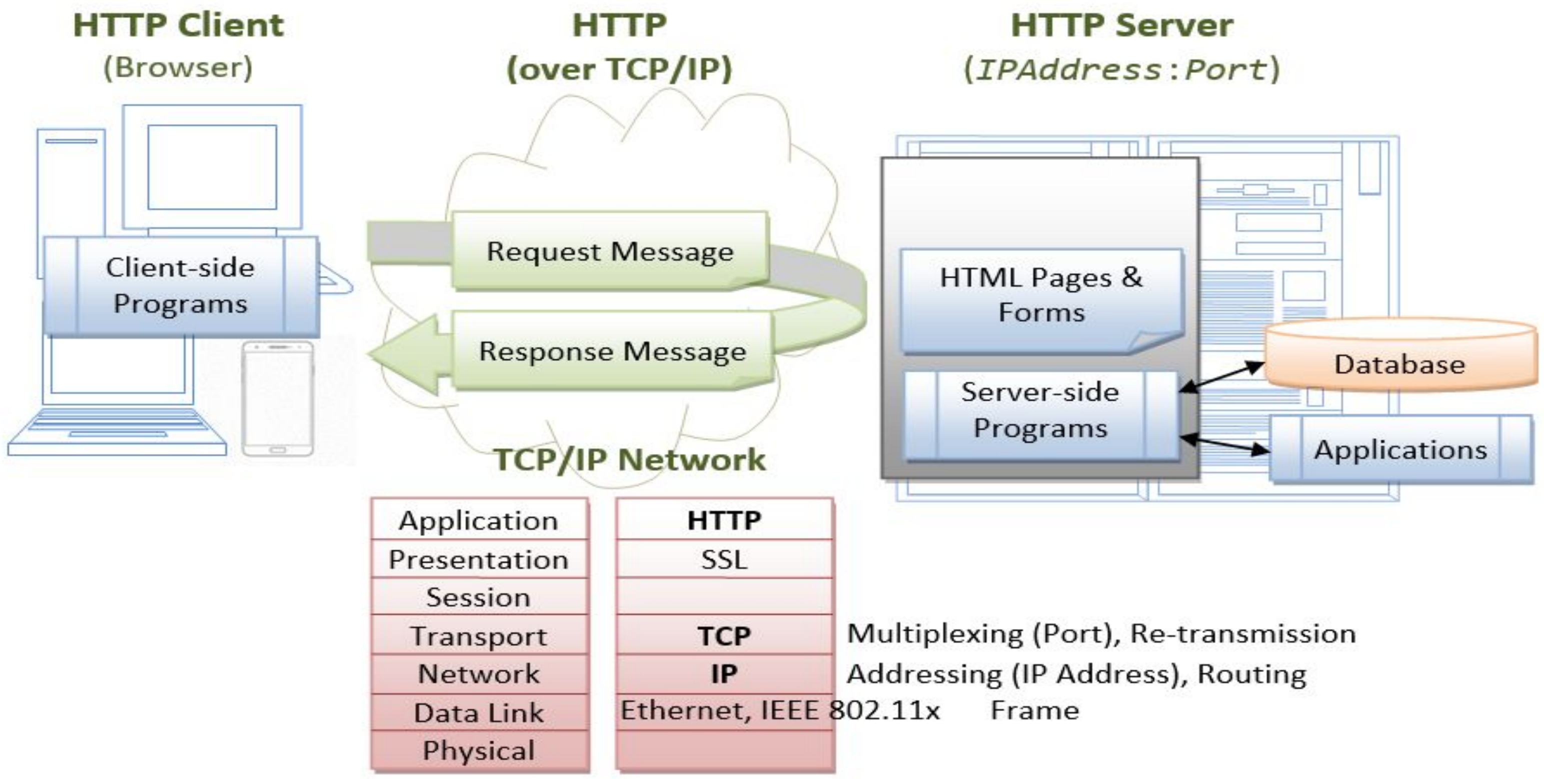
JAVA II

Servlets and JSP as a core for Web applications in Java

How does the web works :/

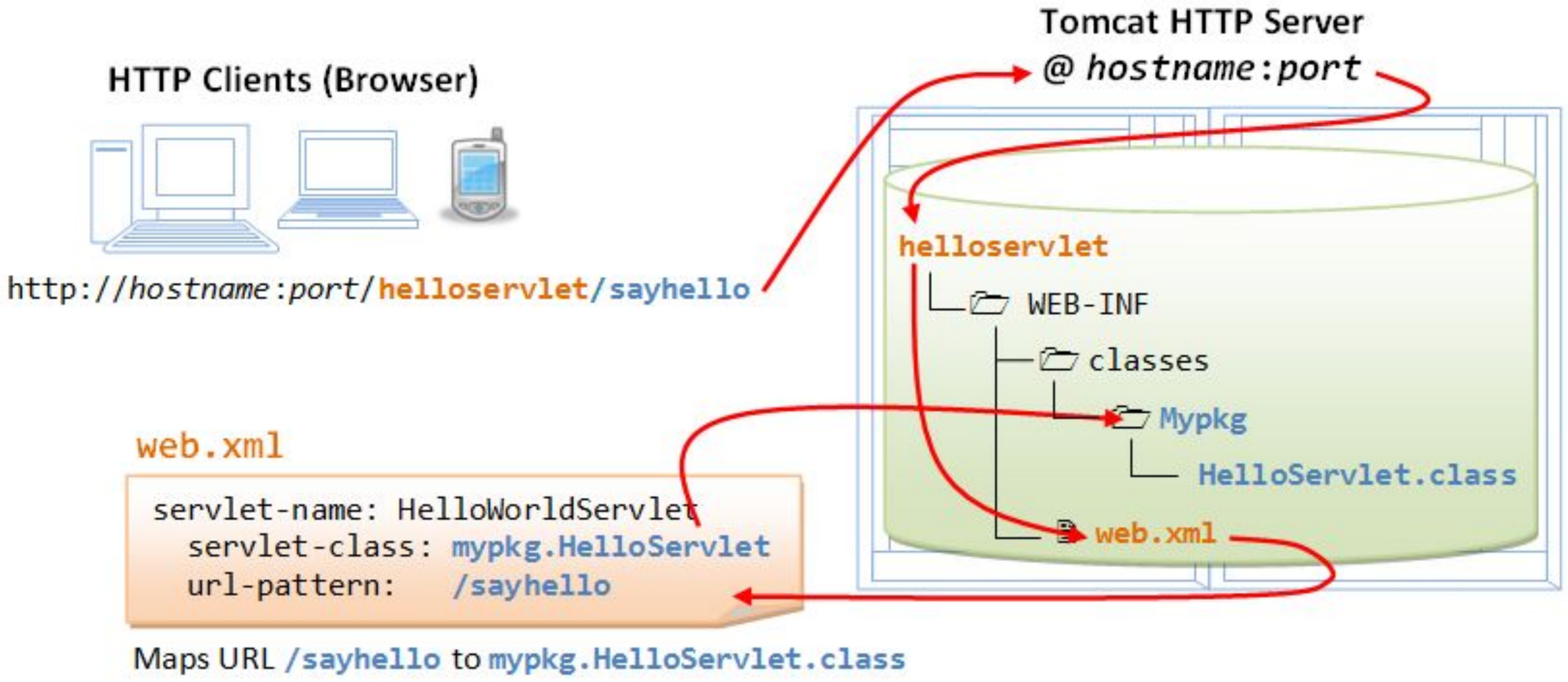


Application	**HTTP**	
Presentation	SSL	
Session		
Transport	**TCP**	Multiplexing (Port), Re-transmission
Network	**IP**	Addressing (IP Address), Routing
Data Link	Ethernet, IEEE 802.11x	Frame
Physical		

Java Servlet - description

A Servlet is a Java application programming interface (API) running on the application server which can intercept requests made by the client and can generate/send a response accordingly. A well-known example is the HttpServlet which provides methods to hook on HTTP requests using the popular HTTP methods such as GET and POST. You can configure HttpServlet to listen on a certain HTTP URL pattern, which is configurable in web.xml, or more recently with Java EE, with @WebServlet annotation. Many Java EE web frameworks are built on top of servlets, such as JSF, JAX-RS, Spring MVC, Struts, Wicket, etcetera.

