Web Technologies Lab

Sibi Chakkaravarthy Sethuraman
Computer Science and Engineering
VIT-AP, Amaravati, India

Points to Ponder

- Don’t use laptop
- Keep your bag outside

Launch your own (first) github.io page

Launch your first static page in any (web) server

- Signup → https://in.000webhost.com/ or hostinger or infinityfree
- Signup → http://www.dot.tk/en/index.html?lang=en or Dottk or freenom

Registering Domain and Configuring Nameserver

Basic HTML Tags and Exercises

1. Create a webpage that prints your name to the screen.
2. Create a webpage and set its title to “Welcome to VIT-AP".
3. Create a webpage that prints your name in Sky blue to the screen.
4. Print the numbers 1 - 10, each number being a different color.
5. Create a web page to print your name in a Calibri font.

Solution in https://github.com/sibichakkaravarthy/sibichakkaravarthy.github.io

Basic HTML Tags and Exercises – Cont'd..

6. Print a paragraph with 4 - 5 sentences. Each sentence should be a different font.

7. Print a paragraph that is a description of a book, include the title of the book as well as its author. Names and titles should be underlined, adjectives should be italicized and bolded.

8. Print your name to the screen with every letter being a different heading size.

Solution in https://github.com/sibichakkaravarthy/sibichakkaravarthy.github.io

Basic HTML Tags and Exercises – Cont'd..

9. Print the squares of the numbers 1 - 20. Each number should be on a separate line, next to it the number 2 superscripted, an equal sign and the result. Example: 10^4 = 10000

10. Print two paragraphs that are both indented (Use)

(nbsp - **Non Breaking Space Publishing**)

Solution in https://github.com/sibichakkaravarthy/sibichakkaravarthy.github.io

Basic HTML Tags and Exercises – Cont'd..

11. Print two lists with any information you want. One list should be an ordered list, the other list should be an unordered list.
    - Change the ordered list and represent the ordering in number, alphabets and Roman numerals

12. Prints an h1 level heading followed by a horizontal line whose width is 100%. Below the horizontal line print a paragraph relating to the text in the heading.

13. Use delete and insert tag to print the following
    - HTML is a programming language → Strike this
    - HTML is not a programming language → Insert this

Solution in https://github.com/sibichakkaravarthy/sibichakkaravarthy.github.io

Basic HTML Tags and Exercises – Cont'd..

14. Print a long quote and a short quote. Use tags <blockquote></blockquote> and <q></q>.

15. Definition and Usage Tags (DL/DT/DD)

- Usage <dl><dt><dd></dd></dt></dl>

16. There is **Address tag also**

- Usage <address> </address>

Solution in https://github.com/sibichakkaravarthy/sibichakkaravarthy.github.io

Basic HTML Tags and Exercises – Cont'd..

17. Create a link to Google search engine
18. Create a page with a link at the top of it that when clicked will jump all the way to the bottom of the page.
19. Create a page with a link at the bottom of it that when clicked will jump all the way to the top of the page.
20. Create a page with a link at the top of it that when clicked will jump all the way to the bottom of the page. At the bottom of the page there should be a link to jump back to the top of the page.

Solution in https://github.com/sibichakkaravarthy/sibichakkaravarthy.github.io

Basic HTML Tags and Exercises – Cont'd..

21. Display different images. Skip two lines between each image. Each image should have a title.
22. Display an image that has a border of size 2, a width of 200, and a height of 200.
23. Display an image that when clicked will link to a VITAP home page
24. Display an image that when clicked will link to itself and will display the image in the browser by itself
25. TO comment HTML tags use

Syntax: <!– your code goes here -->

Solution in https://github.com/sibichakkaravarthy/CSE4004-*

Basic HTML Tags and Exercises – Cont'd..

