University of Miskolc
Faculty of Mechanical Engineering and Informatics

Java Web Application Development Technology

N13020008

Servlet technology

Tamás Tompa, PhD
assistant professor
Department of Information Technology
University of Miskolc

2024.

What are Servlets?

- Servlets provide a **component-based, platform-independent method for building Web-based applications**
- Servlets **have access to the entire family of Java APIs**, including the JDBC API to access enterprise databases
- Using Servlets, you can **collect input from users through web page forms**, present records from a database or another source, and **create web pages dynamically**
- Java Servlets often serve the same purpose as programs implemented using the Common Gateway Interface (CGI)

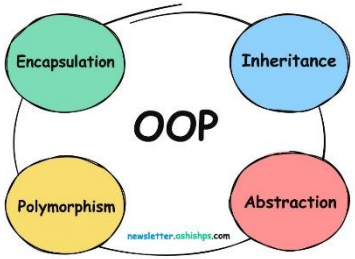
What are Java Servlets?

- **Java Servlets are**
  - **programs (classes)** that run on a Web or Application server
  - **act as a middle layer** between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server

- Servlets can be created using the **javax.servlet** and **javax.servlet.http** packages
  - which are a standard part of the Java's enterprise edition

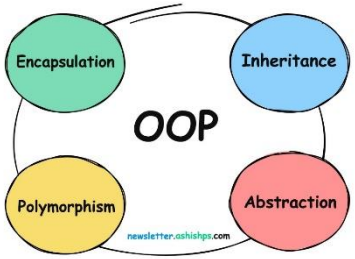
- Tasks
  - read the explicit data sent by the clients, read the implicit HTTP request data sent by the clients, process the data and generate the results, send the explicit data (i.e., the document), send the implicit HTTP response, and so on…

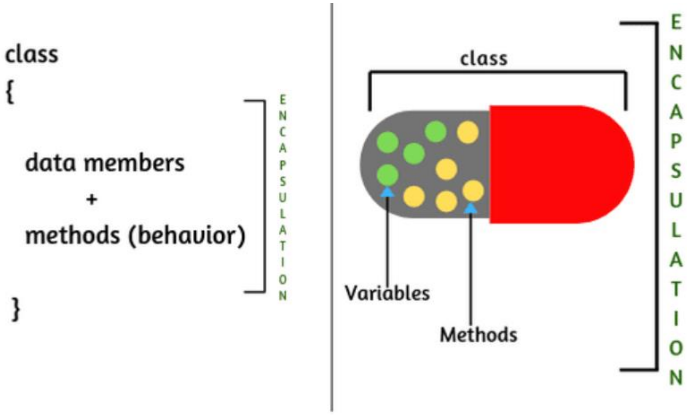
OOP principles

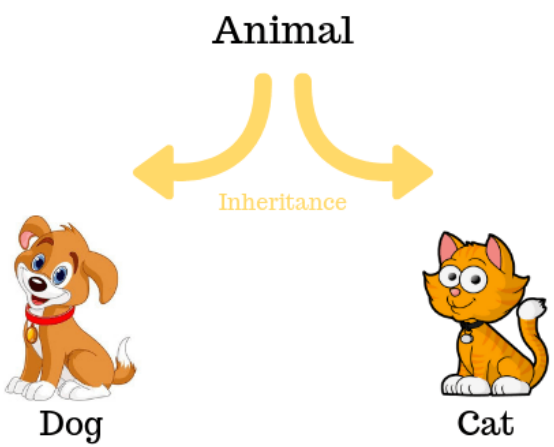


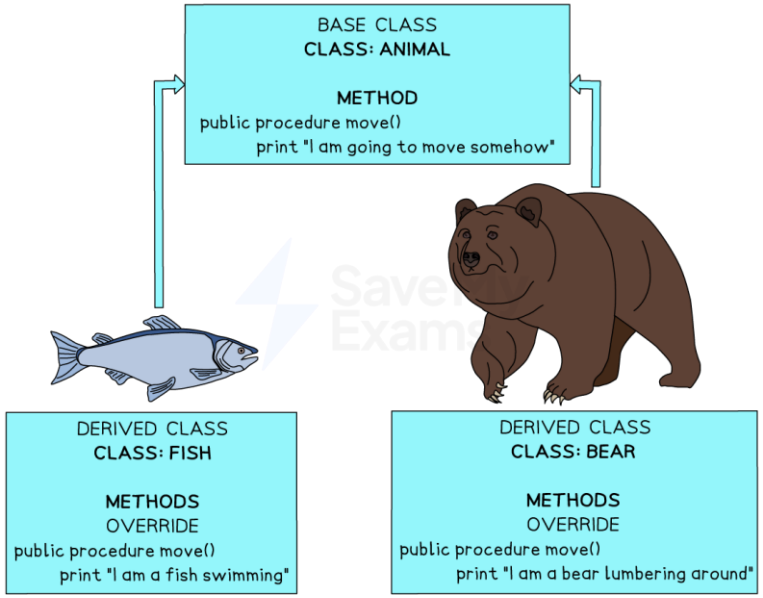
- Objects contain data, referred to as attributes or properties, and methods
- OOP allows objects to interact with each other using four basic principles:
  - **encapsulation**
    - data into a structured unit, along with the methods used to work with that data
  - **inheritance**
    - mechanism that allows a class to inherit properties and behaviors from another class
  - **abstraction**
    - used to hide unnecessary information and display only necessary information to the users interacting
  - **polymorphism**
    - allows a specific routine to use variables of different types at different times, gives a program the ability to redefine methods for derived classes

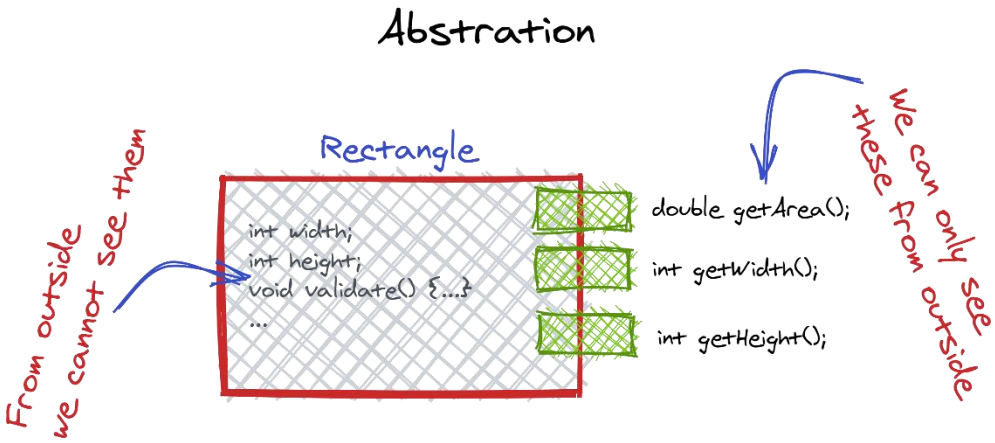
OOP principles





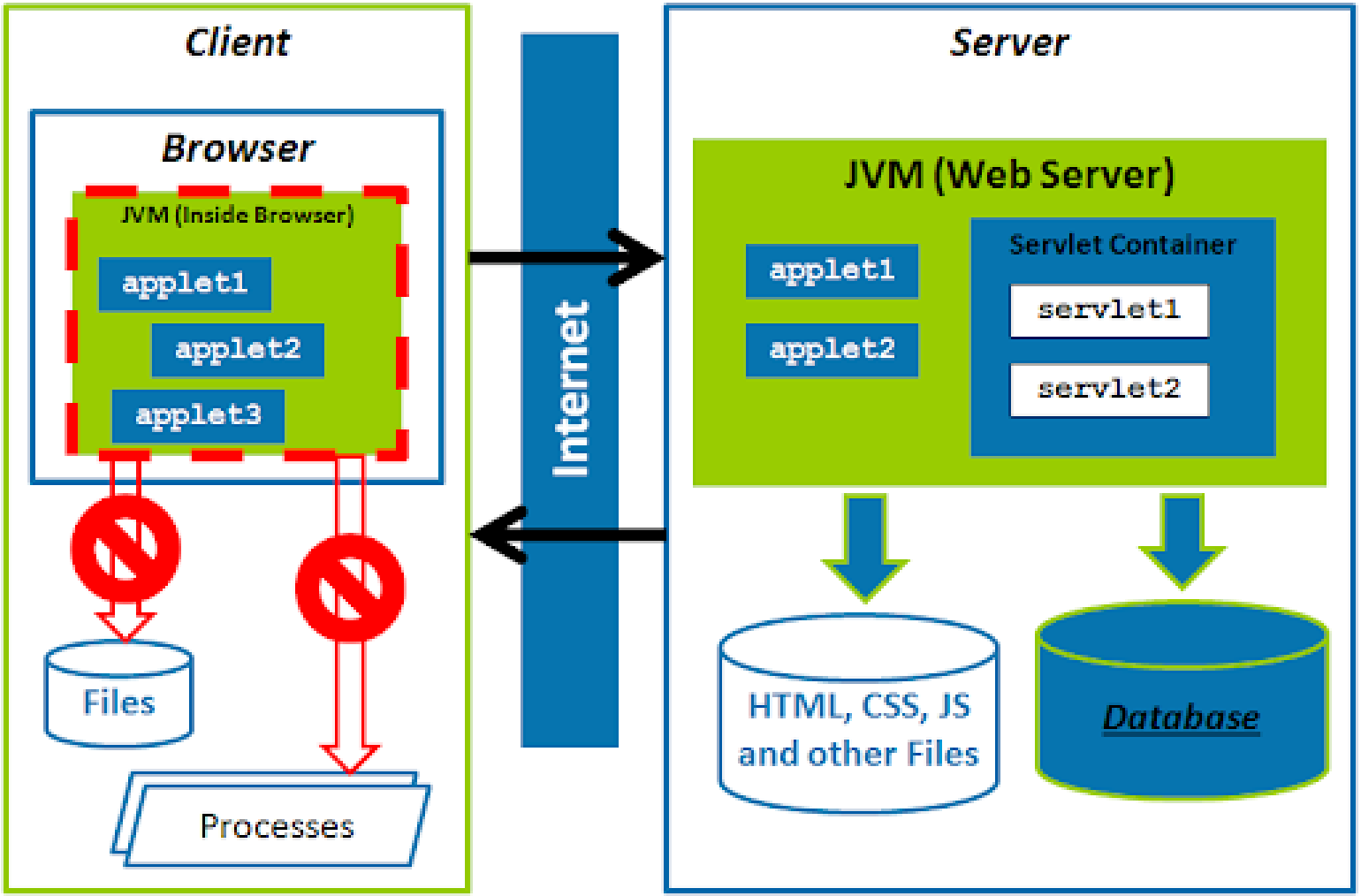






AlgoDaily

Servlets Architecture



Environment Setup

- **JDK (Java Development Kit)**
  - download an implementation of the Java Software Development Kit (SDK)
  - setup PATH environment variable appropriately

