BridgeLabz

Employability Delivered

Java Web Development

Agenda

BridgeLabz
Employability Delivered

- Firstly we will start with installation of nginx Web Server to serve Static Web Pages
- Creating a simple web server in java
- Install Tomcat Web Server and run Examples
- Use Servlet to create Dynamic Web Pages

What is nginx

- Nginx pronounced "engine X”.
- It is a Popular light weight Web Servers.
- Predominantly used for serving static web pages
- Also used for reverse proxy and load balancing
- Check if nginx is installed
  nginx –v
- Install with Homebrew
  brew install nginx
- Start and Stop
  nginx, ***nginx –s stop***
  – s => Signals (stop, quit, reload)
  http://localhost:8080/

Configure nginx to serve static files

BridgeLabz
Employability Delivered

- Configuration File – ***nano –T 3 /usr/local/etc/nginx/nginx.conf***
- Serving Static Contents. Open Configuration File and change the root to a new directory to serve html files. Create a html file and copy to this folder.

```
server {
    listen       8080;
    server_name  localhost;

    #charset koi8-r;
    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
    }
}
```

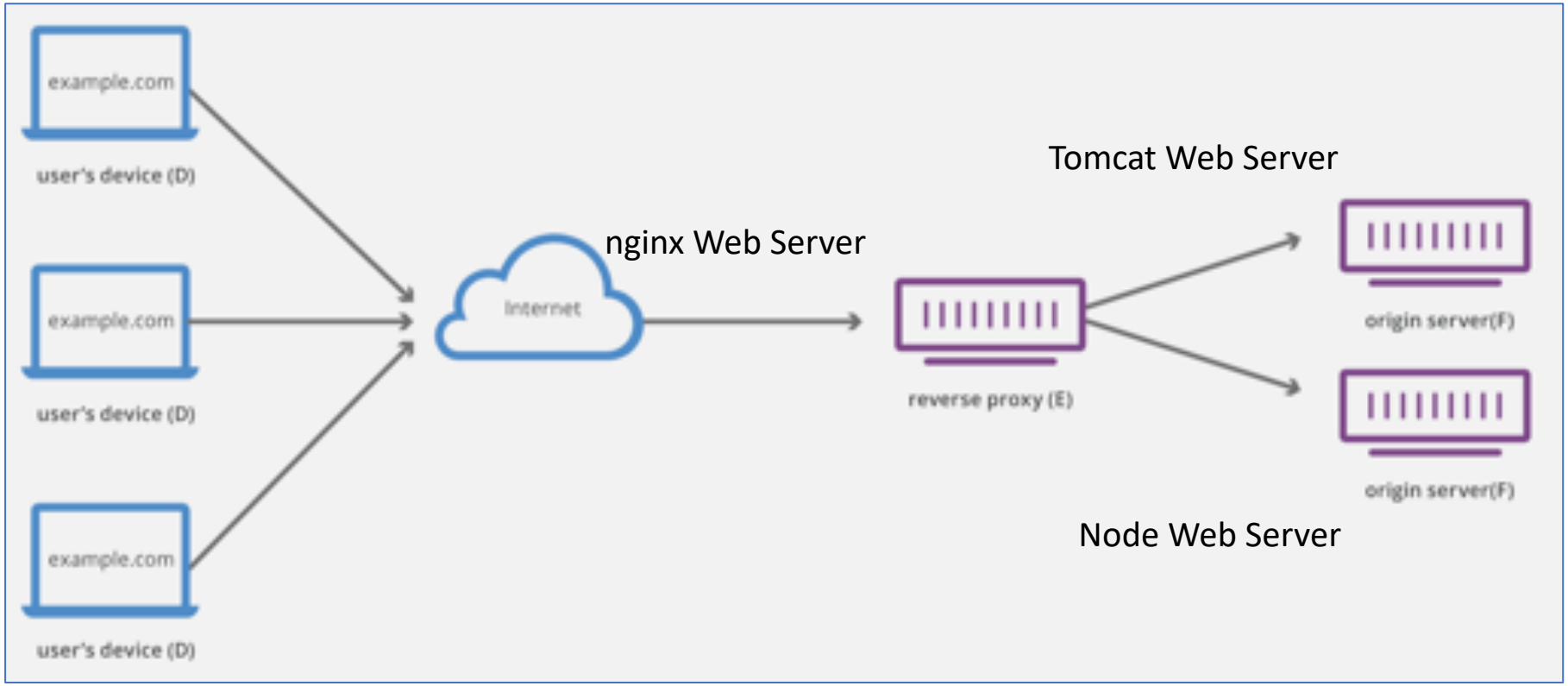
```
server {
    listen       8080;
    server_name  localhost;

    #charset koi8-r;
    #access_log  logs/host.access.log  main;

    location / {
        root /Users/narayan/Development/html/;
        index  index.html index.htm;
    }
}
```

What is reverse proxy

One of the frequent uses of nginx is setting it up as a proxy server, which means a server that receives requests, passes them to the proxied servers, retrieves responses from them, and sends them to the clients.