26. Defining HTML table
27. Align attribute (right, left, top, bottom)

28. Classwork – Design your own resume

Solution in https://github.com/sibichakkaravarthy/sibichakkaravarthy.github.io

HTML and CSS

- Create a web page with basic tags such as paragraph, heading, anchor, hyperlink
- Add inline style to all the tags
- Add an external and embedded styles

Stylesheets - font

- Create a web page with basic tags such as paragraph, heading
- Create an external stylesheets and add the following styles
  - Change font
  - Change font style
  - Set font size with pixels, pica, point, em and percentage
  - Add strokes to the font
  - Change the letters inside <p> to all caps
  - Align all the paragraph to center
  - Change the font color

Note: 1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px

Stylesheets - text

- Adding shadow to the text using **text-shadow** property
- Transforming text to all caps, small using **text-transform** property

Playing with HTML and CSS – Exercise 1

- **Body**
  - Pick a nice background color, font family/color/size and line-height for the <body>
  - Setting font properties at the body-level will apply it on all basic texts (<p> ..etc.,)
- **Headers**
  - Choose a nice color and font-family for headers (<h1…h6>
  - Choose font size and line-height for headers
  - Resize image with width
- **Links**
  - Change links colors & text decoration
  - Add some hover effects on links using the hover property
  - Add some hover effects using background-image property

```
a:hover
{
    color: black !important;
}
```

```
background-image: url('3.jpg')
```

Playing with HTML and CSS – Exercise 2

- Create a web page with basic tags such as paragraph, heading, anchor, hyperlink
- Add inline styles to each tag
- Add an embedded styles on the same web page and try to override the styles of paragraph

Embedding JS into HTML page

- Use **<script>** tag to specify your JavaScript
- <head
- <script>

- // your JS code

- </script>
- </head>

Types of JavaScript

- Inline JavaScript
  - Drawback: **JS code will not be cached.**
  - **(The browser** will not be able to store it in **cache**, thus requiring it to fetch the **full HTML file each time).**
- Embedded or Internal JavaScript
- External JavaScript

Hello world in JS

```
<!DOCTYPE html>
<html>
<head>

<title> Demo JavaScript</title>
<!--<script type="text/javascript" src="demo1.js"></script>-->


<script>
    document.write(" Hellow world");
</script>



</head>
<body>

</body>
</html>
```

JavaScript Variable: Declare, Assign a Value

- use the keyword **var** to declare a variable

- Example:

Var name = “sibi”;

or

Var name;

Name = “sibi”;

JS – Example code

```
    <title> This is JS demo page</title>
    <script type="text/javascript">
        var age = prompt("Please enter your age");
        var a = 22;
        var b= 3;
        var add = a+ b;
        var minus = a - b;
        var multiply = a * b;
        var divide = a/b;
        document.write("User age is: =" + age + "<br/>");
        document.write("First No: = " + a + "<br />Second No: = " + b + " <br />" + "A simple demo of JS </br> ");
        document.write(a + " + " + b + " = " + add + "<br/>");
        document.write(a + " - " + b + " = " + minus + "<br/>");
        document.write(a + " * " + b + " = " + multiply + "<br/>");
        document.write(a + " / " + b + " = " + divide + "<br/>");
    </script>
</head>
```

Embedding html tags and CSS in JavaScript

```
<body>

    <p id="sibi"> This is demo for embedding css inside JS </p>

    <script>
        var age = prompt("Please enter your age");
        var myElement = document.getElementById('sibi');
        myElement.style.color="green";
        document.write("<h1> User age is: = </h1>" + age + "<br/>");
    </script>

</body>
```

HTML Events and JS Event Handling

- HTML events are **"things"** that happen to HTML elements.

or

- HTML events are the **characteristics** of the HTML elements
  - **Example: An HTML button was clicked**

- When **JavaScript** is used in HTML pages, JavaScript can **"react"** on these events.

Event1 – Text event handling

```
<!DOCTYPE html>
<html>
<body>
<h1 onclick="this.innerHTML=‘Text Changed!’">Click on this text!</h1>
</body>
</html>
```

Event 2 – Button event

```
<!DOCTYPE html>
<html>
<body>
<button onclick="this.innerHTML=Date()">The time is?</button>
</body>
</html>
```

Event 3 – Button + HTML element event

```
<!DOCTYPE html>
<html>
<body>
<button onclick="document.getElementById('demo').innerHTML=Date()">The Date and time is?</button>
<p id="demo">Now in this place Date and Time will be displayed</p>
</body>
</html>
```