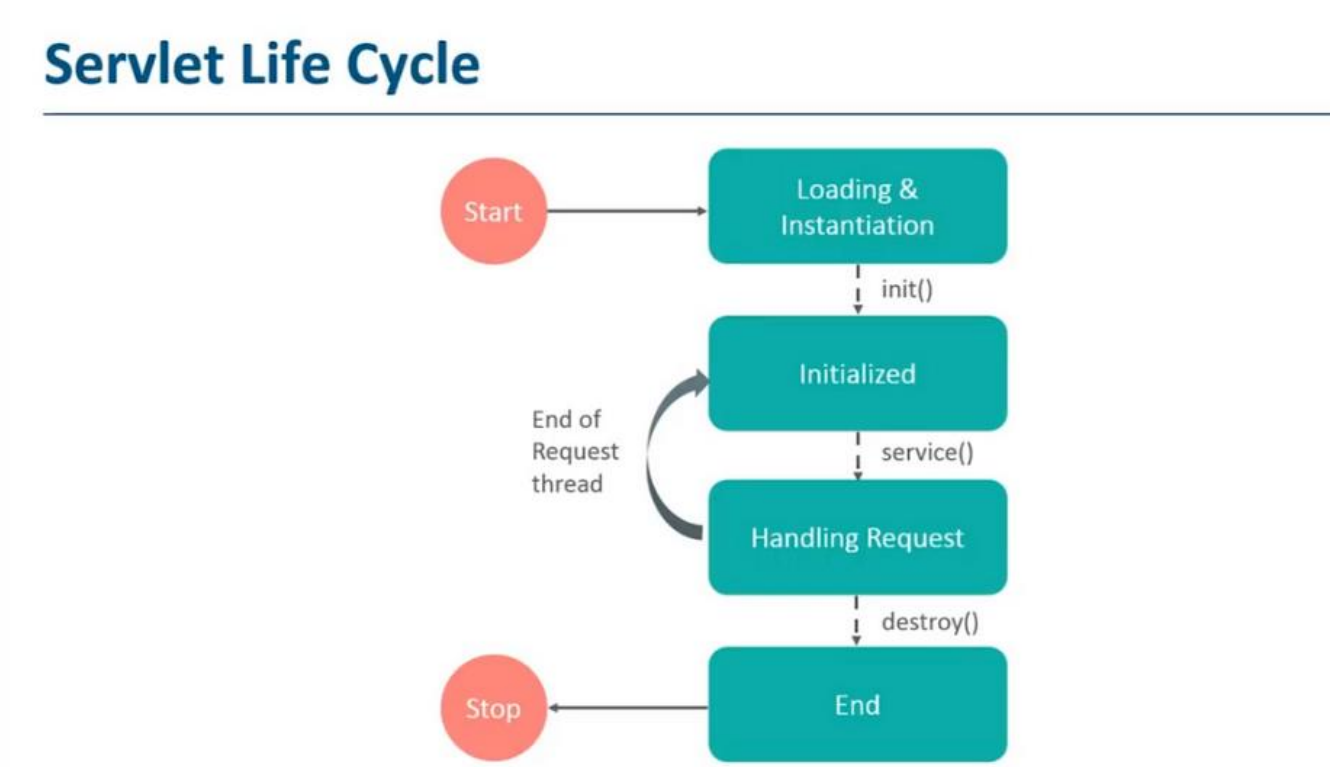
- **Web Server – Tomcat**
  - download and install
  - setup PATH environment variable appropriately
  - these steps were introduced in the JSP course…

- **Eclipse IDE**
  - https://eclipseide.org/
  - Eclipse IDE for Java EE Developers
  - https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2024-09/R/eclipse-jee-2024-09-R-win32-x86_64.zip

Life Cycle

Servlets

- **The life cycle of Servlets:**
  - The servlet is initialized by calling the **init()** method
  - The servlet calls **service()** method to process a client's request
  - The servlet is terminated by calling the **destroy()** method
  - Finally, servlet is garbage collected by the garbage collector of the JVM



Life Cycle

- **The life cycle of Servlets:**
  - The servlet is initialized by calling the **init()** method
    - called only once, when the servlet is created
    - servlet is normally created when a user first invokes a URL corresponding to the servlet
    - when a user invokes a servlet, a single instance of each servlet gets created
    - each user request resulting in a new thread that is handed off to `doGet` or `doPost` as appropriate

```
public void init() throws ServletException {
        // Initialization code...
}
```

Life Cycle

- **The life cycle of Servlets:**
  - The servlet calls **service()** method to process a client's request
    - the main method to perform the actual task
    - servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client
    - each time the server receives a request for a servlet, the server spawns a new thread and calls service
    - service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls `doGet`, `doPost`, `doPut`, `doDelete`, etc. methods as appropriate

```
public void service(ServletRequest request, ServletResponse response)
throws ServletException, IOException {

}
```

Life Cycle

- **The life cycle of Servlets:**
  - The servlet calls **service()** method to process a client's request
    - **A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by `doGet()` method**

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // Servlet code
}
```

Life Cycle

- **The life cycle of Servlets:**
  - The servlet calls **service()** method to process a client's request
    - **A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by `doPost()` method**

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // Servlet code
}
```

Life Cycle

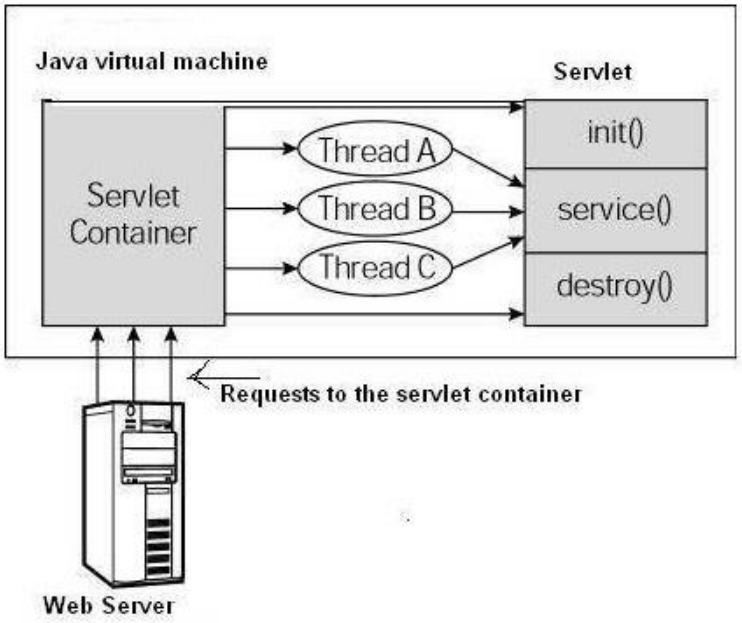
- **The life cycle of Servlets:**
  - The servlet is terminated by calling the **destroy()** method
    - called only once at the end of the life cycle of a servlet
    - it gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities
    - the servlet object is marked for garbage collection

```
public void destroy() {
        // Finalization code...
}
```

Life Cycle

Servlets

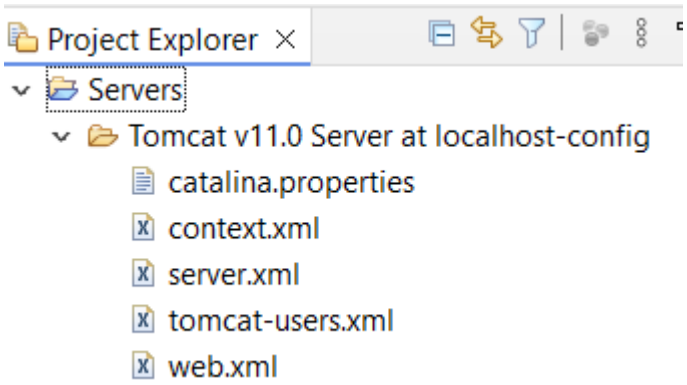
- **The life cycle of Servlets:**
  - First the HTTP requests coming to the server are delegated to the servlet container
  - The servlet container loads the servlet before invoking the service() method
  - Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet

