Servlets and JSPs:

https://www.youtube.com/watch?v=tkFRGdUgCsE
http://www.javatpoint.com/creating-servlet-in-eclipse-ide

Pre-requisites:

Should have basic knowledge on OOPs concepts and Java Programming in general.

Agenda

- Introduction
  - Introduction of Servlet and web world in general.
- Servlet Architecture
- Servlet lifecycle
  - Different life cycle of a method is.
- Request and Response
- Being a Web Container
  - Discussion about Servlet Engine
- Session management
  - How it takes place in Servlet world.
- Overview of JSP
- JSP Elements
- Demo
  - At the end of the session, I will show how to code, develop and execute servlet.

By end of this session you should know:

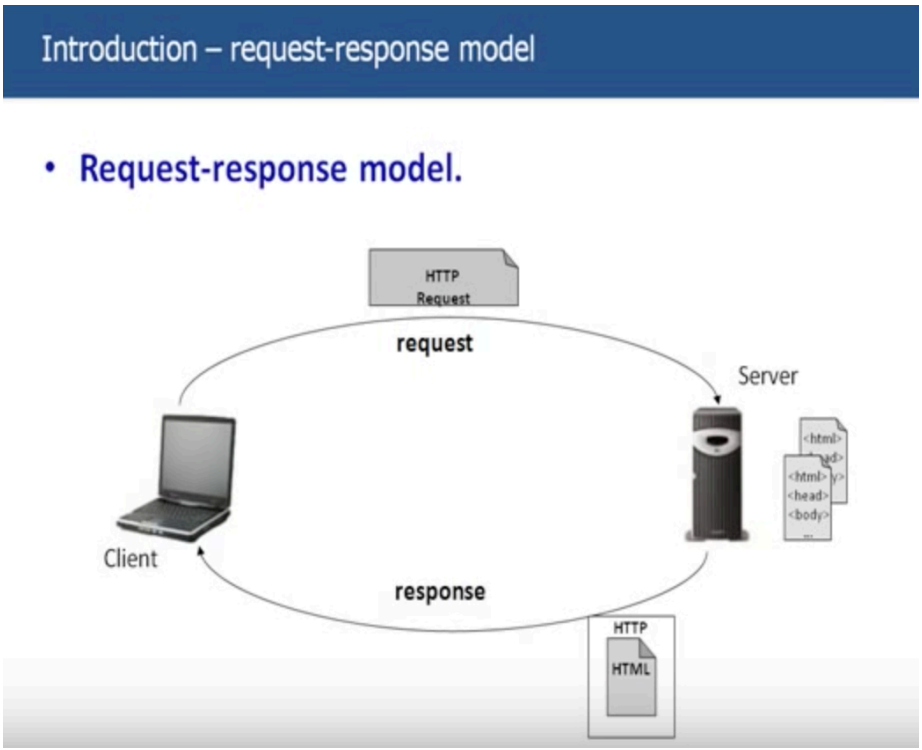
- what a servlet is?
- where servlets are used?

- How to use a servlet?
- What are the advantages of using a servlet?
- You should also able to develop and execute your own servlet apps.

Introduction

Request-response model.

Let us discuss how communication takes place in web world.



There is a client and there is a server. Let's discuss about these 2.
A client is basically, something or someone who is requesting some resource.

And a server is basically a machine, which serves the request which is requested by the client.

Ex: Imagine a customer sitting at a restaurant. He is waiting for the server to come by and take his order. The same rule applies in a client-server network; the client can be laptop, desktop, a smartphone, or pretty much any computerized device, can make a request from the server.
Client uses the network as a way to connect with and speak to the server. Just as the computer speaks to his server, the client uses the network to send and receive communications about its order, or request.

Comment [Office1]: http://study.com/academy/lesson/what-is-a-client-server-network-definition-advantages-disadvantages.html

Ex: Logon to computer and check on to email. So you go to the website, yahoomail.com. So here you are nothing, but a client to yahoo server. Yahoo server is a web server or http server.