How to write Java Servlet and how it is mapped to URL



```
servlet-name: HelloWorldServlet
  servlet-class: mypkg.HelloServlet
  url-pattern:    /sayhello
```

Maps URL /sayhello to mypkg.HelloServlet.class

How to write Java Servlet and how it is mapped to URL

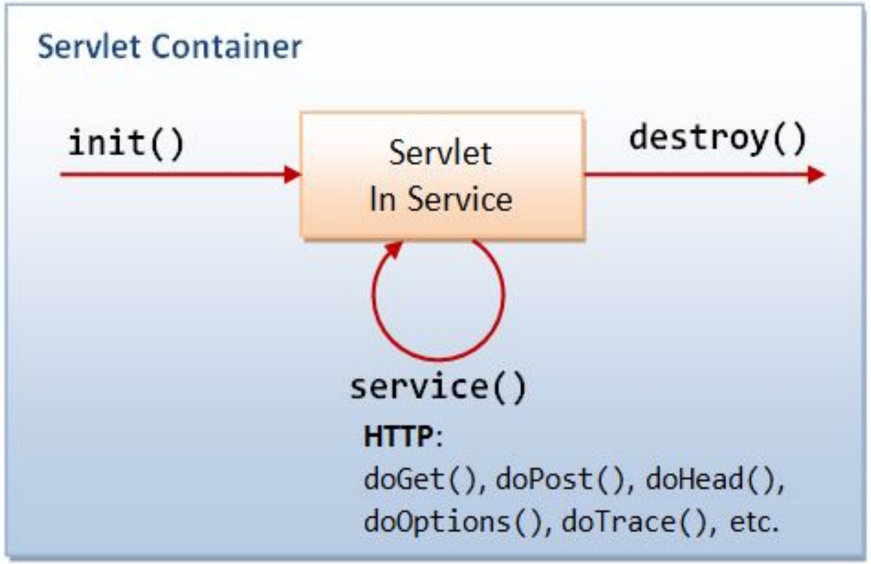
- git clone https://viktornar@bitbucket.org/viktornar/baltictalents.git
- cd baltictalents\lesson2
- mvn install
- http://localhost:8080/lesson2/first-servlet
- http://localhost:8080/lesson2/second-servlet
- http://localhost:8080/lesson2/html-from-servlet/test?test=test&hello=viktor

Where are java servlets (FirstServlet, SecondServlet and etc.) in project source code

What is the difference between them?

Java Servlet lifecycle

When a Servlet is requested for the first time or when the web app starts up, the servlet container (GlassFish, Tomcat, Jetty) will create an instance of it and keep it in memory during the web app's lifetime. The same instance will be reused for every incoming request whose URL matches the servlet URL pattern. You can access the requested data by HttpServletRequest and handle the response by HttpServletResponse. Both objects are available as method arguments inside of any of the overridden methods of HttpServlet, such as doGet() to preprocess a request and doPost() to post-process a request.



