**What determines whether Browser sends GET or POST requests:**

1. **Typing a URL in Browser makes request to contain GET method**
2. **Clicking on a Hyper link in Browser makes request to contain GET Method**
3. **Submitting the form with method= “get” form attribute in Browser makes request to contain Get method.**
4. **Submitting the form with method = “post” form attribute in Browser makes request to contain “POST” method.**
5. **Submitting the form with “No method form attribute declaration “ in Browser makes request to contain “GET” method.**

Note:

* Depending on the HTTP method present in the request corresponding doXXX()) method get executed at Servlet side
* If a form collects
* “Sensitive Data” like Passowrd (ex: Gmail Login Page/ Profile Creation Page)

OR

* Very Large Data like sending mail (ex: Gmail Compose Mail option)

Then that form should use method –“post”

* If a form collects “In-Sensitive Data” like search words (ex: Google Search) then that form may use method=”post” or method=”get”
* Hence whenever Servlet gets a request via Submitting the form then generally we override doPost() method & for rest of the cases, we have override doGet() method.

**Differences between GET/doGet() & POST/doPost()**

|  |  |
| --- | --- |
| GET/ doGet | POST/ doPost() |
| GET method allows to get the data from server | POST method allows to Post data of unlimited size to the server. |
| GET is a default method | POST is not a default. We have to explicitly declare method =”POST” . |
| GET requests “do not have a Body” OR “have Empty Body” | POST requests “do have a body” |
| Hence incase of GET , Form Data will be present in HEADER IN THE FORM OF Query String. | In case of POST, Form Data will be present in Body. |
| Insecure; because form data get exposed to the outside world | Secure; because form data will be present in the Body & hence it will not be exposed to the outside world |
| The amount of data sent using GET is restricted because URL can contain Only limited characters. | There is no restriction on the amount of data sent using the POST method. |
| We cannot send the files using GET | We can entire files using POST for Ex: Resume Upload, video files Etc. |
| GET requests, by default, they are “idempotent”. i.e we can perform the same operation again & again without any side effects. | POST requests are “Non-idempotent” in nature. |
| We “can bookmark” the GET requests | We “cannot bookmark” POST requests. |

**Servlet Container:**

* Servlet Container is a sub-component of web server that helps both web server & servlet to communicate with each other.
* As the name implies, all Servlets of dynamic web application are directly under the control of Servlet Container.

**HOW**

**Servlet Container Works:**

1. Whenever request comes, web server hand over the complete request to servlet container
2. Container by looking at the URL present in the request & referrring web.xml it comes to know the servlet which handles that request.
3. Container then “creates an instance” of that Servlet.
4. Once Instance creation is successful then it converts the “Raw HTTP Request” to a Java Object of type “HttpServletRequest” & also creates “HttpServletResponse” object.
5. Depending on the HTTP Method present in the request, container invokes corresponding doXXX() method by passing these request & response objects
6. doXXXX() method Once execution is over, container converts the response object to “Raw HTTP Response” & gives it back to web server
7. Once the response has been give back, Servlet Container garbage collects the request & response objects.
8. In other words for every request, container creates new request & response objects.
9. i.e the Life Span of these Objects is Created: once request comes to Servlet Destroyed: once response is given back .

**Advantages of Servlet Container:**

1. **Communication support:**

Container helps both web server & servlet to communicate with each other.

1. **Multithreading Support:**

Container automatically creates a new thread for every incoming request.

1. **Declarative Support**

With “web.xml” which is used by Servlet container we can change the behavior of web application without changing anything in Servlet/JSP Code

1. **Life Cycle Management:**

Container manages/controls the Life Cycle of a Servlet”

1. **JSP Supprt:**

Container takes care of converting JSP into a Servlet.

Assingment: create a HTML Form as Name is : passwd: gender: education : tech language ; I have about me

**Javax.servlet.http.HttpServletRequest**

* “HttpServletRequest” object , in short called as “Request Object”, is an Object representation of “Raw HTTP Request”
* We should make use of this object to get Information from Request.
* HttpServletRequest is an Interface & it extends another Interface by name “javax.servlet.ServletRequest”.
* Request Object has many “Getter Methods” which helps us to get the information from “Raw Http Request”

**Some Methods which are part of “Request Object” are:**

1. **String HttpServletRequest.getMethod()**

This methods returns the HTTP Method present in the request as a String Value

1. **StringBuffer HttpServletRequest.getRequestURL()**

This method returns the URL present in the request as a StringBuffer.

1. **String ServletRequest.getProtocol()**

This method returns the Protocol preseent in the request as a String Value.

1. geString ServletRequest.**getParameter**(String name)
2. String[] ServletRequest.**getParameterValues**(String name)

* Both the above Methods helps us to get the “Form Data/Query String information “ form “Request Object”
* Both these methods return Null if the parameter name does not exist.