Web server is a machine which takes request from the client and can give back a response. And that response is a static html page, any page or any file which is static in nature, its not dynamic.

When your logon to yahoomail.com, you send request to yahoo server. And you are sending the request across the web, over the web and this request is nothing but HTTP Request.
HTTP is a protocol stands for Hyper Text Transfer Protocol. And everything you communicate through web should be HTTP. This webserver understands only understand HTTP nothing else.

When you send an HTTP Request to server, it checks for the page you requested in the repository. For its repository, it picks up one static page from bunch of pages it has and sends that page across to client over the web again. This is again a HTTP Response.

Client here is web browser.

Web Server is a server or a machine which can take in a HTTP Request over the web and can serve the client with the response. And this response is nothing but a static page.

Let's look into more details about HTTP Request and HTTP Response:

Key elements of request are

- HTTP method:

We will discuss about this HTTP methods and difference between the various methods we have in later part of the session. But for now, you just keep in mind that HTTP method is one of the parameters sent from the client to server when making a request for a resource. And this method basically would be GET and POST most of the methods.

- The page to access(a URL): When you access yahoomail.com www.yahoomail.com is a URL and you are sending URL to the server.

- Form Parameters:

Ex: You typed yahoomail.com, request went to the server and then server sends back the response to you. And that response is nothing but the login page for the yahoo mail.
In second page, you see username and password and these are text boxes. After entering the details, click on go. Form Parameters are nothing but username and password which you have typed.

HTTP request contains Username and Password in the Form Parameters.

Key elements of response are

- A status code: This status code will tell the client whether the request was successful or not.
  200 – Successful
  404 – Page Not found
- Content-type: Say for example status code is 200 and server found a valid HTML page for the request. Now server should have some way to tell the client what type of content you are sending to client. The content be anything like text, picture, html etc.

  Why is so important sending content type?

For ex: Content-type was HTML, so the server sends back the content-type as HTML. The client should know what the content type is, what type of data is coming from server. So that, it can accordingly present that information to the end client.

Suppose, server has sent content-type as HTML, so what web browser do, the web browser should make a note is very intelligent, to understand HTML, able to parse HTML file, able to render HTML file to the end client in a very beautiful fashion. As you know, HTML file is a simple text which has html tags. Body tag, Head tag etc., When end user sees that html page, its no more has HTML tags, bit it's a very beautiful web page.

When the server actually sends the client the content-type the web browser, client takes in that, and depending on the content-type it actually renders the information to the end user.

In case of jar file, web browser shows dialogue box for the end user, to save the jar file on desktop.

- The content: The actual content the server is sending back to client.