Configure nginx to serve as reverse proxy

BridgeLabz
Employability Delivered

- Configuration File – ***nano –T 3 /usr/local/etc/nginx/nginx.conf***
- **proxy_pass** directive with the protocol, name and port of the proxied server specified in the parameter. Here proxied to JSON Server
- Check http://localhost:8080/ => notice loads JSON Server

```
server {
    listen       8080;
    server_name  localhost;

    #charset koi8-r;
    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
    }
}
```

```
server {
    listen       8080;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        proxy_pass http://localhost:4000;
    }
}
```

Creating Simple Web Server in Java

- Create a Java HttpServer with a port number
- Create a HttpHandler for the
  - root path,
  - to get the Header Parameters,
  - to get Query Parameters,
  - to get parameters from the Body
- Write the response to Output Stream

Create a Simple HttpServer –

- Here ensure first to create a RootHandler first then create a remaining Handlers

```
public class SimpleHttpServer {
    public static int DEFAULT_PORT = 9000;
    public static int port;
    private HttpServer httpServer;

    private void start(int port) {
        this.port = port;
        try {
            httpServer = HttpServer.create(new InetSocketAddress(port),  backlog: 0);
            System.out.println("server started at " + port);
            httpServer.createContext( path: "/", new Handlers.RootHandler());
            httpServer.createContext( path: "/echoHeader", new Handlers.EchoHeaderHandler());
            httpServer.createContext( path: "/echoGet", new Handlers.EchoGetHandler());
            httpServer.createContext( path: "/echoPost", new Handlers.EchoPostHandler());
            httpServer.setExecutor(null);
            httpServer.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        SimpleHttpServer httpsServer = new SimpleHttpServer();
        httpsServer.start(DEFAULT_PORT);
    }

}
```

Create a RootHandler as inner class

```
public class Handlers {
    public static class RootHandler implements HttpHandler {
        @Override
        public void handle(HttpExchange exchange) throws IOException {
            String response = "<h1>Server start success if you see this message</h1>" + "<h1>Port: " +
                    SimpleHttpServer.port + "</h1>";
            exchange.sendResponseHeaders( rCode: 200, response.length());
            OutputStream os = exchange.getResponseBody();
            os.write(response.getBytes());
            os.close();
        }
    }
}
```

Create a Header and Get Handlers

BridgeLabz
Employability Delivered

```java
public static class EchoHeaderHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange exchange) throws IOException {
        Headers headers = exchange.getRequestHeaders();
        Set<Map.Entry<String, List<String>>> entries = headers.entrySet();
        String response = "";
        for (Map.Entry<String, List<String>> entry : entries)
            response += entry.toString() + "\n";
        exchange.sendResponseHeaders( rCode: 200, response.length());
        OutputStream os = exchange.getResponseBody();
        os.write(response.getBytes());
        os.close();
    }
}

public static class EchoGetHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange exchange) throws IOException {
        Map<String, Object> parameters = new HashMap<~>();
        URI requestedUri = exchange.getRequestURI();
        String query = requestedUri.getRawQuery();
        parseQuery(query, parameters);
        String response = "";
        for (String key : parameters.keySet())
            response += key + " = " + parameters.get(key) + "\n";
        exchange.sendResponseHeaders( rCode: 200, response.length());
        OutputStream os = exchange.getResponseBody();
        os.write(response.getBytes());
        os.close();
    }
}
```

Create Push Handler

```java
public static class EchoPostHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange exchange) throws IOException {
        Map<String, Object> parameters = new HashMap<~>();
        InputStreamReader isr = new InputStreamReader(exchange.getRequestBody(),  charsetName: "utf-8");
        BufferedReader br = new BufferedReader(isr);
        String query = br.readLine();
        parseQuery(query, parameters);
        String response = "";
        for (String key : parameters.keySet())
            response += key + " = " + parameters.get(key) + "\n";
        exchange.sendResponseHeaders( rCode: 200, response.length());
        OutputStream os = exchange.getResponseBody();
        os.write(response.getBytes());
        os.close();
    }
}

private static void parseQuery(String query, Map<String, Object> parameters) throws UnsupportedEncodingException {
    if (query != null) {
        String pairs[] = query.split( regex: "[&]");
        for (String pair : pairs) {
            String param[] = pair.split( regex: "[=]");
            String key = null;
            String value = null;
            if (param.length > 0) {
                key = URLDecoder.decode(param[0], System.getProperty("file.encoding"));
            }
            if (param.length > 1) {
                value = URLDecoder.decode(param[1], System.getProperty("file.encoding"));
            }
            parameters.put(key, value);
        }
    }
}
```

