

**Instructor Notes:**

Add instructor  
notes here.



**Instructor Notes:**

Explain the lesson coverage

## Lesson Objectives



In this lesson, we will learn:

- Process GET and POST Requests from Web Clients
- Retrieve Parameters from HTML Forms
- Retrieve path information
- Retrieve request headers

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

2

Lesson Objectives:

This lesson introduces the Servlet request object. The lesson contents are:

Lesson 04: Request Object

- 4.1: Processing Get and Post Requests from Web Clients
- 4.2: Retrieve Parameters from HTML Forms
- 4.3: Retrieving Path Information
- 4.4: Retrieve Request Headers

**Instructor Notes:**

Introducing GET / POST methods. Their usage and differences need to be explained here.

## 4.1: Processing Get and Post Requests from Web Clients

### Processing Request: GET / POST



#### HTTP GET / POST Requests

- When a client sends a request for processing it could be either GET / POST HTTP method
- If no method is specified it defaults to be GET. These methods are exposed from the HTTP specification.
- In the GET method the parameters would be appended in URL. Thus there would be no security as parameters would be exposed in URL
- To use the POST method it must be explicitly stated in the HTML page in the form tag
- In the POST method the parameters would be appended in the request body. Thus the data would be secured as it is not exposed in URL.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

3

Need to explain that 2 methods could be used in Servlet to process the request (GET / POST).

The methods are present in the HTTP specification which are exposed in Servlet API.

Differences between them being:

GET : not secured and POST: secured

GET: data gets appended in URL and POST: data is appended as a part of request body

GET: limitation of data to be appended and POST: unlimited data could be appended

Other methods of the Http Specification are:

PUT : Replaces all current representations of the target resource with the uploaded content

HEAD : Same as GET, but transfers the status line and header section only.

DELETE : Removes all current representations of the target resource given by a URI.

TRACE : Performs a message loop-back test along the path to the target resource

OPTIONS : Describes the communication options for the target resource

CONNECT : Establishes a tunnel to the server identified by a given URI

**Instructor Notes:**

We shall be seeing each of these methods in our demo coming next

## 4.2: Retrieve Parameters from HTML Forms

### Processing Request: Handling Form data



The servlet receives parameters using these methods of ServletRequest object:

- `public String ServletRequest.getParameter (String name)`
- `public String[] ServletRequest.getParameterValues (String name)`
- `public Enumeration ServletRequest.getParameterNames()`
- `public String ServletRequest.getQueryString()`

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved.

4

#### Processing Get and Post Requests from Web Clients:

Request – Information / data / parameters sent from the client (browser) to Server for processing and formulating a response.

When a servlet accepts a request from a client (`service(Request , Response)` method), the container creates and hands over two objects, `ServletRequest` and `ServletResponse`. The `ServletRequest` object encapsulates the communication from the client to the server, while the `ServletResponse` object encapsulates the communication from the server back to the client.

Since we are using HTTP protocol, the container would handover `HttpServletRequest` and `HttpServletResponse` objects to the `service()` method of `HttpServlet` class. This is done by downcasting `ServletRequest` to `HttpServletRequest` and `ServletResponse` to `HttpServletResponse`.

#### Request parameters:

The `ServletRequest` (or `HttpServletRequest` object if using `HttpServlet`) contains a lot of information. We shall see how servlet processes form parameters.

`public String ServletRequest.getParameter (String name)`: This returns the value of the named parameter as a `String` or `null` if parameter wasn't specified.

`public String[] ServletRequest.getParameterValues (String name)`: It returns values of named parameter as a `String` array if parameter could have multiple values, as in the case of list element with multiple select.

`public Enumeration ServletRequest.getParameterNames()`: It returns a list of parameter names.

`public String ServletRequest.getQueryString()`: It returns the raw query string of the request.

**Instructor Notes:**

Run the  
ProcessRequest  
Servlet and  
userInput.html to  
show how to  
retrieve form  
parameters

### Demo : Handling Form data



Execute the ProcessRequest servlet that reads form data.

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved.

5

**Note:**

Refer the com.igate.ch4.ProcessRequest.java class and userInput.html for data to be submitted from html page.

Invoke this servlet as follows:

<http://localhost:9090/RequestResponseDemoApp/userInput.html>.

Make appropriate choices and key in data. On form submit, the ProcessRequest servlet retrieves all form parameters and displays them.

**Instructor Notes:**

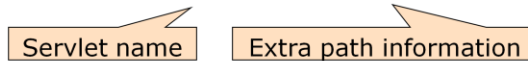
### 4.3: Retrieving Path Information

#### Methods for Retrieving Path Information



The servlet can use the `getPathInfo( )` method to get extra path information:

- `public String HttpServletRequest.getPathInfo( )`
- Example: `http://server:port/servlet/ViewFile/index.html`



The `getPathTranslated( )` method returns the extra path information translated to a real file system path or null if there is no extra path information.