**javax.servlet.http.HttpServletResponse**

* “HttpServletResponse” Object, in short called as “Response Object”, is an Object Representation of “Raw HTTP Response”
* We should make use of this object to send Info as part of “Raw HTTP Response”
* HttpServletResponse is an Interface & it extends an another Interface by name “javax.servlet.ServletResponse”
* Response Object has Methods which helps us to send the information as part of “Raw HTTP Response”

**Methods in Response Object:**

1. void ServletResponse.setContentType(String contentType)

* “setContentType()” Method present in Response object helps us to set the Content Type Info or Mime Type

1. PrintWriter ServletResponse.getWriter() throws IOException
2. void PrintWriter.println(String response)

void PrintWriter.print(String response)

* These methods helps us to provide “Actual Content” info in Response Object
* First we should get “java.io.PrintWriter” from Response Object by invoking a method by name “getWriter()”
* PrintWriter has “print()/println()” methods which helps us to add Actual Content to Response Object.

Note:

* PrintWriter is a “Concrete Class” but we SHOULD NOT create our own instance of this class , instead we Should get it from Response Object.
* Between “print()” & println()” methods print() reduces the size of the Actual Content there by increases the Performance.

1. void HttpServletResponse.sendError(int statusCode, string errMsg) throws IOException

* This method helps us to send the Error Response.

**Assignments 2:**

* **Create a HTML Form as CreateProfile.html**
* **Create Servlet by name “CrateProfileServlet” which gets the request from this form,**

1. **Get the form data**
2. **Strore the form data into corresponding Tables**
3. **Generate Proper Response(Success/ Error Message)**

**eXtensible Markup Language(XML) Introduction:-**

* implies As the name it’s an extension of HTML & this Language helps to Transfer or to store the Data between different Applications.
* XML looks similar to HTML but it’s not a HTML Comparison between HTML & XML.

|  |  |
| --- | --- |
| **HTML** | **XML** |
| 1. **HTML** helps to display the Data in the Browser | 1. XML helps to store and Transfer the Data between Application |
| 1. **HTML has Pre-defined Tags** | 1. **XML has User-defined Tags** |
| 1. **HTML Tags are “Case In-Sensitive”** | 1. **XML Tags are “Case Sensitive”** |
| 1. **First Line of HTML is**   **<!DOCTYPE html---->** | 1. **First Line of XML is**   **<?xml version=”1.0” encoding=”UTF-8”?>** |
| 1. **HTML is “Not Strictly Typed” Language** | 1. **XML is “Strictly Typed Language** |
| 1. **File extension of HTML is “.html/.htm”** | 1. **File extension of XML is “.xml”** |

**NOTE:**

* Hence XML file should have
* “.xml” as file extension
* First line of that file should be

<?xml version=”1.0” encoding=”UTF-8”?>

* **Since XML** CONSIST OF User-defined Tags, These tags information is defined in another file by name “XML Schema Document (XSD).
* **XSD** file will have “.xsd” as a file extension.
* **Hence** every application must obey the rules defined in XSD file while constructing the XML file & reading the data from XSD file.

**Deployment Descripter**

* **It’s kind of “instruction sheet” to a Servlet Container & container always refer this to handle incoming requests**
* **It MUST**
* **Be a XML file**
* **Have the name “web.xml”**
* **Be present inside WEB-INF folder.**
* **Hence every dynamic web application must have ONLY ONE web.xml.**

1. **<welcome-file-list> Tag:-**

* **This tag is used to configure default page for the web application**
* **If no resource name is specified in URL Path, then container searches the resources present in this tag in the order they have declared**

**Example:**

**<welcome-file-list>**

**<!—Static Resource-->**

**<welcome-file>index.html</welcome-file>**

**<!—Dynamic Resource -- >**

**<welcome-file>currentDateTime</welcome-file>**

**</welcome-file-list>**

1. **Configuring a URL for a Servlet:**

* Every Servlet must have a URL & web.xml helps to configure a URL for a Servlet
* Container uses this information to identify a specific servlet to handle a given request
* There must be an at least One Url configured for a Servlet. Also Servlet can have “more than One” URL
* Below are the different ways to configure the URL for a Servlet.

**Below are the different ways to configure the URL for a Servlet**

1. **Exact Matching**
2. **Directory Matching**
3. **Extension/Pattern Matching**

**Example:**

**<!—I. Declaring the Servlet -- >**

**<servlet>**

**<servlet-name>someName</servlet-name>**

**<servlet-class>ffsa.jsdf</servlet-class>**

**</servlet>**

**<!—II. Configuring URL for a Serlet -- >**

**<! – 1. Exact Matching -- >**

<servlet-mapping>

<servlet-name> someName</servlet-name>

<url-pattern>/**firstUrl**</url-pattern>

<servlet-mapping>

**<! – 1. Directory Matching -- >**

<servlet-mapping>

<servlet-name> someName</servlet-name>

<!-- <url-pattern>/abc/\*</url-pattern> -- >

<url-pattern>/**abc/SomeURL**</url-pattern>

<servlet-mapping>

**<! – 1. Pattern Matching -- >**

<servlet-mapping>

<servlet-name> someName</servlet-name>

<!-- <url-pattern>/abc/\*</url-pattern> -- >

<url-pattern>\*.do</url-pattern>

<servlet-mapping>

**Order of Preference**

1. Exact Match
2. Directory Match
3. Extension/Pattern Match