HTML elements

- Finding HTML elements by id
  - Example: var myElement = document.getElementById(“sibi");
- Finding HTML elements by tag name
  - Example: var x = document.getElementsByTagName("p");
- Finding HTML elements by class name
  - Example: var x = document.getElementsByClassName("intro");

Event 4 - Function

```
<script>
function myFunction() {
  var x = document.getElementById("firstname");
  x.value = x.value.toUpperCase();
}
</script>
```

```
<body>
Enter your name: <input type="text" id="firstname" onchange="myFunction()">
<p>When you leave the input field, a function is triggered which transforms the input text to upper case.</p>
</body>
```

Event 5 – Mouse Event

- Onmouseover() and onmouseout()
- Onmousedown() and onmouseup()

or

- Onclick()

onmouseover() and onmouseout()

```
<div id = "sibi" onmouseover="mOver()" onmouseout="mOut(this)" 
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>
<script>

function mOver()
{
document.getElementById("sibi").innerHTML = "<span style='color: green;'>**Message**</span></br>";
}

function mOut(obj)
{
  obj.innerHTML = "Mouse Over Me"
}
</script>
```

Onmousedown() and onmouseup()

```
<div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background-color:#D94A38;width:90px;height:20px;padding:40px;">
Click Me</div>
<script>
function mDown(obj)
{
  obj.style.backgroundColor = "#1ec5e5";
  obj.innerHTML = "Release Me";
}
function mUp(obj)
{
  obj.style.backgroundColor="#D94A38";
  obj.innerHTML="Click -on- me";
}
</script>
```

Event 6 – A simple graphics using JS

```
<script>
function myMove() {
  var anim = document.getElementById("animate");
  var pos = 200;
  var id = setInterval(frame, 60);
  function frame() {
    if (pos == 350) {
      clearInterval(id);
    } else {
      pos++;
      anim.style.top = pos + "px";
      anim.style.right = pos + "px";
    }
  }
}
</script>
```

What is this pos denotes?

You can control framerate also

Why navigation till 350px

Clears the frame on the current rendered frame

A simple graphics using JS

```
<p><button onclick="myMove()">Animate me</button></p>
<div id ="container">
  <div id ="animate"></div>
</div>
<script>
function myMove() {
  var anim = document.getElementById("animate");
  var pos = 200;
  var id = setInterval(frame, 60);
  function frame() {
    if (pos == 350) {
      clearInterval(id);
    } else {
      pos++;
      anim.style.top = pos + "px";
      anim.style.right = pos + "px";
    }
  }
}
</script>
```

JavaScript – Classwork Lab Practice – 30.07.2019

- Create a web page with basic tags such as paragraph, heading, anchor, hyperlink
- Add inline styles to each tag
- Create a div container at the top left and bottom right corner to display date and time.
- Create a button which changes the inner html text when clicked.
- Create a div container and load the image.
  - Perform mouse over operation to change the image.
  - Perform alert message “ Mouse near me” when the mouse is hover the image.
  - Perform alert message “ Mouse near me” when the mouse is clicked on the image.
  - When mouse is clicked change the position of the div container to the top right.

Solution -> https://github.com/sibichakkaravarthy/CSE4004-*

Create your own web server to handle and serve client request

1. Create a web socket
2. Bind the socket connection using IP and Port
3. Listen for the connection
4. Accept the connection using connection IP and Port
5. Receive the request
6. Process the request and acknowledge

Create your own web server to handle and serve client request

- Create a socket

(WebSockets provide a persistent connection between a client and server that both parties can **use** to start sending data at any time. The client establishes a **WebSocket** connection through a process known as the **WebSocket** handshake. This process starts with the client sending a regular HTTP request to the server).

```
import socket

HOST = '127.0.0.1'
PORT = 8000
listen_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Binding and Listening for the connection

- Binding the socket connection using IP and Port and listening for the connection

```
listen_socket.bind((HOST, PORT))
listen_socket.listen(2)
```

Accepting the request from the client

- Accept the connection using connection IP and Port

```
connection, address = listen_socket.accept()
request = connection.recv(1024)
```

Serving the clients with static web page