- `public String HttpServletRequest.getPathTranslated( )`

#### Retrieving Path Information:

In addition to parameters, an HTTP request can include something called “extra path information”. This extra path information can be used to indicate a file on the server or some piece of information that the servlet can use for something. This path information is encoded in the URL of an HTTP request. An example URL looks as follows:

`http://server:port/servlet/ViewFile/index.html`

This invokes the `ViewFile` servlet, passing “/index.html” as extra path information. A servlet can access this path information using the `getPathInfo( )` method. `String pathname = request.getPathInfo();`

If the extra path information is the name of a file, then a servlet can get the actual file system location of the file using `getPathTranslated( )` method:

`String filename = request.getPathTranslated();`

This method returns the extra path information translated to a real file system path or null if there is no extra path information.

Demo for this would be seen later during the File Upload concepts discussion.

**Instructor Notes:**

Emphasize that request headers are not part of servlet. They are sent automatically by browser. Explain with few examples how such information can be useful to servlet. For example, if servlet knows the encoding scheme that browser understands, it will send response encoded accordingly!

#### 4.4: Retrieve Request Headers

##### Concept of Request Headers



When browser makes a request, it also sends HTTP information in the form of request headers.

- E.g. :- Accept, Accept-Language, Accept-Encoding etc

To access header information :

- `public String HttpServletRequest.getHeader(String name)`
- `public Enumeration HttpServletRequest.getHeaderNames( )`

#### Request Headers:

When the client makes a request it also sends some HTTP information in the form of request headers. These are indirectly set by the browser, contain some extra information about the request and can be used by servlet to customize responses to the browser. Some examples of HTTP request headers are given below:

Accept: It specifies the MIME type that client accepts

Accept-Language: It specifies the language(s) that client can receive

Accept-Encoding: It specifies the encoding format that client can use

User-Agent: It gives information about client software

Header information can be accessed through the request object using several methods:

`public String HttpServletRequest.getHeader(String name)`: It returns the value of the header as a String or null if header was not sent as part of request.

`public String HttpServletRequest.getDateHeader(String name)`: It returns the value of the specified request header as a long value that represents a Date object.

`public String HttpServletRequest.getIntHeader(String name)`: It returns the value of the specified request header as an int.

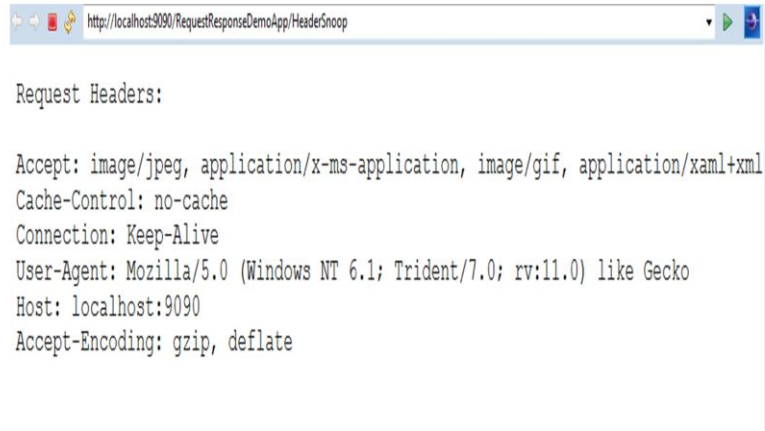
`public Enumeration HttpServletRequest.getHeaderNames( )`: It returns an enumeration of all the header names this request contains.

**Instructor Notes:**

Run the HeaderSnoop servlet to show how to retrieve request headers

### Demo : Request Headers

Execute the HeaderSnoop servlet that reads and displays request headers



```
Request Headers:

Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml
Cache-Control: no-cache
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Host: localhost:9090
Accept-Encoding: gzip, deflate
```

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved.

8

**Note:** Refer `com.igate.ch4.HeaderSnoop.java` class.

Invoke this servlet as follows:

`http://localhost:9090/RequestResponseDemoApp/HeaderSnoop`

Output could be seen as shown in the figure given in the above slide.



**Instructor Notes:**

## Summary



In this lesson, we have learnt:

- Process GET and POST Requests from Web Clients
- Retrieve Parameters from HTML Client Forms
- Retrieve path information
- Retrieve request headers

Instructor Notes:

Answers for Match the Following:

Please give participants about 5 minutes to refer to Appendix-B and extra API's for the Request object. Then only conduct review quiz on this slide.

- 1-b
- 2-d
- 3-e
- 4-c
- 5-f
- 6-a

Review – Match the Following

1.getPathInfo()	A.Returns the means by which HTTP request was made
2.getQueryString()	B.Returns extra path info
3.getRemoteHost()	C.Translates given path into real path
4.getRealPath()	D.Returns string following the URL
5.getContentType()	E.Returns the fully qualified name of the client
6.getMethod ()	F.Returns the MIME type of the body of the request

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

10

To Participants : refer to Appendix-B and extra API's for the Request object.