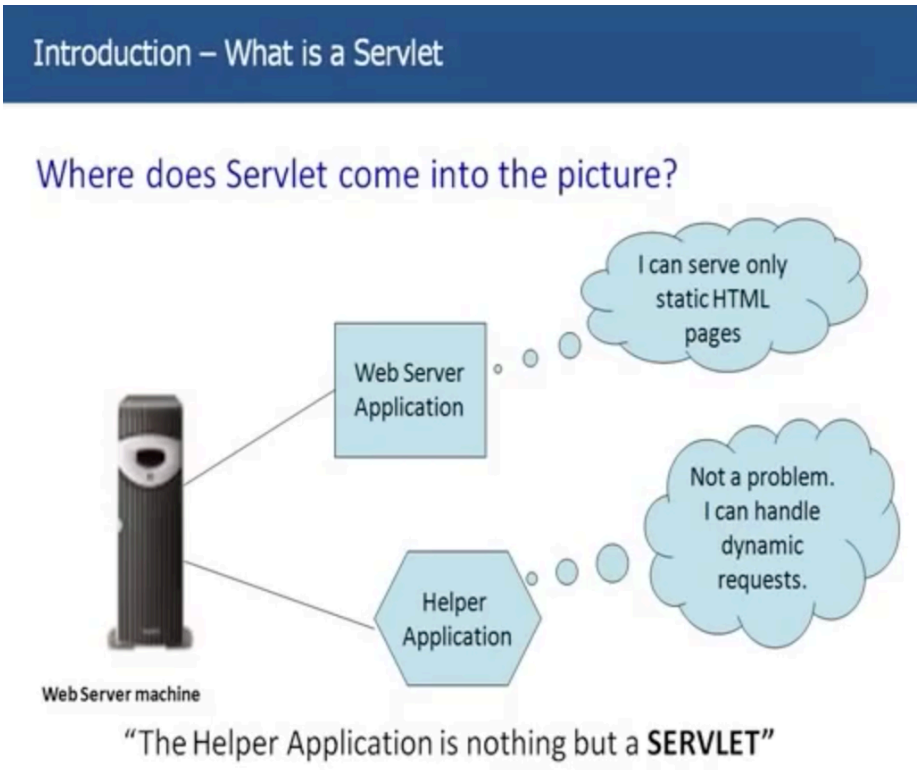
Till now, what we have discussed was only about static pages. The client requests for a static page and Server sends back the static page. Now assume that, server or client needs something which is not static in nature, which is dynamic in nature.

Coming back to the yahoo mail, the first page you get is the static page. But when you type in user name and password click on go, to get my mail, that is the page you get is not a static page, it's a dynamic page.

But web server handles only static pages, and it can serve only static pages to the client. It doesn't have ability to develop dynamic pages and sent it across the client.

How do you address this issue of developing a dynamic web page and sending back to client? This is where Servlets comes into picture.

When a request comes from client for dynamic page, web server cannot serve that request, it can only serve static HTML pages.



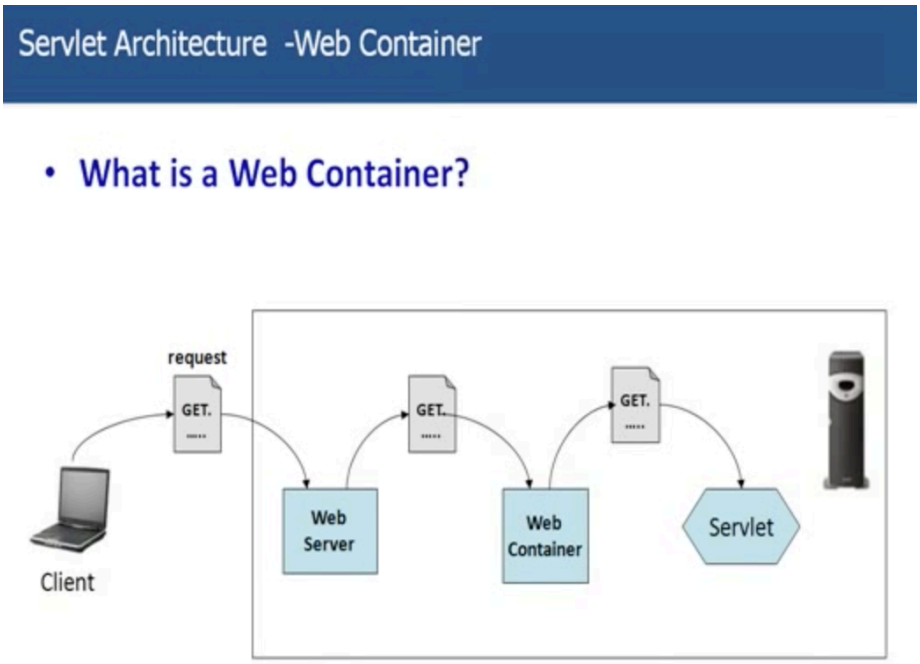
So, this Helper Application comes into play, it says not a problem I can handle dynamic requests.