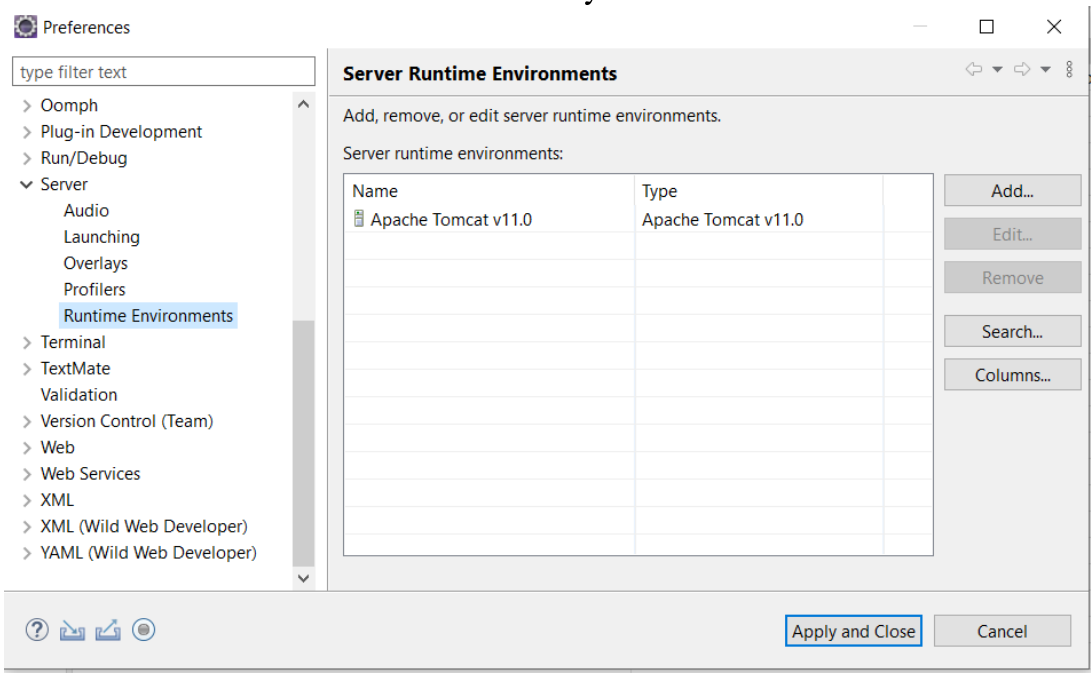


Create new project in Eclipse

Servlets

- **Add the TomCat server to Eclipse:**
  - Go to the “Window” menu -> “Preferences”
  - Expand “Server” -> “Runtime Environments”
  - Click on “Add…” to add a new server runtime environment
  - Select “Apache Tomcat” from the list of server types
  - Click “Next”
  - Browse and select the Tomcat installation directory
  - Click “Finish”