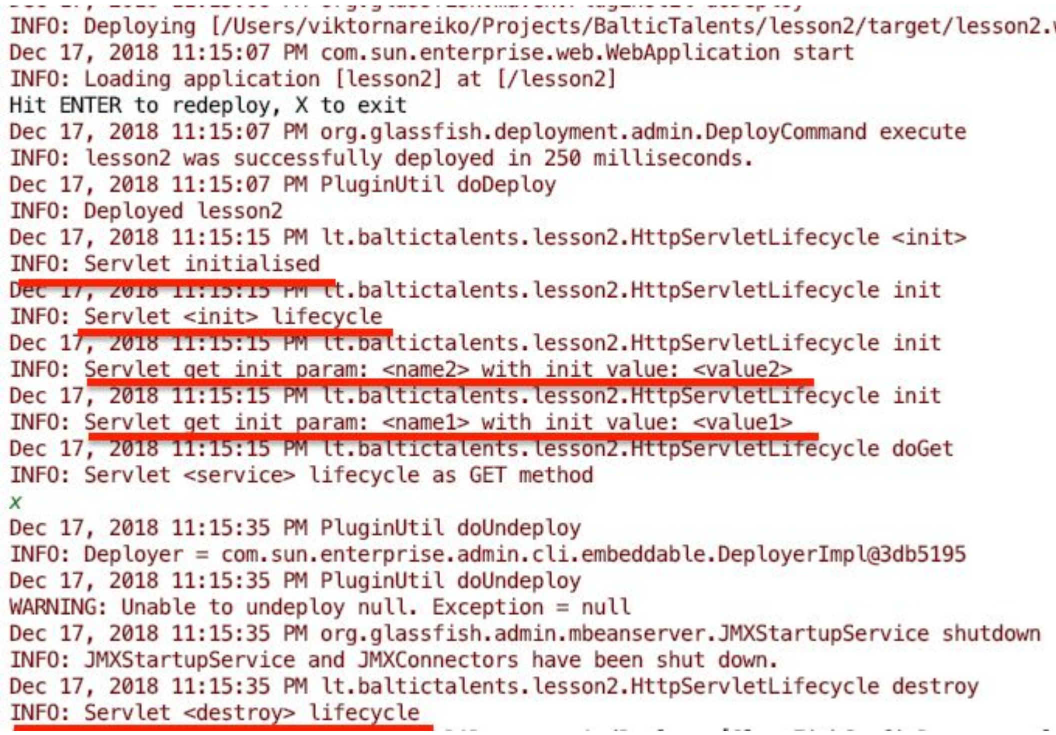
Java Servlet lifecycle

The servlet container invokes the init(ServletConfig) method of the servlet, providing a ServletConfig object as an argument. init() runs only once. It is usually used to read persistent configuration data and initialize costly resource. This ServletConfig object allows the servlet to access initialization parameters for this particular servlet.

```
<servlet>
                    <servlet-name>ServletName</servlet-name>
             <servlet-class>ServletClassFile</servlet-class>
                                            <init-param>
                          <param-name>initParam1</param-name>
                   <param-value>initParam1Value</param-value>
                                           </init-param>
                                            <init-param>
                          <param-name>initParam2</param-name>
                   <param-value>initParam2Value</param-value>
                                           </init-param>
</servlet>
```

Java Servlet lifecycle example

- stop glassfish
- mvn install
- clear console
- http://localhost:8080/lesson2/servlet-lifecycle
- Do console output analyze



```
INFO: Deploying [/Users/viktornareiko/Projects/BalticTalents/lesson2/target/lesson2.
Dec 17, 2018 11:15:07 PM com.sun.enterprise.web.WebApplication start
INFO: Loading application [lesson2] at [/lesson2]
Hit ENTER to redeploy, X to exit
Dec 17, 2018 11:15:07 PM org.glassfish.deployment.admin.DeployCommand execute
INFO: lesson2 was successfully deployed in 250 milliseconds.
Dec 17, 2018 11:15:07 PM PluginUtil doDeploy
INFO: Deployed lesson2
Dec 17, 2018 11:15:15 PM lt.baltictalents.lesson2.HttpServletLifecycle <init>
INFO: Servlet initialised
Dec 17, 2018 11:15:15 PM lt.baltictalents.lesson2.HttpServletLifecycle init
INFO: Servlet <init> lifecycle
Dec 17, 2018 11:15:15 PM lt.baltictalents.lesson2.HttpServletLifecycle init
INFO: Servlet get init param: <name2> with init value: <value2>
Dec 17, 2018 11:15:15 PM lt.baltictalents.lesson2.HttpServletLifecycle init
INFO: Servlet get init param: <name1> with init value: <value1>
Dec 17, 2018 11:15:15 PM lt.baltictalents.lesson2.HttpServletLifecycle doGet
INFO: Servlet <service> lifecycle as GET method
x
Dec 17, 2018 11:15:35 PM PluginUtil doUndeploy
INFO: Deployer = com.sun.enterprise.admin.cli.embeddable.DeployerImpl@3db5195
Dec 17, 2018 11:15:35 PM PluginUtil doUndeploy
WARNING: Unable to undeploy null. Exception = null
Dec 17, 2018 11:15:35 PM org.glassfish.admin.mbeanserver.JMXStartupService shutdown
INFO: JMXStartupService and JMXConnectors have been shut down.
Dec 17, 2018 11:15:35 PM lt.baltictalents.lesson2.HttpServletLifecycle destroy
INFO: Servlet <destroy> lifecycle
```

Interface ServletContext

The ServletContext interface defines a servlet's view of the webapp (or web context) in which it is running (a better name is actually ApplicationContext). Via the ServletContext object, a servlet can communicate with the container, e.g., write to event log, get the URL reference to resources, and get and set attributes that other servlets in the same context can access. There is one ServletContext object for each web application deployed into a container. You can specify initialization parameters for a web context (that are available to all the servlet under the web context) in the web application deployment descriptor, e.g.,