This Helper Application is nothing but a **SERVLET**.

Servlets came into picture because client needs a dynamic content and not static content always. So, whenever you need a dynamic content, we use servlets.

Let's see how the flow goes from Client to Server

Client sends the request to Web Server and is type of GET. Web Server will now take the help of Servlet to develop a dynamic page.



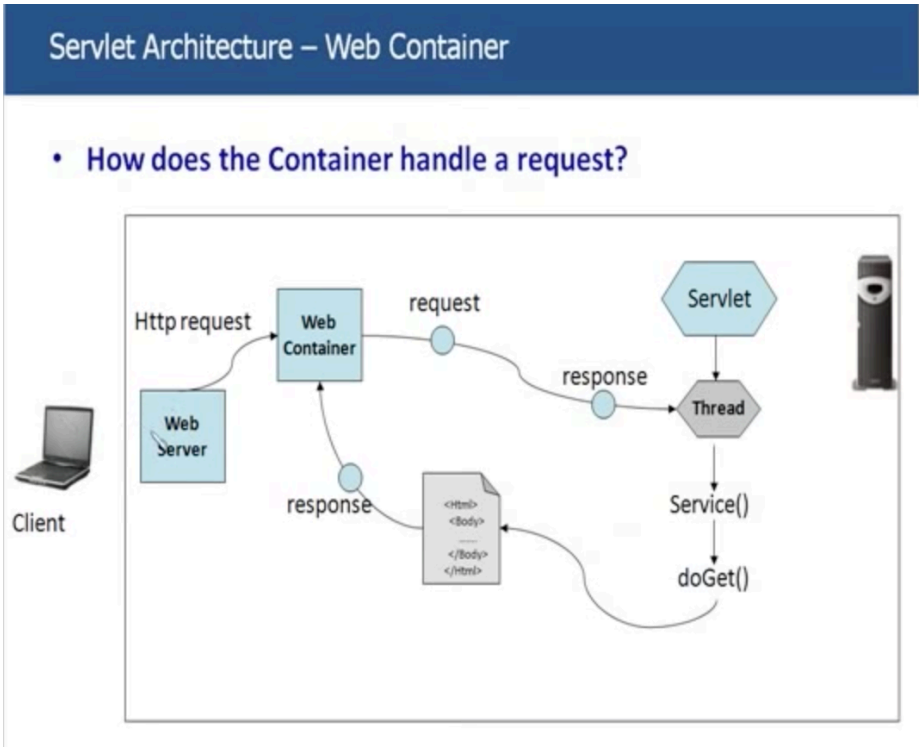
Servlet is nothing but a Java Program, some kind of Special Java Program which resides on the server. But this Java Program doesn't have main method. It has got some callback methods.

How does the web server communicate to Servlet?

Web Container/ Servlet Engine. This is one actually helps web server communicate with the Servlet. Servlet lives and dies within a web container. So, web container is responsible for invoking methods on a Servlet. So Servlets basically has some

callback methods, which the web container knows that is responsible to call those methods on the Servlet.

How does the Container handle a request?



Client sends the request to web server and web server sends the request to web container. Now, it's the web containers responsibility to communicate with the servlet and allow the servlet to build the dynamic content and send the response back to web container again which will return to the web server, which is ultimately go to the client.
But one thing, I want you to notice here is request, we have here which is sent to Servlet and HTTP Request which is sent to Web Container from web server is different. The request sent by the client is HTTP Request, because, this server can understand only HTTP. Basically, client sends the HTTP Request and Web server

sends same request to Web container. But, as we have discussed, a Servlet is nothing but a Java Program so how can a Java program doesn't understand what's that HTTP Request is? It can only understand objects.

Web Container actually converts this HTTP Request into Valid Request Object. And it sends across and it also creates a response objects to the Servlet.