What is Tomcat

- Apache **Tomcat** is an open-source web server and servlet container developed by the Apache Software Foundation (ASF).
- **Tomcat** implements several **Java** EE specifications including **Java** Servlet, JavaServer Pages (JSP), **Java** EL, and WebSocket, and provides a "pure **Java**" HTTP web server environment for **Java** code to run.
- Check if Tomcat is installed

  catalina version
- If not install then installing using brew

  brew install tomcat
- Start, Stop and Help

  catalina start, ***catalina stop***, ***catalina -h***

Tomcat Important Folders

BridgeLabz
Employability Delivered

- **Step 1:** ***catalina version*** shows

```
Tomcat $ catalina version
Using CATALINA_BASE:   /usr/local/Cellar/tomcat/9.0.30/libexec
Using CATALINA_HOME:   /usr/local/Cellar/tomcat/9.0.30/libexec
```

- **Step 2:** ***ls -l /usr/local/Cellar/tomcat/9.0.30/libexec/*** shows

```
Tomcat $ ls -l /usr/local/Cellar/tomcat/9.0.30/libexec/
total 56
-rw-r-----   1 narayan  admin  18982 Dec  7 22:16 BUILDING.txt
-rw-r-----   1 narayan  admin   5409 Dec  7 22:16 CONTRIBUTING.md
drwxr-x---  19 narayan  admin    608 Jan  4 22:40 bin
drwx------  13 narayan  admin    416 Jan  4 22:41 conf
drwxr-x---  34 narayan  admin   1088 Dec  7 22:13 lib
drwxr-x---  13 narayan  admin    416 Jan  5 07:44 logs
drwxr-x---   3 narayan  admin     96 Dec  7 22:13 temp
drwxr-x---  11 narayan  admin    352 Jan  5 10:20 webapps
drwxr-x---   3 narayan  admin     96 Jan  4 22:41 work
```