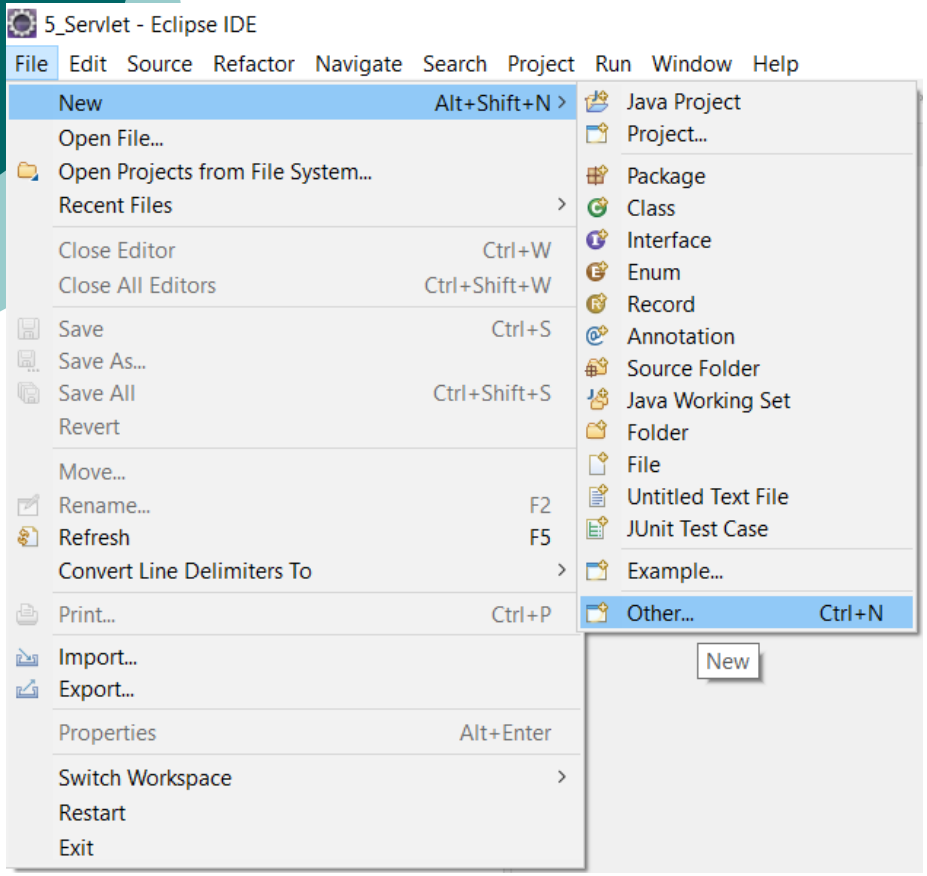


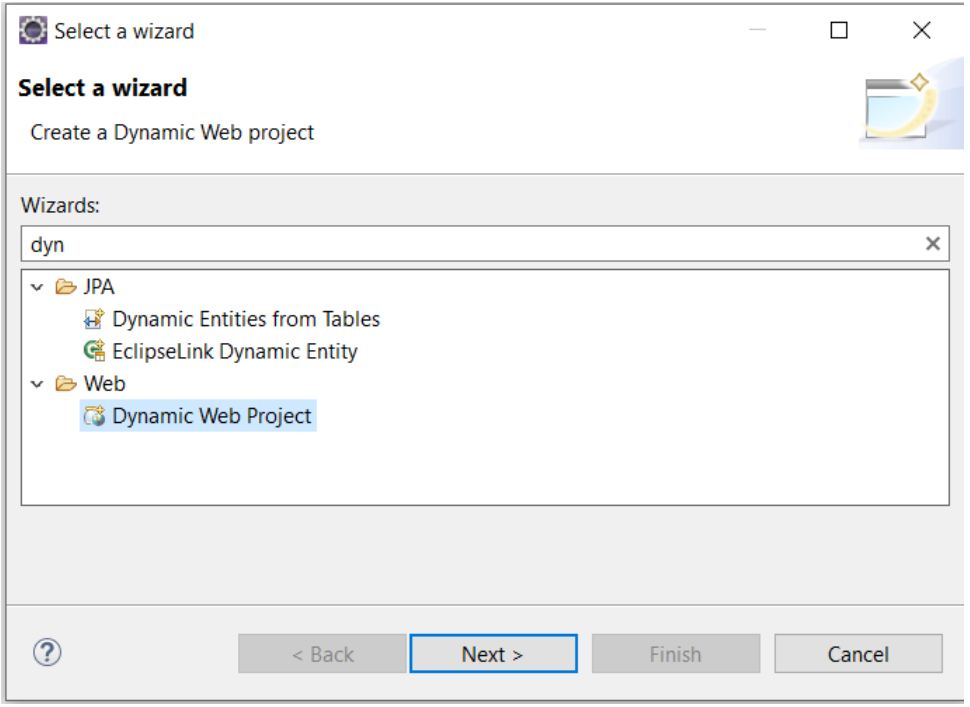


Create new project in Eclipse

Servlets

- **Create the Dynamic Web project:**

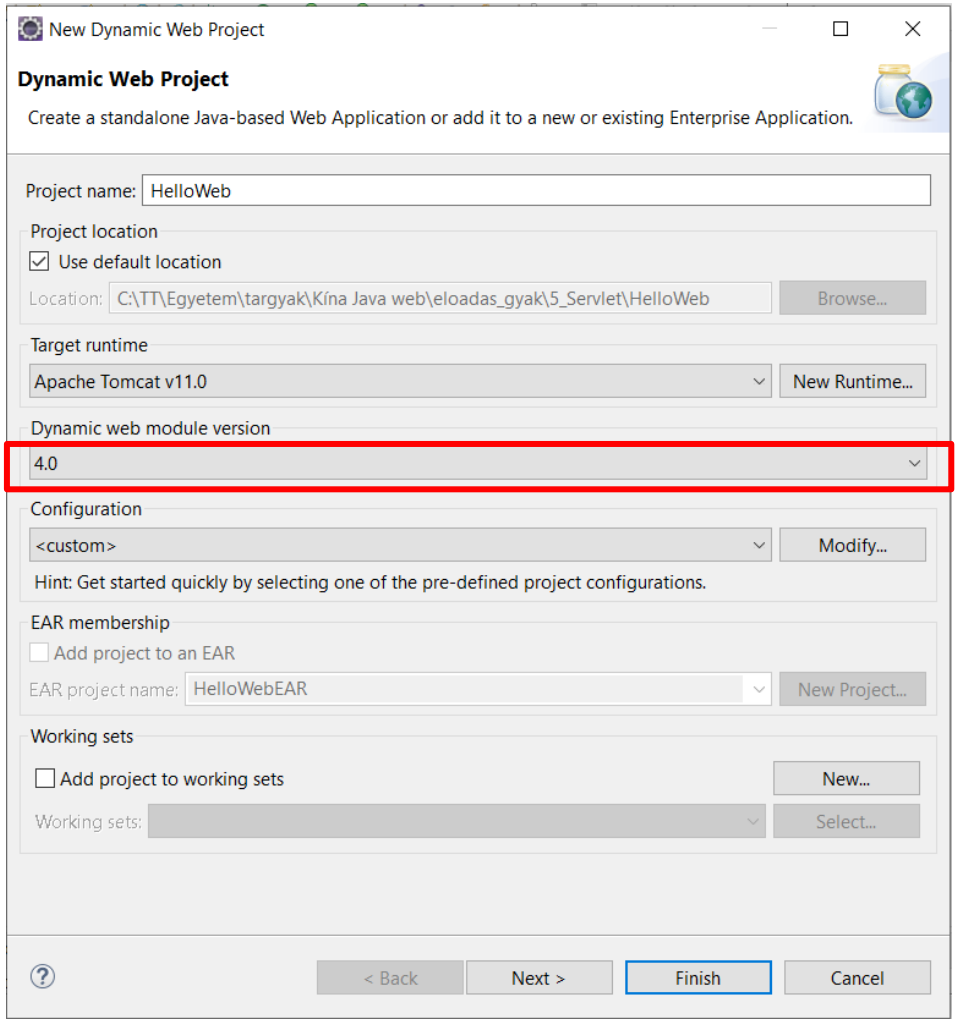




Create new project in Eclipse

Servlets

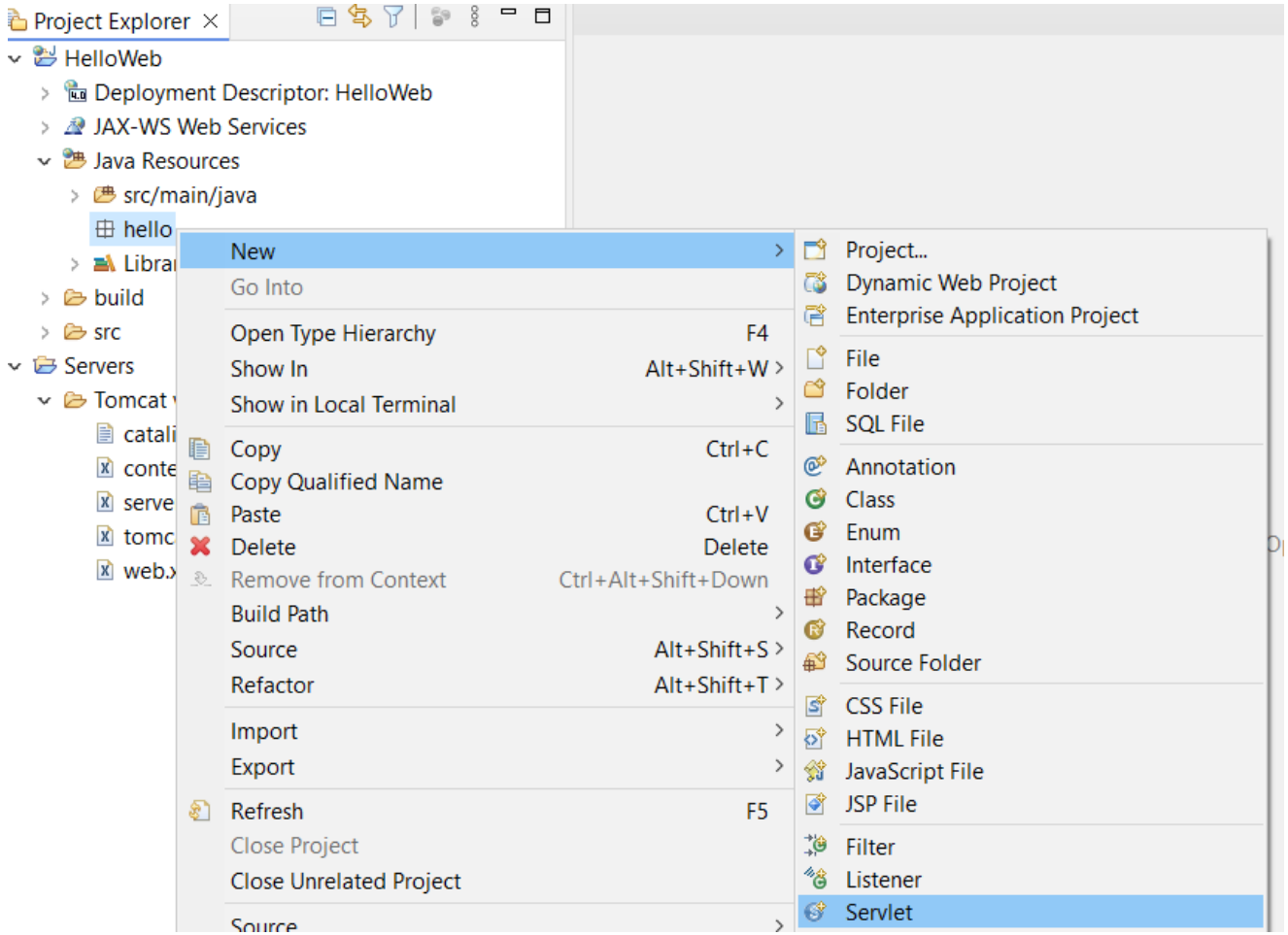
- **Create the Dynamic Web project:**



Create new project in Eclipse

Servlets

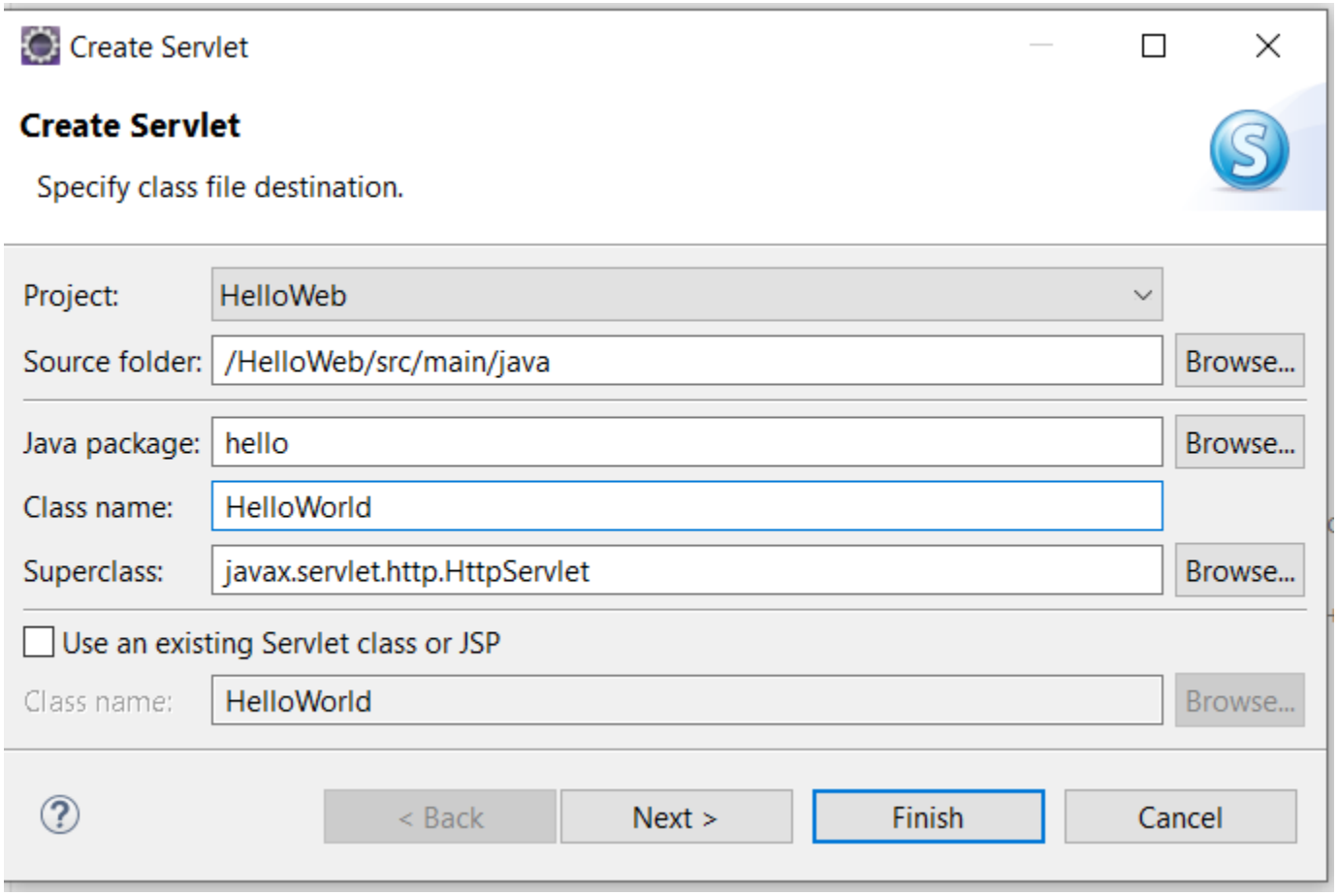
- **Create the Dynamic Web project:**



Create new project in Eclipse

Servlets

- **Create the Dynamic Web project:**



Create new project in Eclipse

Servlets

- **Create the Dynamic Web project:**

Project Explorer

- HelloWeb
  - Deployment Descriptor: HelloWeb
  - JAX-WS Web Services
  - Java Resources
    - src/main/java
      - hello
        - HelloWorld.java
    - Libraries
  - build
  - src
- Servers
  - Tomcat v11.0 Server at localhost-config
    - catalina.properties
    - context.xml
    - server.xml
    - tomcat-users.xml
    - web.xml

