**Servlets**

* It is a thread-based technology, which improves the performance of application.
* For the first request a servlet object is created and then for every other request separate thread will be created.

**A screenshot of a cell phone

Description generated with very high confidence**

**A screenshot of a cell phone

Description generated with very high confidence**

**A picture containing indoor, table, screenshot

Description generated with very high confidence**

**MVC pattern:**

**A close up of a map

Description generated with very high confidence**

**A screenshot of a cell phone on a table

Description generated with high confidence**

**A screenshot of a cell phone

Description generated with very high confidence**

**A close up of text on a white background

Description generated with high confidence**

Http allows 7 types of methods:

* GET
* POST
* HEAD
* OPTIONS
* PUT
* TRACE
* DELETE

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

**A screenshot of a cell phone

Description generated with very high confidence**

**Head:**

The response will have only header information.

**Options:**

To get the information about what all methods are supported by the server.

**Status Codes:**

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence

* When you install server software in your computer server is available in two parts
  + Main server
  + Container
* When you click a button then http protocol will create a ReqFmt and will carry it to Main server. Main server will check the ReqFmt is valid or not. If it is not valid it will send the immediate Error response. If it is valid then ReqFmt is forwarded to container. Then container will check the locater information and will find that particular resource and execute. Then the response is given back to Main Server and then to Protocol. Protocol will create a RespFmt and will carry over the network and back to client.
* Two types of containers as per server-side tech we used to prepare the application.
  + **Web Container**: responsible to execute only web components like servlets and JSP
    - **Servlet Container**: for servlet execution
    - **JSP Container**: for JSP execution.
  + **EJB Container**: responsible for EJB component.
* As per containers physical existence there are three types of containers.
  + **Standalone Container:** Integration of Main server and container
  + **In-Process Container:** Container is a part in Main server
  + **Out-of-process Container:** Container is outside of the Main server.

**Steps to Prepare 1st Web Application:**

**A screenshot of a cell phone

Description generated with very high confidence**

1. **Install server software**
2. **Prepare web Application Directory Structure.**

**A screenshot of a cell phone

Description generated with high confidence**

Directory structure is divided in to two parts:

* Public or client area: under application folder outside of WEB-INF folder
* Private or server area: under WEB-INF folder

1. **Prepare Deployment Descriptor:**
   * It is Web.xml file, it provides metadata about our application that is required to server to identify server-side resources and execute them.
   * In web applications it provides the following details
     1. Display names Configuration
     2. Welcome Files Configuration
     3. Context Parameters Configuration
     4. Servlets Configuration
     5. Filters Configuration
     6. Listener Configuration
     7. Initialization Configuration
     8. Session timeout Configuration
     9. Load on startup Configuration
     10. Error pages Configuration
     11. Taglib Configuration
     12. Security Configuration and many…..
   * **Servlet Configuration is required:**
     1. To define alias name or locators or URL patterns for servlets and to provide mapping between URL patterns and servlet classes we must Configure servlets in **web.xml file**
     2. In order to access private area members from outside we need to define alias names in servlet Configuration.

A screenshot of a social media post

Description generated with very high confidence

**To define url pattern there are 3 methods:**

* **Exact Match method:**
  + <url-pattern>/abc/xyz</url-pattern>
  + Client address bar url should match the servlet configuration.
  + <http://localhost:1010/app/abc/xyz>
  + When we want to access servlets individually then you should you this method.
* **Directory Match method:**
  + <url-pattern>/abc/\*</url-pattern> //ends with Star
  + Anything after /abc is match the servlet.(prefix is enough)
  + When we want to send multiple requests to a particular server side resource then we have to use Directory match method.
  + In general we use this method for filters.
  + Used in MVC based applications.

**A close up of a map

Description generated with high confidence**

* **Extension match method:**
  + <url-pattern>\*.do</url-pattern>
  + Anything that ends with extension do will be allowed.
  + When we want to submit all request to particular server side resource and if we want to perform a particular action based on the url pattern name.
  + Used in MVC based applications.

**A close up of a map

Description generated with very high confidence**

1. **Prepare web Resources like Servlet, JSP’s and HTML:**

* Java servlet API has provided two packages.
  + Javax.servlet
  + Javax.servlet.http package

**A screenshot of a cell phone

Description generated with very high confidence**

**A screenshot of a cell phone

Description generated with very high confidence**

* Three ways to prepare servlets by implementing Servlet interface or by extending abstract class GenericServlet or by extending abstract class HttpServlet.

1. **Start server and deploy web application:**
2. **Access web application from client:**

**Servlet Design:**

**Implementing servlet Interface:**

Public interface Servlet{

**Public void init(ServletConfig config) throws ServletException;**

* Container will perform the life cycle. It will call the init method by passing object of ServletConfig as parameter.
* We do initializations in here

**Public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException**

* Application logic comes here.
* Container will identify the service method and executes.
* Container passes two objects.