```
connection.sendall("""HTTP/1.1 200 OK
Content-type: text/html


<html>
    <body>
        <h1>Hello, World!</h1>
    </body>
</html>""")
```

Closing the socket

- Terminating the communication between the client and server

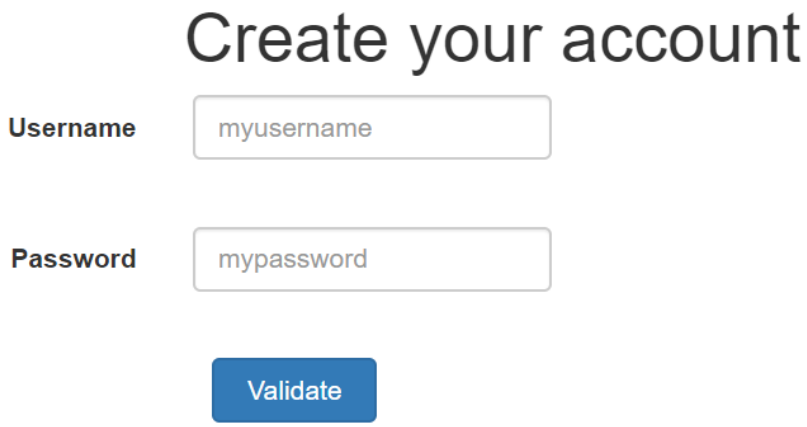
```
connection.close()
```

Create your own web server to listen continuously and handle all the client request

JS,HTML,CSS – Classwork Lab Practice – 06.08.2019

- Design a form as given

(With proper margin)



- On clicked the input text box should be highlighted
- On button click validate for bad username and password and highlight the same.
- Create another button on the middle of view port (use box model for button positioning)
- On button click retrieve the location

Data representation – Classwork lab practice – 27.08.2019

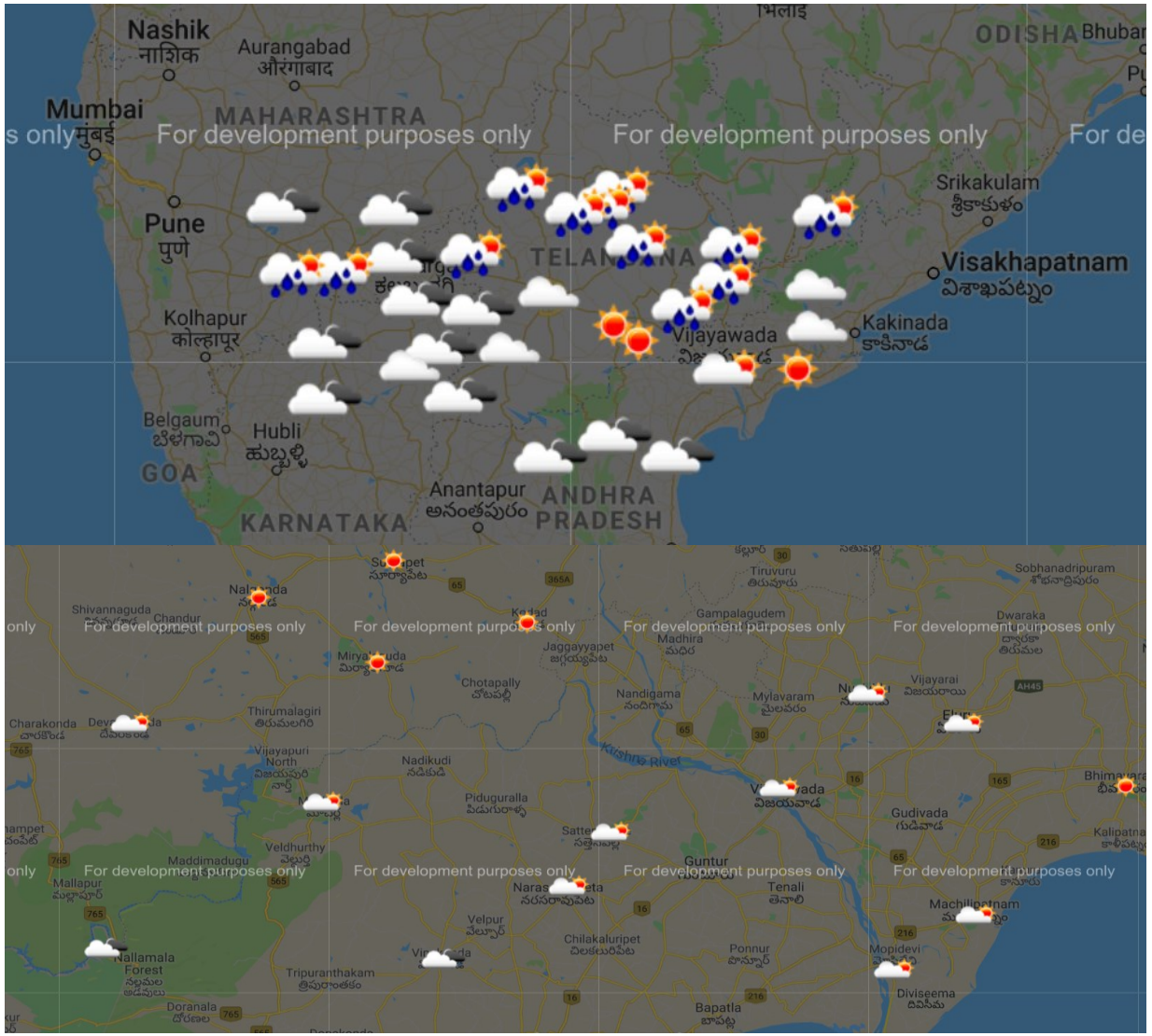