- http://localhost:8080/lesson2/third-servlet
- http://localhost:8080/lesson2/nth-servlet

```
<web-app                                                    ......>
                                                     <context-param>
                                  <param-name>jdbcDriver</param-name>

<param-value>com.mysql.jdbc.Driver</param-value>
                                                    </context-param>
                                                     <context-param>
                                 <param-name>databaseUrl</param-name>

<param-value>jdbc:mysql://localhost/eshop</param-va
lue>
                                                    </context-param>
......
</web-app>
```

Dispatch Request - RequestDispatcher

When building a web application, it is often useful to forward a request to another servlet, or to include the output of another servlet in the response. The **RequestDispatcher** interface supports these. The **RequestDispatcher** can be obtained via **ServletContext**

```
//                                                                    ServletContext
RequestDispatcher           getRequestDispatcher(String           servletPath)
RequestDispatcher getNamedDispatcher(String servletName)
```

```
RequestDispatcher       rd        =       request.getRequestDispatcher("/test.jsp?isbn=123");
rd.include(request,                                                                response);
//                                                                                        or
rd.forward(request, response);
```

- http://localhost:8080/lesson2/form-servlet

Filters in Java Servlet context

A filter is a reusable piece of code that can transform the content of HTTP requests, responses, and header information. Examples of filtering components are:

- Authentication filters
- Logging and auditing filters
- Image conversion filters
- Data compression filters
- Encryption filters
- Tokenizing filters

```
public class RequestTimerFilter implements Filter {
  private static final Logger LOGGER =
Logger.getLogger(RequestTimerFilter.class.getName());

  @Override
  public void init(FilterConfig filterConfig) throws ServletException {
     LOGGER.info("RequestTimerFilter initialized");
  }

  @Override
  public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException, ServletException {
     long before = System.currentTimeMillis();
     chain.doFilter(request, response);
     long after = System.currentTimeMillis();
     String path = ((HttpServletRequest)request).getRequestURI();
     LOGGER.info(String.format("%s: %s msec", path, (after - before)));
  }

  @Override
  public void destroy() {
     LOGGER.info("RequestTimerFilter destroyed");
  }
}
```

Session in Java Servlet context

Java Servlet API provides a session tracking facility, via an interface called javax.servlet.http.HttpSession. It allows servlets to:

- View and manipulate information about a session, such as the session identifier, creation time, and last accessed time.
- Bind objects to sessions, allowing user information to persist across multiple user requests.

```
// Retrieve the current session, create one if not exists
HttpSession session = request.getSession(true);
// Place the shopping cart inside the session
synchronized (session) { // synchronized to prevent concurrent updates
    session.setAttribute("cartName", "some fancy cart name");
}
```

Any page within the session can retrieve the shopping cart:

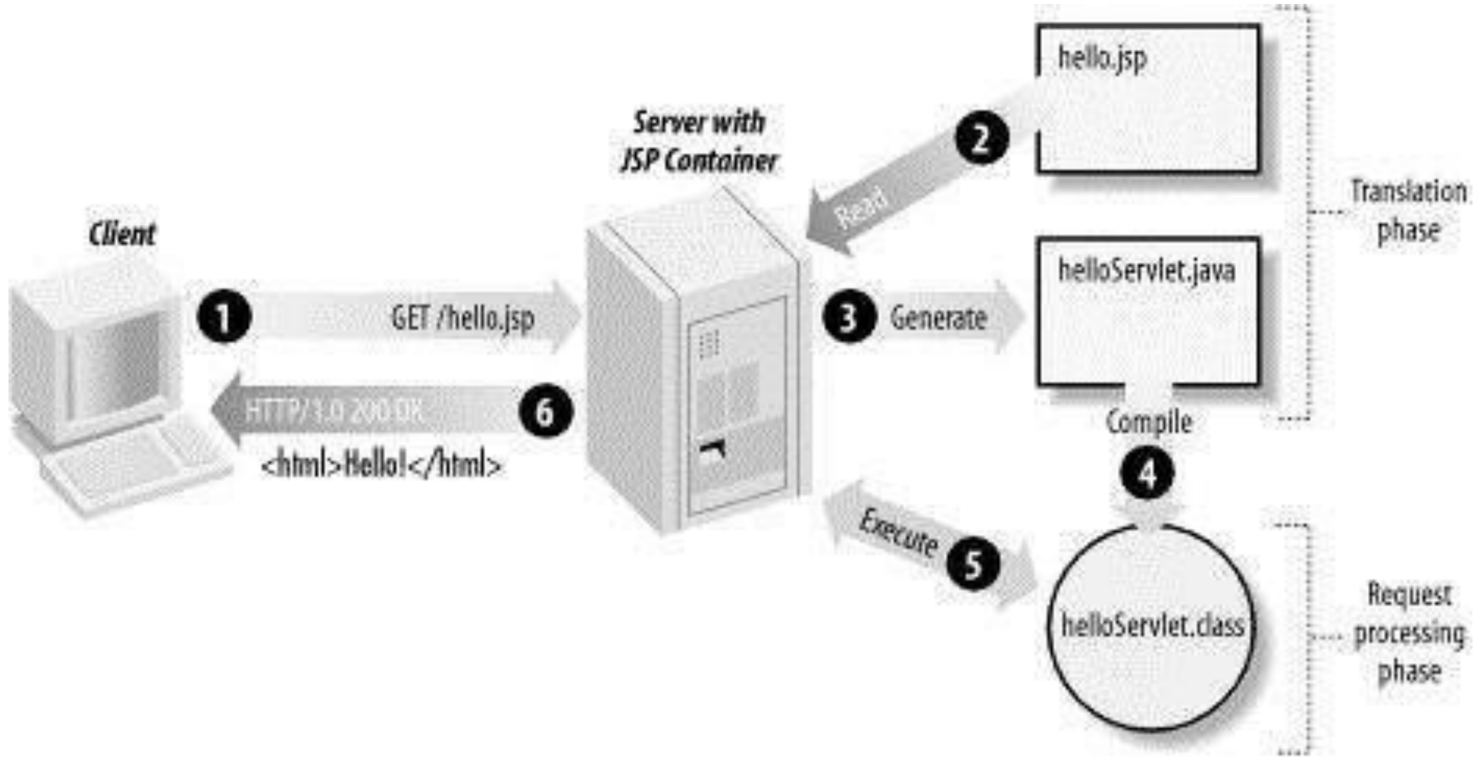
```
// Retrieve the current session, do not create new session
HttpSession session = request.getSession(false);
if (session != null) {
    String cartName = (String)session.getAttribute("cartName");
}
```

JSP as view templating engine

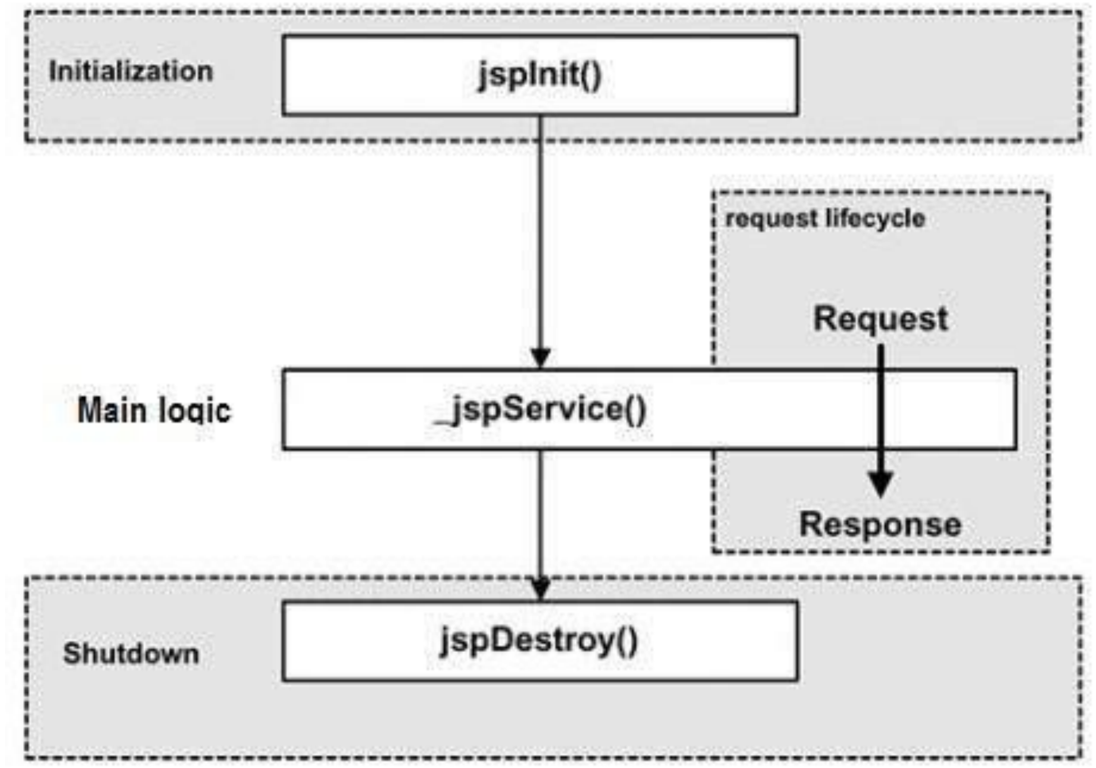
Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. This tutorial will teach you how to use Java Server Pages to develop your web applications in simple and easy steps.