Web container will create a thread for each request which comes into servlet and it calls the callback methods. As, I told you, Servlet doesn't have main method, but it has callback methods, and web container knows, what callback methods has to call on the Servlet.
Call back method is doGet(). Say for example, if the request is type GET, the callback method which will be called on the Servlet is doGet(). Within this doGet() you basically write whole code which will for dynamic content. All your business logic goes either in doGet() or doPost() which will be called based on type of method from the client was POST.

The doGet() method is called, Servlet all the processing in doGet(), builds a response and sends response back to Web Container. Now this response is again a Java Object. This again, converts Response object to HTTP Response and sends back to the Web Server.

This is how Web Container is responsible to manage the whole life cycle of the Servlet. Container is responsible to invoke the Servlet methods and communicate between the Server and Servlet and send response back to client ultimately.

What is the role of Web Container?

Container is nothing but a Servlet engine in which Servlets and JSPs as well live and die. It is responsible for everything that happens to a Servlet.

- Communication Support
- Lifecycle Management:
    - As I told you, Servlet has got some callback methods, its got some methods to initialize, it got methods to destroy the servlet. All these methods were invoked by container. And it knows when to invoke which

method. Basically, web container is responsible for creating a Servlet, invoke methods on the Servlet, execute the Servlet and ultimately even destroy the Servlet.

- Multi-threading support:
    - As I told you, for every request comes in for that particular servlet, web container sponsor a thread i.e., separate thread for every request comes in for the Servlet.
- Security:
    - Since, client directly cannot talk to Servlet, it has to go through Web Container. So you can put all kind of security code in that container. So that, only valid request will go to the Servlet. Any other invalid request will not have taken over to the Servlet.
- JSP Support

We discussed that Servlet comes into picture when you need dynamic content. We also discussed, client doesn't directly talk to the Servlet, it has to go via the web container. But now,

how does the web container know which Servlet can serve the request requested by the client?

There might be 5 or 6 Servlets in my application which do different kind of jobs. Now, how the container knows, when a request comes into web server which servlet can help that request.
Now, take the example you clicked on yahoo.com entered the username and password and send the request to the server. Now, how does the Web container knows which particular servlet in the web application can serve the request for that yahoo email.
Let's assume, it has a Servlet, YahooServlet. Now, the container has to know, this YahooServlet can handle the request which has sent by client for seeing his yahoo mails.

Now let's see how that happens. Basically, there is a file called web.xml and this is also called as deployment descriptor. This is the main file/master file for web container. Everything that web container needs is available in this file. Basically, you have information about Servlet in this file.

Every Servlet in the web application should have an entry into this web.xml. LoginServlet, is the servlet which handles all the request for the yahoo login. I will have an entry in web.xml file. So, there is a Servlet element within which I have a Servlet name (Some readable name) and Servlet class(Path). There is also section called Servlet mapping. We have already mapped Servlet class to Servlet name in Servlet element.

But how will the Servlet will know, request is for yahoomail.com/logon, how will it know, that request is for this Servlet class. That is derived from Servlet mapping element. Again Servlet mapping element has 2 tags i.e., Servlet name and URL Pattern. Servlet name matches with Servlet name in Servlet Tag and URL pattern is nothing but the client has sent in as a request.

Servlet can have 3 names:

- Client known URL name: url-pattern
- Deployer known secret name: servlet-name
- Actual file name: servlet-class

Servlet Architecture – Deployment Descriptor

- **How does the Container know which Servlet the client has requested for?**

A Servlet can have 3 names

- Client known URL name
- Deployer known secret internal name
- Actual file name

```
<web-app>
 ……..
 <servlet>
  <servlet-name>LoginServ</servlet-name>
   <servlet-class>com.Login</servlet-class>
 </servlet>

 <servlet-mapping>
  <servlet-name>LoginServ</servlet-name>
  <url-pattern>/Logon</url-pattern>
 </servlet-mapping>
 ……..
 ……..
</web-app>
```

web.xml