- Design a web application to retrieve the weather data from https://openweathermap.org
- Task 1:
  - Draw a vertical line to split the web page into two verticals.
  - Retrieve the city name and coordinates (use forecast id)
  - Create a table and display Vijayawada & Mumbai weather (use current weather data and forecast id)

- Note: Use json formatters to format json data for better visualization.

Data representation – Classwork lab practice – 03.09.2019

- Design a web application to retrieve the weather data from https://openweathermap.org
- Task 2:
  - Integrate google map api.
  - Retrieve the city name and coordinates (use forecast id)
  - Retrieve the icon img from openweathermap and plot the weather conditions (use icons from openweathermap)

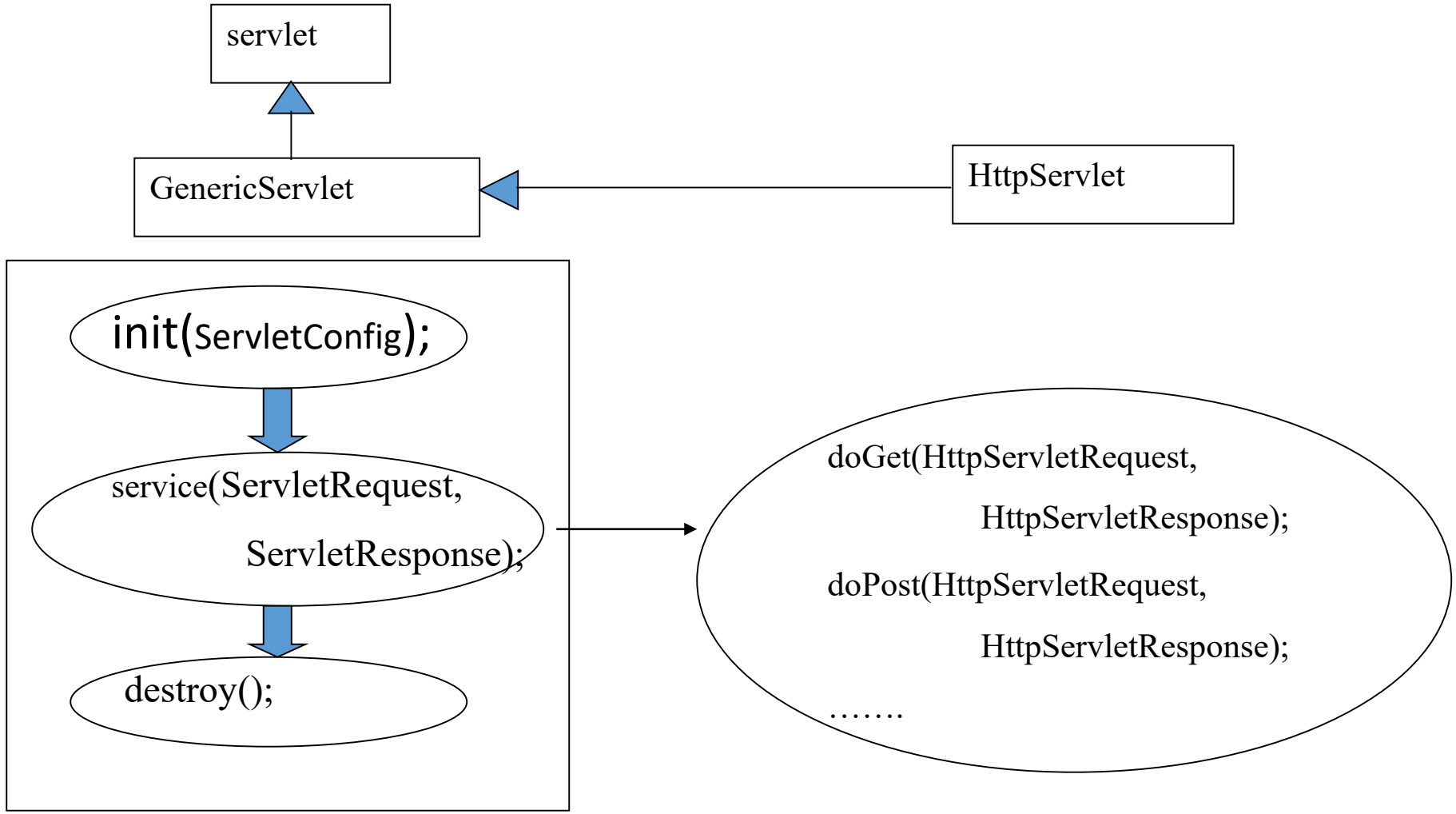
Sample output