HelloWorld.java

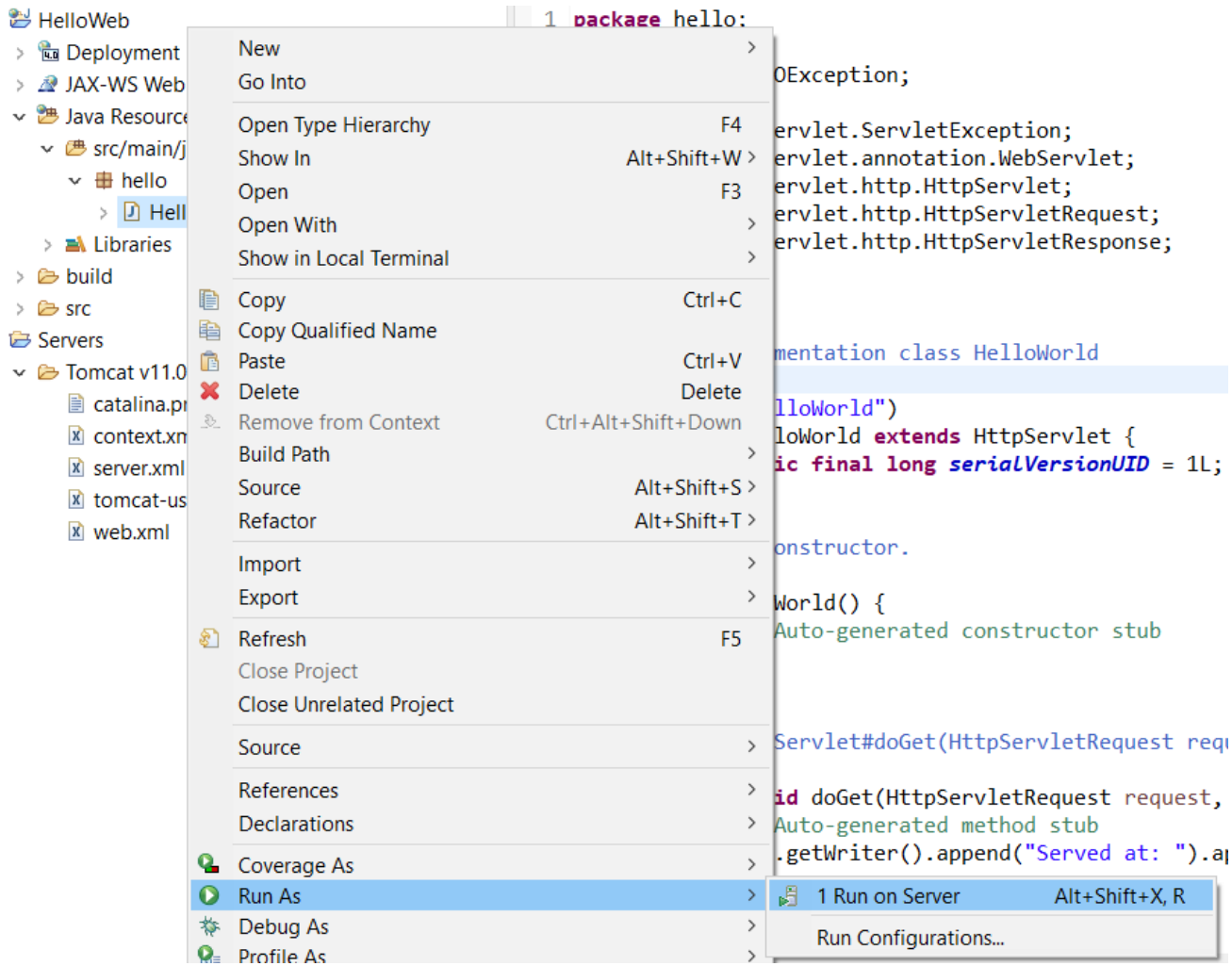
```java
package hello;

import java.io.IOException;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;


/**
 * Servlet implementation class HelloWorld
 */
@WebServlet("/HelloWorld")
public class HelloWorld extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * Default constructor.
     */
    public HelloWorld() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
```