Used to set up configuration to run Tomcat Server from IntelliJ

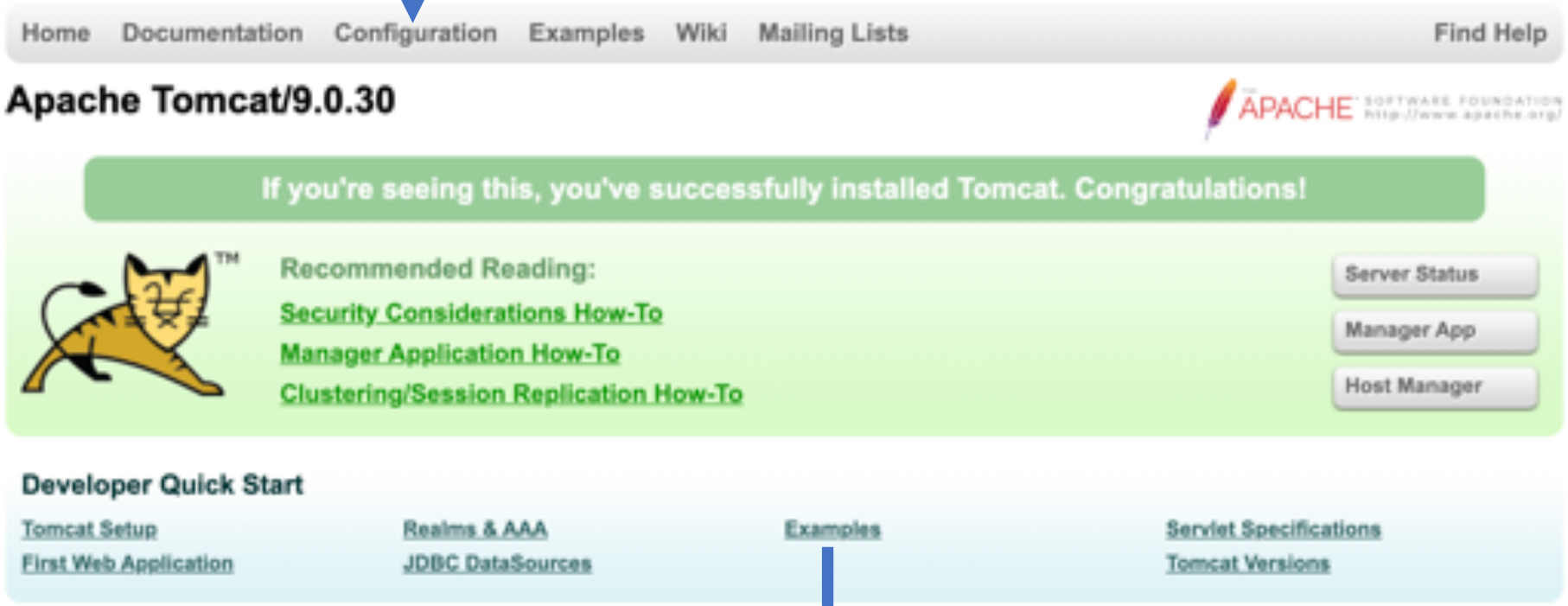
Deployment Folder for all Web Apps

- **Note -** Since nginx and tomcat are starting by default on 8080 port. Ensure one of them is stopped.

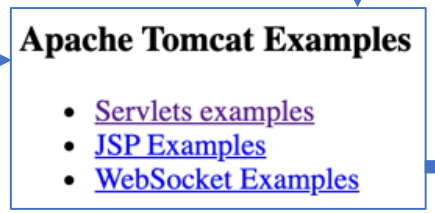
Tomcat Start and Run Examples

BridgeLabz
Employability Delivered

- **Step 1:** ***catalina start***
- **Step 2:** click http://localhost:8080/



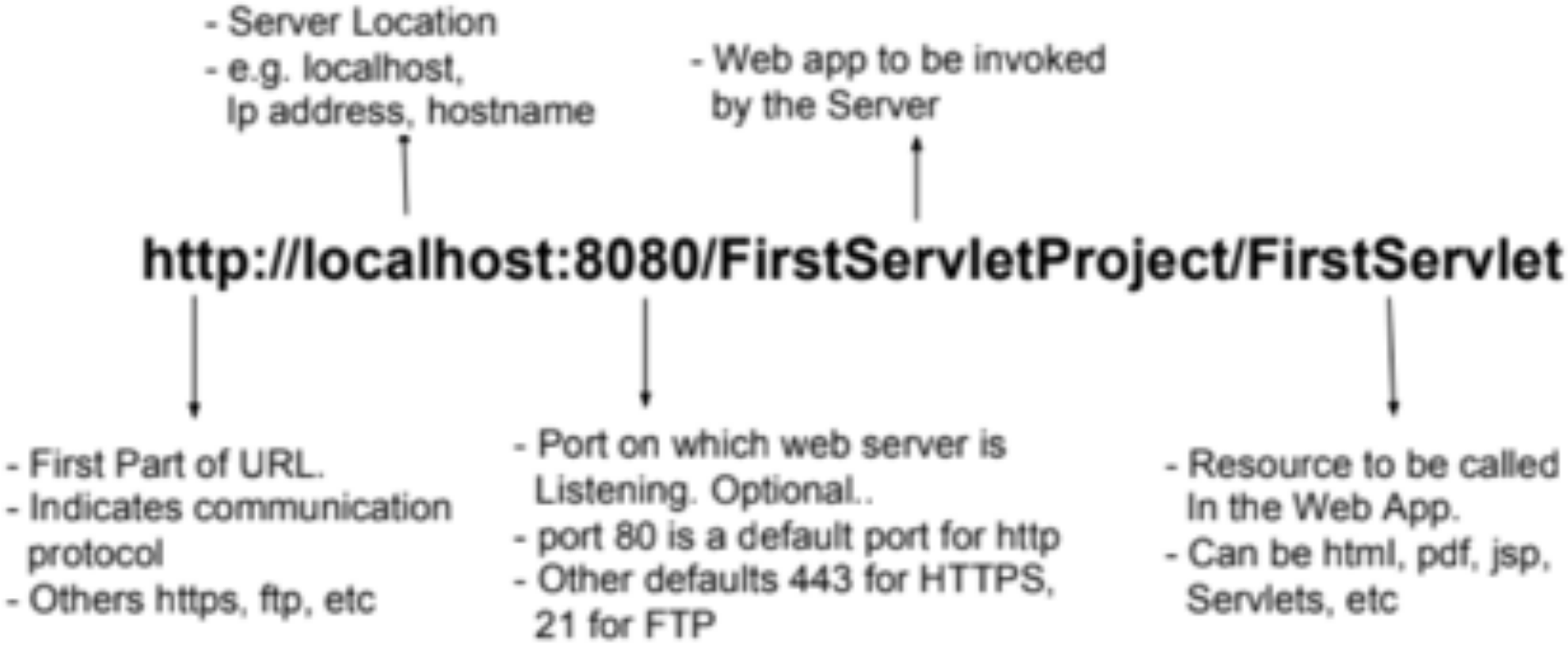
- **Step 3 –** Click on Examples and try each one.