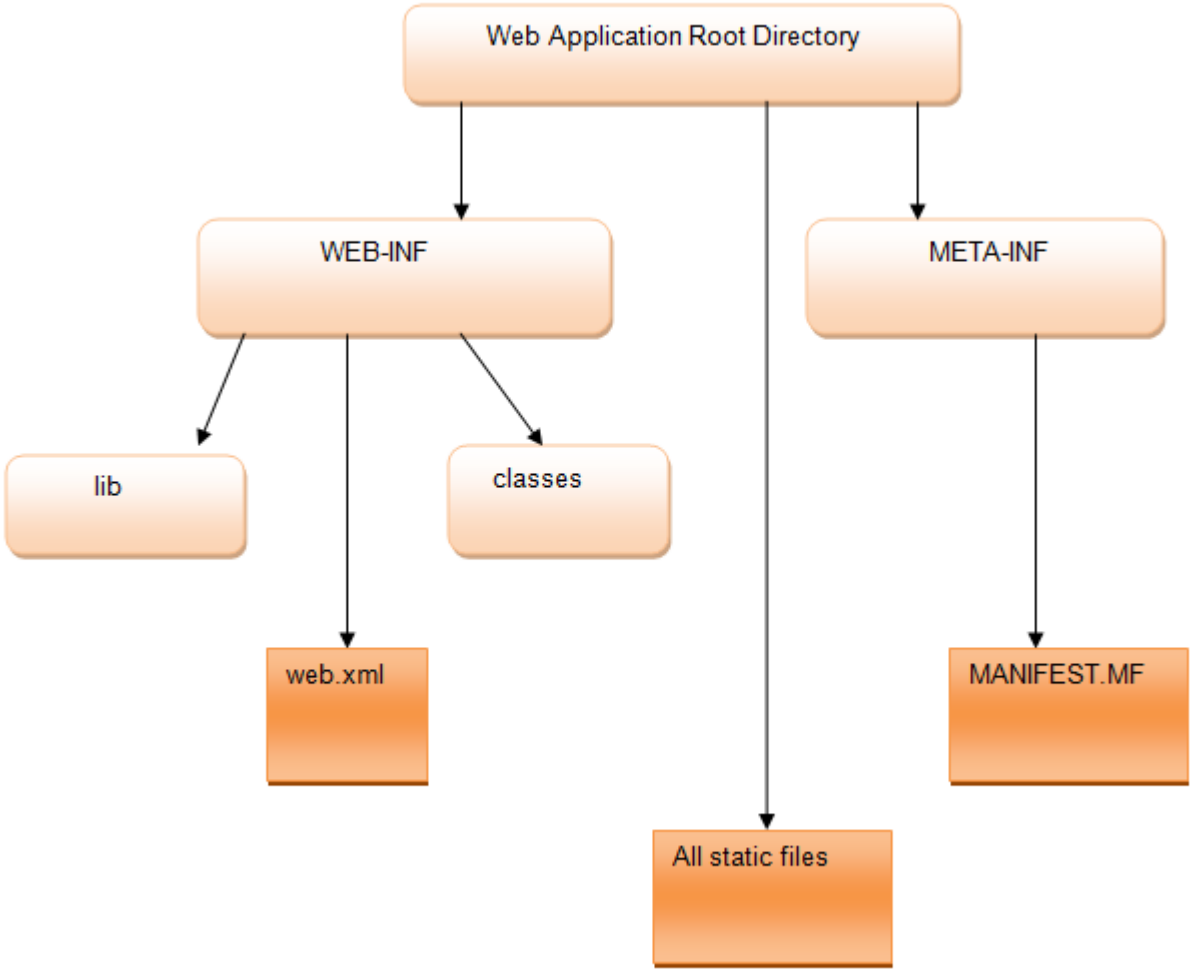
Working with Servlets

- Create a simple servlet to print basic <h> tags.
- Create a simple button and invoke a servlet on button click.

Life Cycle of Servlet



Servlet container folder structure



Servlets – How multiple request, response gets handled

- Inside the web (servlet) container, we have a special file called deployment descriptor (web.xml)

- Inside web.xml, we define, for which request which servlet should be called.

Web.xml

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app>
    <servlet>
        <servlet-name>MyFirstDynamicWebPage</servlet-name>
        <servlet-class>MyFirstDynamicWebPage</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>MyFirstDynamicWebPage</servlet-name>
        <url-pattern>/cs</url-pattern>
    </servlet-mapping>
</web-app>
```