change to jakarta

Servlet URL: http://localhost:8080/HelloWeb/HelloWorld

Create new project in Eclipse

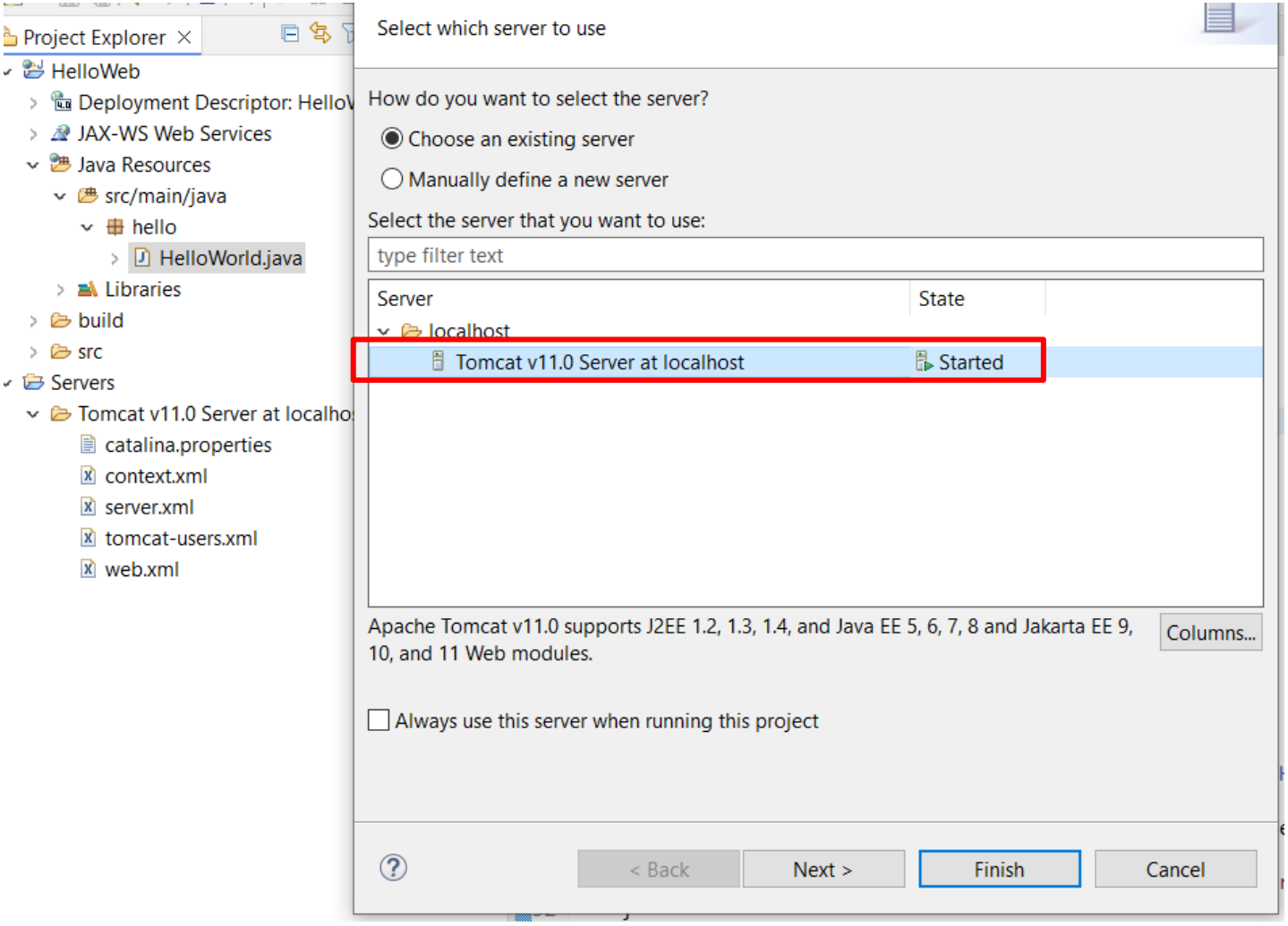
- **Run the Dynamic Web project:**



Create new project in Eclipse

Servlets

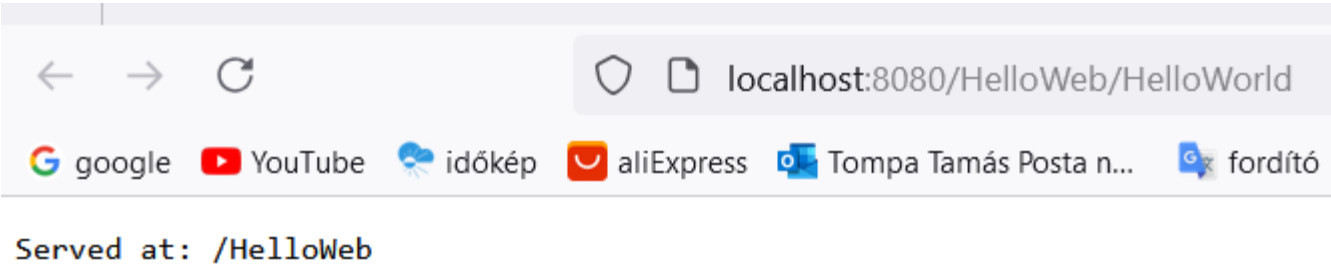
- **Run the Dynamic Web project:**



Create new project in Eclipse

Servlets

- **Run the Dynamic Web project:**



Create new project in Eclipse

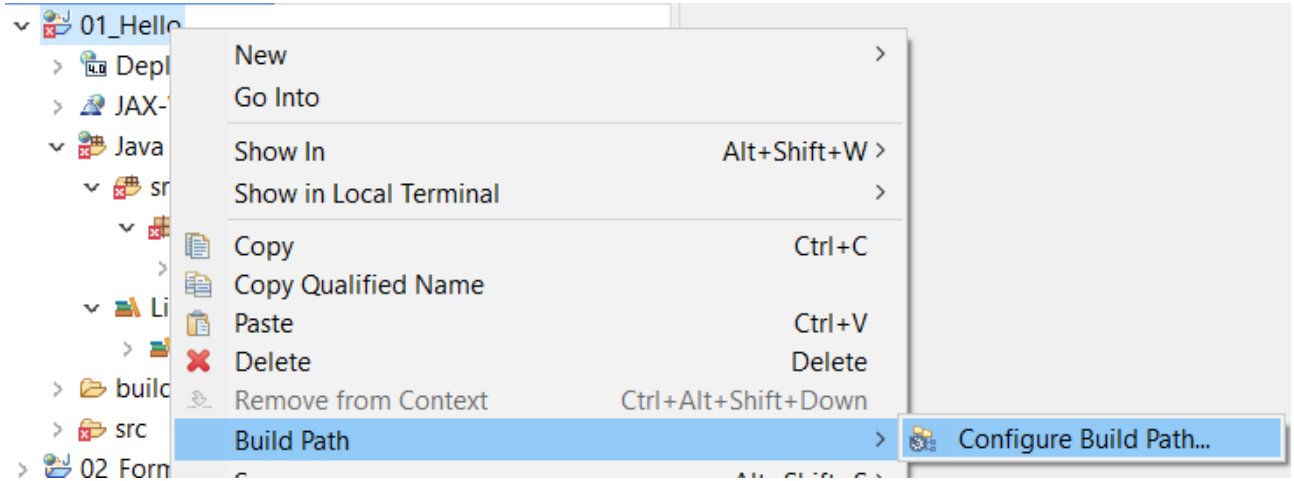
Servlets

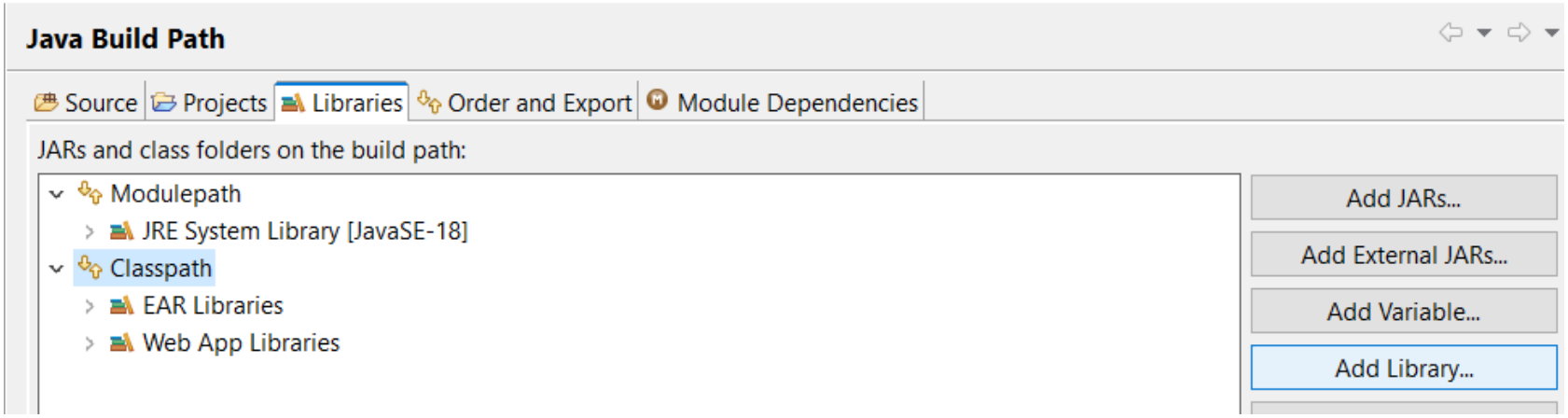
- **Add server libary to the project (if it is necessary):**

Create new project in Eclipse

Servlets

- **Add server libary to the project (if it is necessary):**

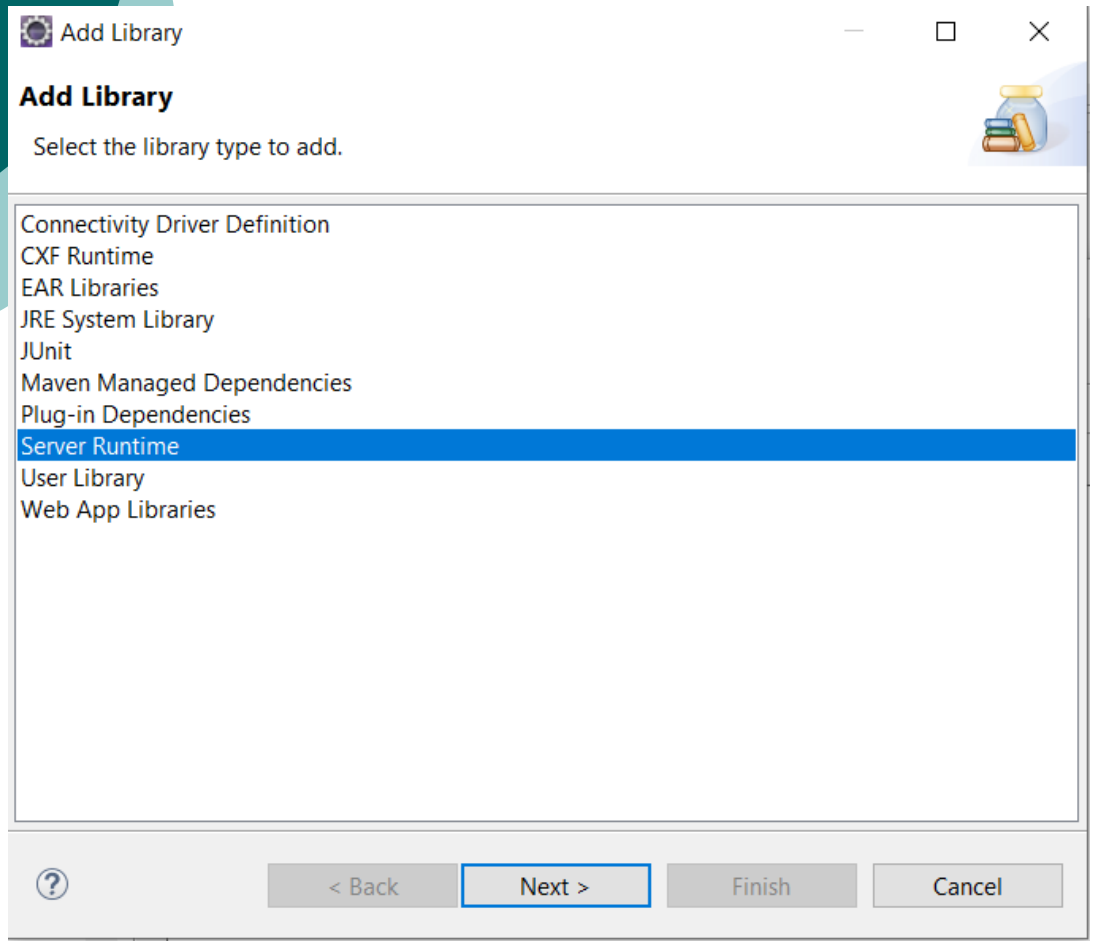


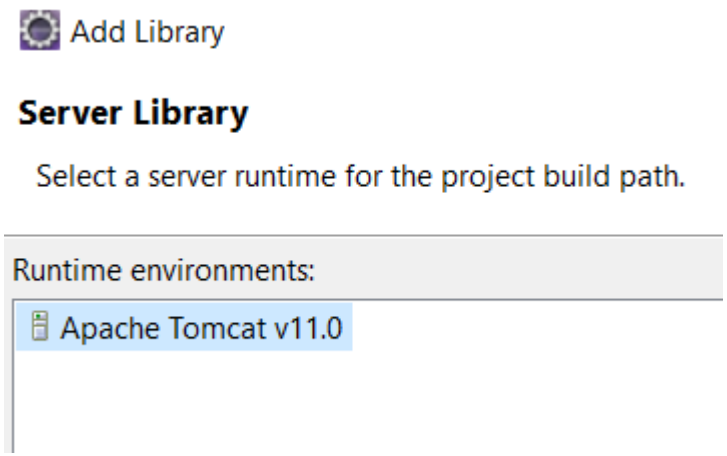


Create new project in Eclipse

Servlets

- **Add server libary to the project (if it is necessary):**

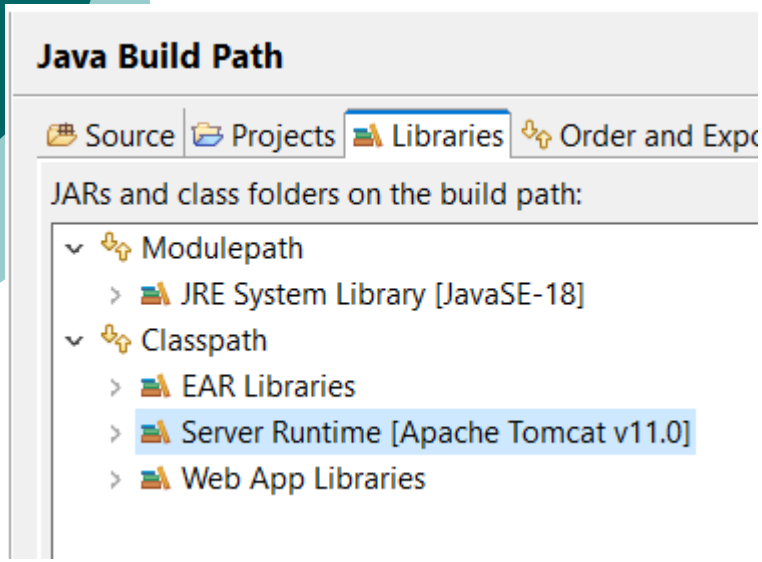


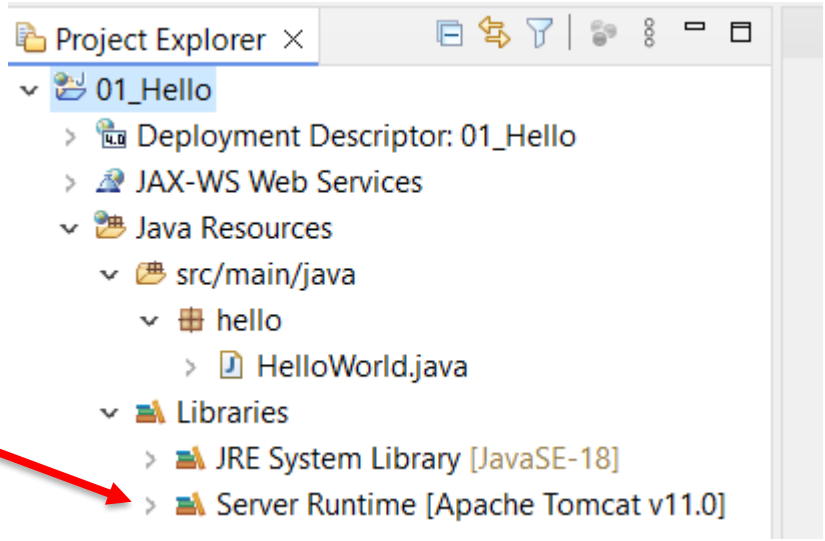


Create new project in Eclipse

Servlets

- **Add server libary to the project (if it is necessary):**





Server runtime library added successfully

Servlet deployment

- By default, a **servlet application** is located at the path **<Tomcat-installationdirectory>/webapps/ROOT** and the **class file** would reside in **<Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes**