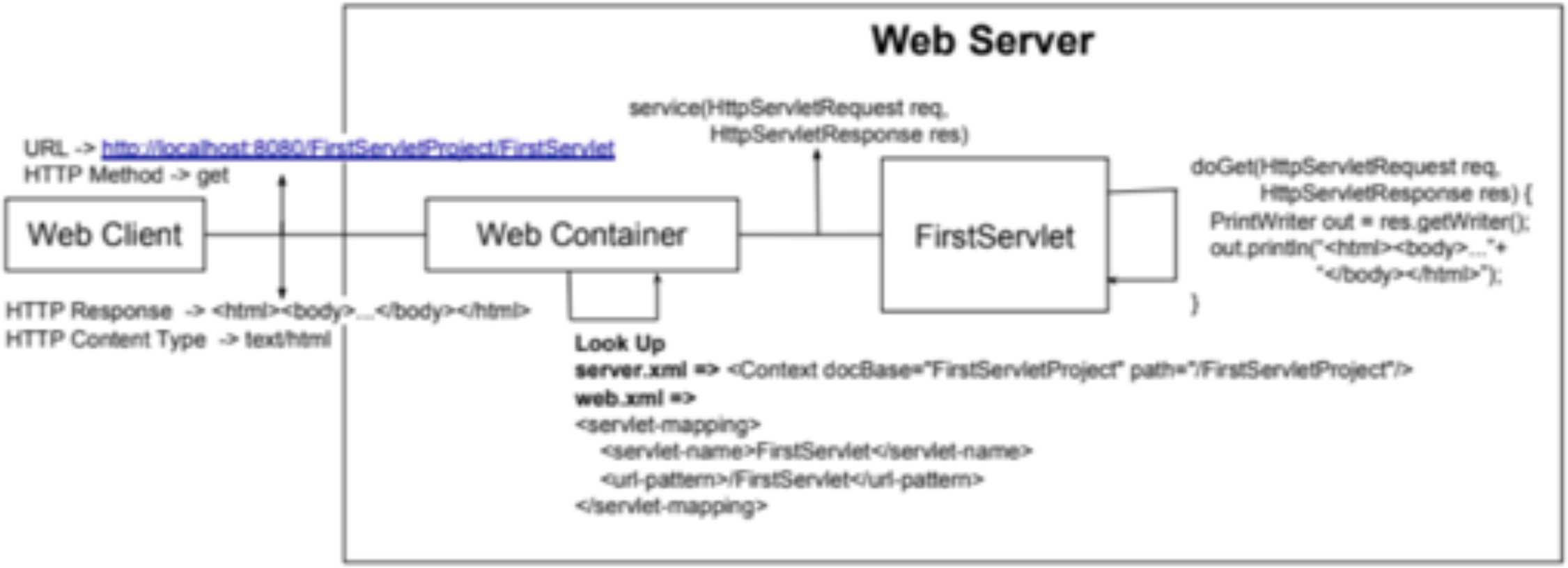
Note – ***ls -l /usr/local/Cellar/tomcat/9.0.30/libexec/webapps/examples/***

```
Tomcat $ ls -p /usr/local/Cellar/tomcat/9.0.30/libexec/webapps/examples/
WEB-INF/        index.html      jsp/            servlets/       websocket/
```

Understanding URL



Web Container



Word done by Web Container

- Communication Support
- Lifecycle Management
- Resource Management
- Multithreading Support
- JSP Support
- Others – Resource Pooling, Security, Multi Apps, Memory Optimization, Garbage Collection, Hot Deployment, etc..

Web Application Directory Structure

- **Source –** Contains a Source Files
- **POM –** Project Object Model file of all project dependencies
- Target – Packaged as WAR file (Web Archive)
  - META-INF – MANIFEST.MF
  - WEB-INF – Compiles Classes, web.xml, jsp and html files

UC 1

Create Function Module for User Registration System.

BridgeLabz

Employability Delivered

Thank You