https://www.tutorialspoint.com/jsp/

JSP Architecture



JSP lifecycle



How to use JSP

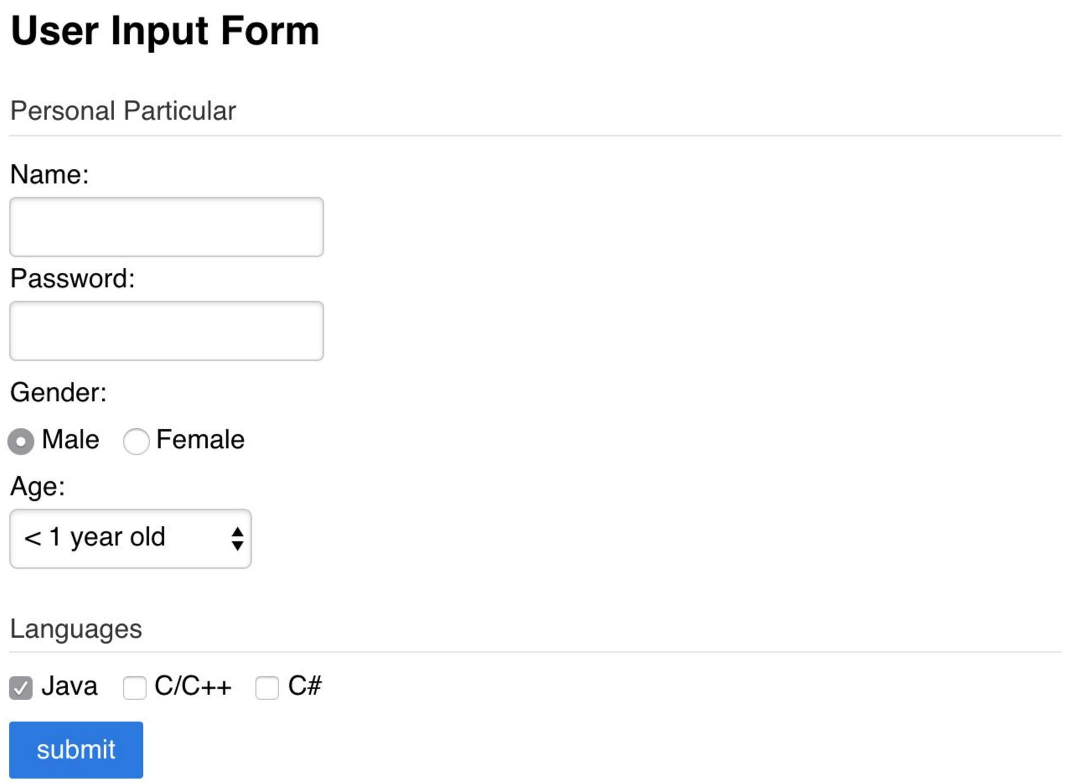
Look at:

- https://www.tutorialspoint.com/jsp/jsp_syntax.htm
- https://www.tutorialspoint.com/jsp/jsp_java_beans.htm
- https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm

Look at for real life example:

- lesson2/src/webapp/example.jsp

Let’s build an application :)



User Input Form

Personal Particular

Name:

Password:

Gender:

● Male ○ Female

Age:

< 1 year old

Languages

☑ Java ☐ C/C++ ☐ C#

submit

Build a similar application (homework)

Build a similar application, but only for collecting data about cars. You should be able to enter:

- car type;
- car years;
- car engine type;
- car brand;
- something else;