**Public servletConfig getServletConfig();**

* The purpose is to return servletConfig object for init method.

**Public String getServletInfo();**

* Information

**Public void destroy();**

* DE instantiation

}

* There is no main method in servlets, so it should have a life cycle
  + Servlet loading
  + Servlet instantiation
  + Servlet initialization -**init**
  + Request processing - **service**
  + Servlet de-instantiation - **destroy**

**Steps to prepare Service() method implementation:**

* Set content type to Response Header.
  + **Public void setContentType(String type)** : //text/html , img/jpg, text/xml.
  + To let know browser about the kind of response
* Get PrintWriter object.
  + To keep response in response object in service method.
  + **Public PrintWriter getWriter()**
* Generate Dynamic Response:
  + **Public void println()**

**Life Cycle:**

**While staring**

* When you start the server at first container will come to web-apps folder and deploy all the applications and prepare separate object for each and every deployed application called as **ServletContext** object.
* At the time of deployment container will check for web.xml file and loads it and parses it, then it stores the application related data in to ServletContext object.

**When Running:**

* When URL is specified in client address bar, then request will come to protocol and it establishes a virtual socket connection between client and server. Then protocol creates a request Format and forwarded to main server. Then main server does validations on ReqFmt and then it forward the request to container.
* Then Container recognizes the servlet application and if resource has any extension like html or jsp then it checks in public area if not it comes to WEB-INF folder and then to web.xml and finds the respective servlet mapping and searches it in classes folder. It also searches in lib folder. If it not found in both places it raises ClassNotFound Exception.
* If it finds Servlet class file, then container will execute life cycle actions.
  1. Servlet loading: servlet takes that class file and loads it in to memory.
  2. Servlet Instantiation: After loading Container creates object for servlet.
  3. Servlet Initialization: Container calls init method with **servlet config object.**

After initialization container will create a thread to execute service method using **thread pooling concept**. For this container creates two objects request and response. By executing service method response is generated in response object and thread will come to dead state. By this container knows execution is completed and response is given back to main server.

Then main server gives it to protocol which in turn creates RespFmt and carry and gives to client, it executes the response and displays.

Then protocol terminates the virtual socket connection, by this container destroys request and response objects and then it goes to waiting state. It again checks for any request if no requests then it destroys servlet object called DE instantiation.

When we shut down or undeployed the application Container will destroy servlet context object.

A close up of a map

Description generated with very high confidence

**Drawbacks:**

1. Irrespective of the actual application requirement we need to implement all methods of servlet interface.
2. **Extending Generic servlet class:**

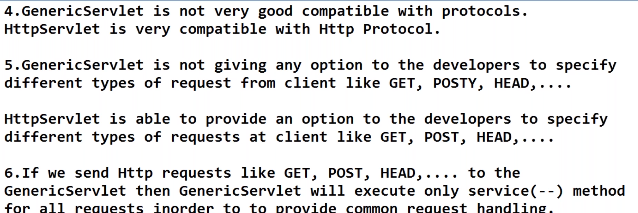
* Using **adapter design pattern.**
* It is an abstract class which has mandatory methods as abstract.
* Service method is abstract.

**A screenshot of a social media post

Description generated with very high confidence**

A screenshot of a cell phone

Description generated with very high confidence



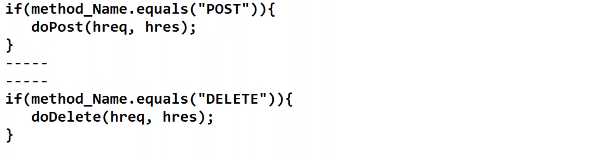
A screenshot of a cell phone on a table

Description generated with high confidence

**HTTP SERVLET:**

A picture containing indoor, text

Description generated with high confidence



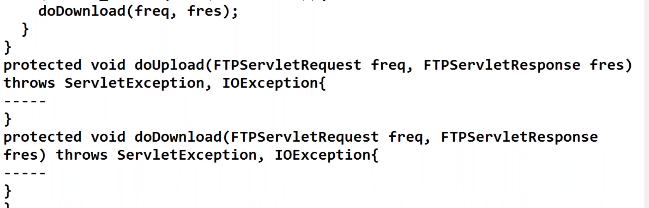
A screenshot of a cell phone

Description generated with very high confidence

Similarly implemented for FTP Protocol:

A screenshot of a cell phone

Description generated with very high confidence



A screenshot of a social media post

Description generated with very high confidence

A close up of a map

Description generated with high confidence