- If you have a fully qualified class name of **com.myorg.MyServlet**, then this servlet class must be located in **WEB-INF/classes/com/myorg/MyServlet.class**

Servlet deployment

- For now, let us copy HelloWorld.class into <Tomcat-installationdirectory>/webapps/ROOT/WEB-INF/classes and create following entries in web.xml file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

```
<servlet>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>HelloWorld</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
```

- http://localhost:8080/HelloWorld

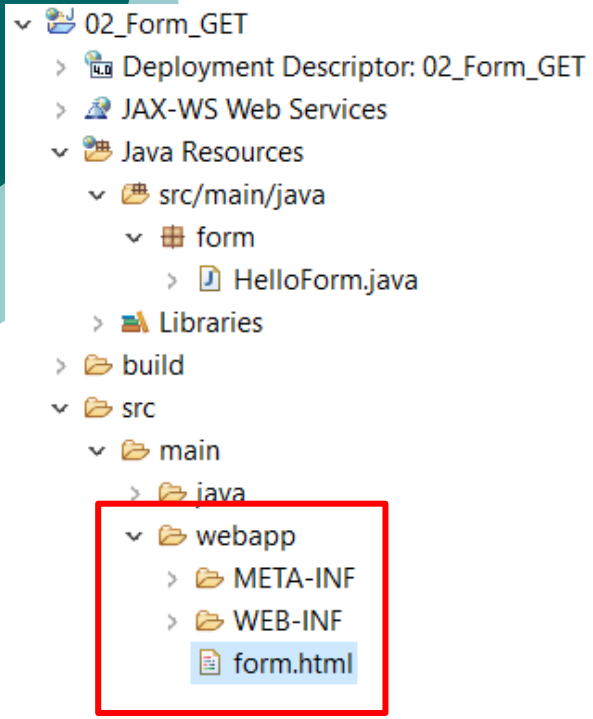
Task2: Form data - GET

- Source code: https://www.tutorialspoint.com/servlets/servlets-form-data.htm
- http://localhost:8080/HelloForm?first_name=ZARA&last_name=ALI

Task2: Form data - GET

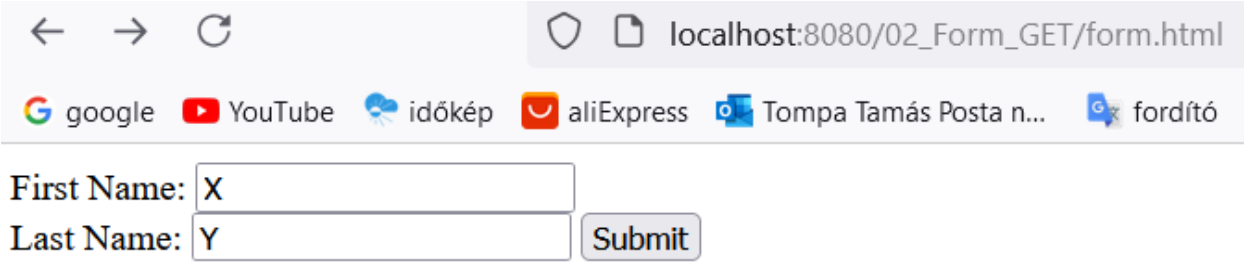
Servlets

- **Add an HTLM Form:**




```
<html>
<body>
    <form action = "HelloForm" method = "GET">
        First Name: <input type = "text" name = "first_name">
        <br />
        Last Name: <input type = "text" name = "last_name" />
        <input type = "submit" value = "Submit" />
    </form>
</body>
</html
```

URL: http://localhost:8080/02_Form_GET/form.html



Task2: Form data - POST

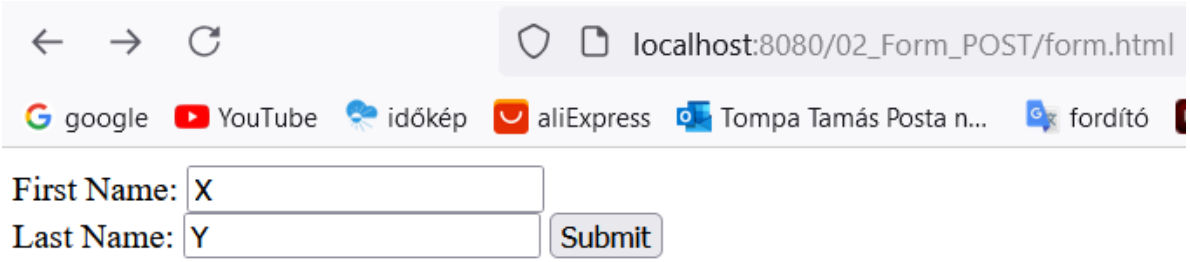
- Source code: https://www.tutorialspoint.com/servlets/servlets-form-data.htm

- Form code:

```
<html>
<body>
    <form action = "HelloForm" method = "POST">
        First Name: <input type = "text" name = "first_name">
        <br />
        Last Name: <input type = "text" name = "last_name" />
        <input type = "submit" value = "Submit" />
    </form>
</body>
</html>
```

Task2: Form data - POST

- Source code: https://www.tutorialspoint.com/servlets/servlets-form-data.htm
- URL: http://localhost:8080/02_Form_POST/form.html



Task3: Checkbox

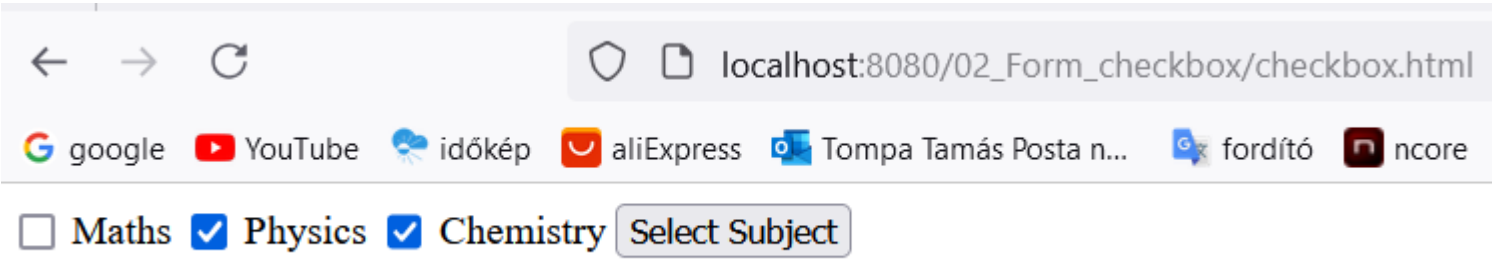
- Source code: https://www.tutorialspoint.com/servlets/servlets-form-data.htm
- URL: http://localhost:8080/02_Form_checkbox/checkbox.html

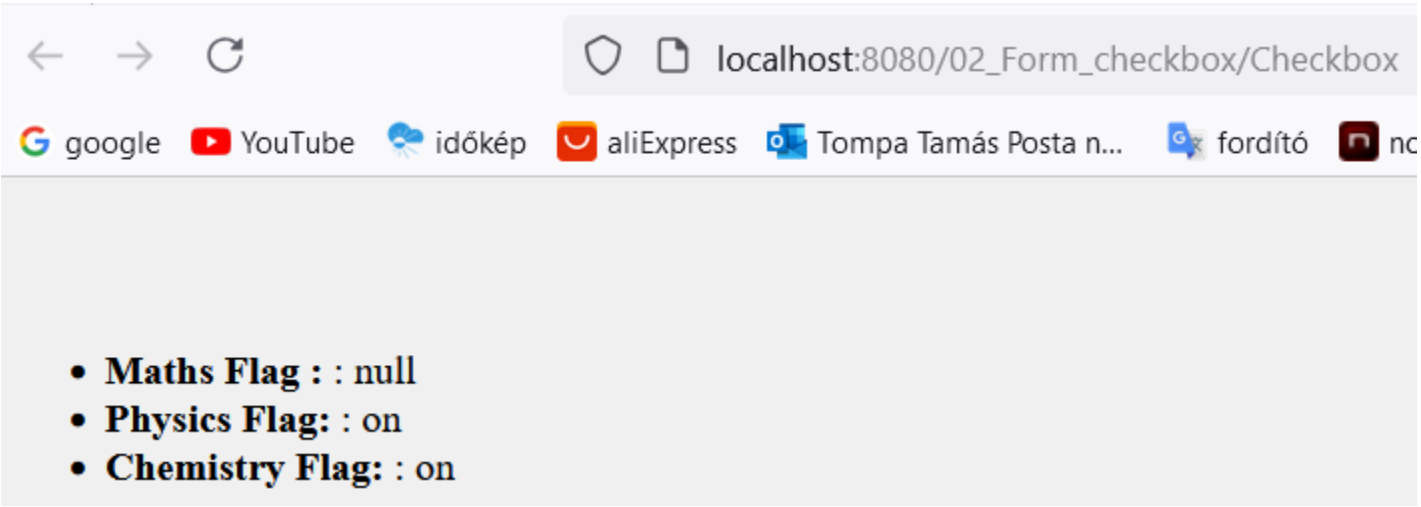
- Form code:

```
<html>
<body>
    <form action = "Checkbox" method = "POST" target = "_blank">
       <input type = "checkbox" name = "maths" checked = "checked" /> Maths
       <input type = "checkbox" name = "physics" /> Physics
       <input type = "checkbox" name = "chemistry" checked = "checked" /> Chemistry
       <input type = "submit" value = "Select Subject" />
    </form>
</body>
</html>
```

