are stored in a partition assigned to the web server's domain name. Cookies can be further partitioned by a path within the domain.
- All cookies for that domain (and path) are sent in every request to that web server.
- Cookies have a life span and are flushed by the client browser at the end of that life span.

Using Cookies Example

In your servlet, you could use the following code to store that cookie:

```
String name = request.getParameter("firstName");
Cookie c = new Cookie("yourname", name);
response.addCookie(c);
```

Later, when the visitor returns, your servlet can access the yourname cookie using the following code:

```
Cookie[] allCookies = request.getCookies();
for ( int i=0; i < allCookies.length; i++ ) {
if ( allCookies[i].getName().equals(“yourname”) ) {
name = allCookies[i].getValue();
}
}
```

Using URL-Rewriting for Session Management

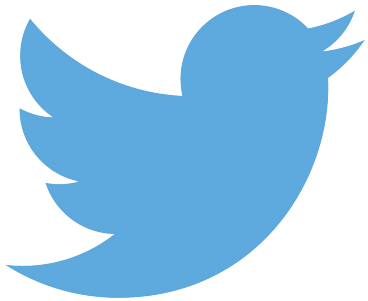
URL-rewriting is an alternative session management strategy that the web container must support. Typically, a web container attempts to use cookies to store the session ID. If that fails (that is, the user has cookies turned off in the browser), then the web container tries to use URL-rewriting.

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe



https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

EMERTXE

Thank You