HttpServlet

- Base class for web-based servlets
- Overrides method **`void service ()`**
  - Request methods:
    - **`GET`**
    - **`POST`**
- Methods **`doGet`** and **`doPost`** respond to **`GET`** and **`POST`**
  - Called by **`service`**
  - Receive **`HttpServletRequest`** and **`HttpServletResponse`** objects

GET and **POST method** is used to transfer data from client to server in HTTP protocol but Main difference between **POST** and **GET method** is that **GET** carries request parameter appended in URL string while **POST** carries request parameter in message body which makes it more secure way of transferring data from client to server

HttpServletRequest

- **`HttpServletRequest`** interface
  - Object passed to **`doGet`** and **`doPost`**
  - Extends **`ServletRequest`**
- Methods
  - **`String getParameter( String name )`**
    - Returns value of parameter **`name`** (part of **`GET`** or **`POST`**)
  - **`Enumeration getParameterNames()`**
    - Returns names of parameters (**`POST`**)
  - **`String[] getParameterValues( String name )`**
    - Returns array of strings containing values of a parameter
  - **`Cookie[] getCookies()`**
    - Returns array of **`Cookie`** objects, can be used to identify client

HttpServletResponse

- **`HttpServletResponse`**
  - Object passed to **`doGet`** and **`doPost`**
  - Extends **`ServletResponse`**
- Methods
  - **`void addCookie( Cookie cookie )`**
    - Add **`Cookie`** to header of response to client
  - **`ServletOutputStream getOutputStream()`**
    - Gets byte-based output stream, send binary data to client
  - **`PrintWriter getWriter()`**
    - Gets character-based output stream, send text to client
  - **`void setContentType( String type )`**
    - Specify MIME type of the response (Multipurpose Internet Mail Extensions)
    - MIME type “text/html” indicates that response is HTML document.
    - Helps display data