Task3: Checkbox

- Source code: https://www.tutorialspoint.com/servlets/servlets-form-data.htm
- URL: http://localhost:8080/02_Form_checkbox/checkbox.html

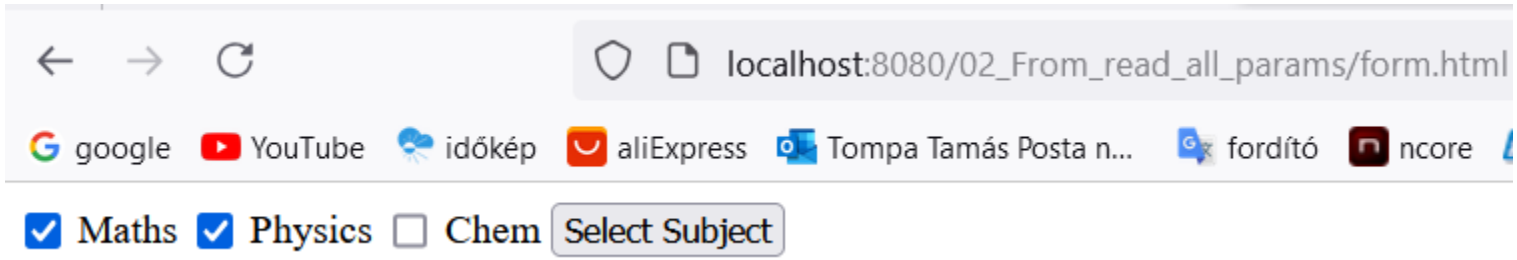


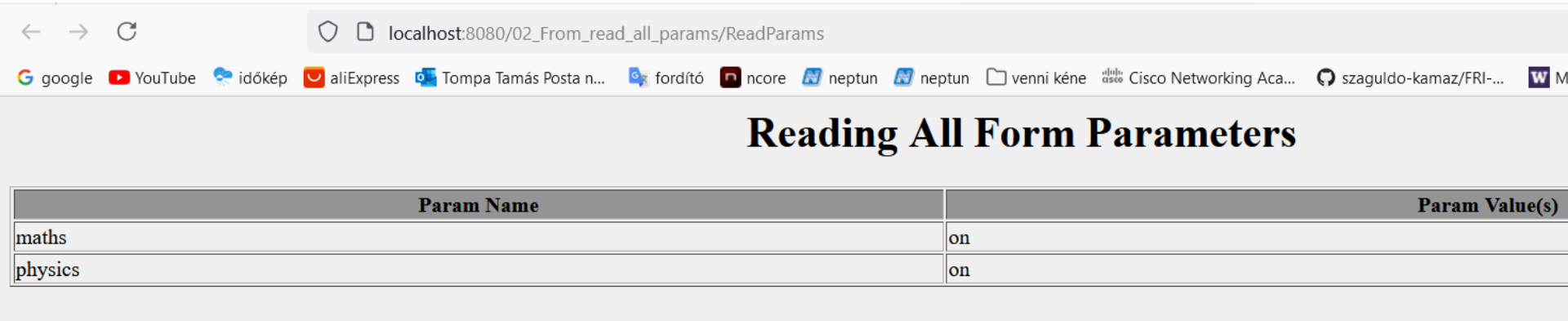


Task4: Read all of form params

Servlets

- Source code: https://www.tutorialspoint.com/servlets/servlets-form-data.htm
- URL: http://localhost:8080/02_From_read_all_params/form.html





Param Name	Param Value(s)
maths	on
physics	on

Task5: HTTP Header Request

- Source code: https://www.tutorialspoint.com/servlets/servlets-client-request.htm
- **`getHeaderNames()`**
  - **HttpServletRequest to read the HTTP header information**
  - **returns an Enumeration** that contains the header information associated with the current HTTP request

  - **HTTP header is used to pass additional information between the client (such as a browser) and the server during an HTTP request or response**
    - **It contains metadata about the communication:**
      - Content-Type: Specifies the media type of the resource (e.g., text/html, application/json)
      - Content-Length: Indicates the size of the resource in bytes
      - User-Agent: Identifies the client making the request
      - Authorization: Contains credentials for authenticating the client
      - Cache-Control: Defines caching policies for the response

Task5: HTTP Header Request

- Source code: https://www.tutorialspoint.com/servlets/servlets-client-request.htm
- **`getHeaderNames()`**

HTTP Header Request Example

Header Name	Header Value(s)
host	localhost:8080
user-agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
accept-language	hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
accept-encoding	gzip, deflate, br, zstd
connection	keep-alive
upgrade-insecure-requests	1
sec-fetch-dest	document
sec-fetch-mode	navigate
sec-fetch-site	none
sec-fetch-user	?1
priority	u=0, i

Task6: Server HTTP Response

- Source code: https://www.tutorialspoint.com/servlets/servlets-server-response.htm
- server responds to an HTTP request, the response typically consists of a status line, some response headers, a blank line, and the document

- **`setContentType()`**
  - This method is used to **set the MIME** (Multipurpose Internet Mail Extensions) type of the HTTP response
  - It **informs the client** (like a browser) **about the type of data** it will receive, such as text/html, application/json, or text/plain
  - `response.setContentType("text/html");`

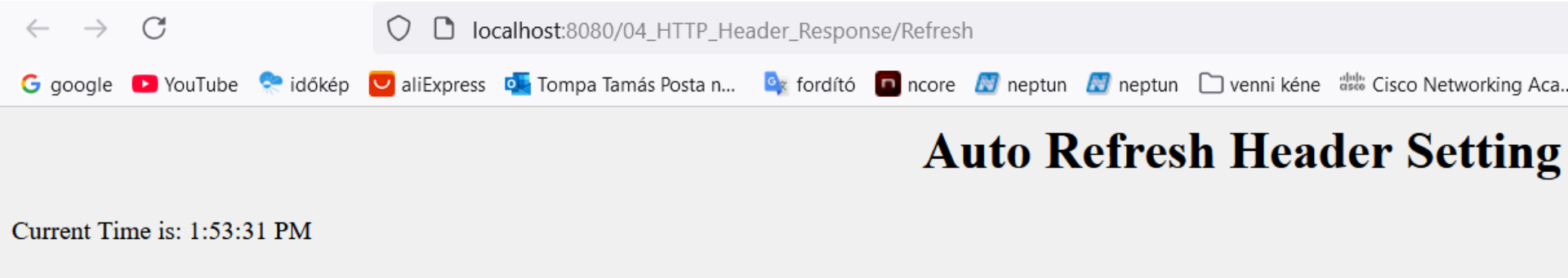
- **`setIntHeader()`**
  - This method sets an **HTTP response header with an integer value**
  - It's **used to set or overwrite specific headers**, such as setting a status code or defining a header like Content-Length
  - `response.setIntHeader("Content-Length", 1024);`

Task6: Server HTTP Response

Servlets

- Source code: https://www.tutorialspoint.com/servlets/servlets-server-response.htm
- server responds to an HTTP request, the response typically consists of a status line, some response headers, a blank line, and the document

```
// Set refresh, autoload time as 5 seconds
response.setIntHeader("Refresh", 5);
```



Task7: Http Status Codes

- Source code: https://www.tutorialspoint.com/servlets/servlets-http-status-codes.htm
- Methods to Set HTTP Status Code
  - **`public void setStatus ( int statusCode )`**
    - sets an arbitrary status code. The setStatus method takes an int (the status code) as an argument. If your response includes a special status code and a document, be sure to call setStatus before actually returning any of the content with the PrintWriter
  - **`public void sendRedirect(String url)`**
    - generates a 302 response along with a Location header giving the URL of the new document
  - **`public void sendError(int code, String message)`**
    - sends a status code (usually 404) along with a short message that is automatically formatted inside an HTML document and sent to the client.

Task7: Http Status Codes

- Source code: https://www.tutorialspoint.com/servlets/servlets-http-status-codes.htm

```
// Set error code and reason.
response.sendError(407, "Need authentication!!!");
```

HTTP Status 407 – Proxy Authentication Required

Type Status Report

Message Need authentication!!!

Description This status code is similar to 401 (Unauthorized), but it indicates that the client needs to authenticate itself in order to use a proxy.

Apache Tomcat/11.0.0

Task8: Writing filters

- **Servlet Filters are Java classes**
  - **To intercept requests from a client before they access** a resource at back end
  - **To manipulate responses from server before they are sent back** to the client
  - Types: Authentication Filters, Data compression Filters, Encryption Filters, Filters that trigger resource access events, Image Conversion Filters, etc.

- Servlet Filter Methods
  - `public void doFilter (ServletRequest, ServletResponse, FilterChain)`
  - `public void init(FilterConfig filterConfig)`
  - `public void destroy()`

Task8: Writing filters

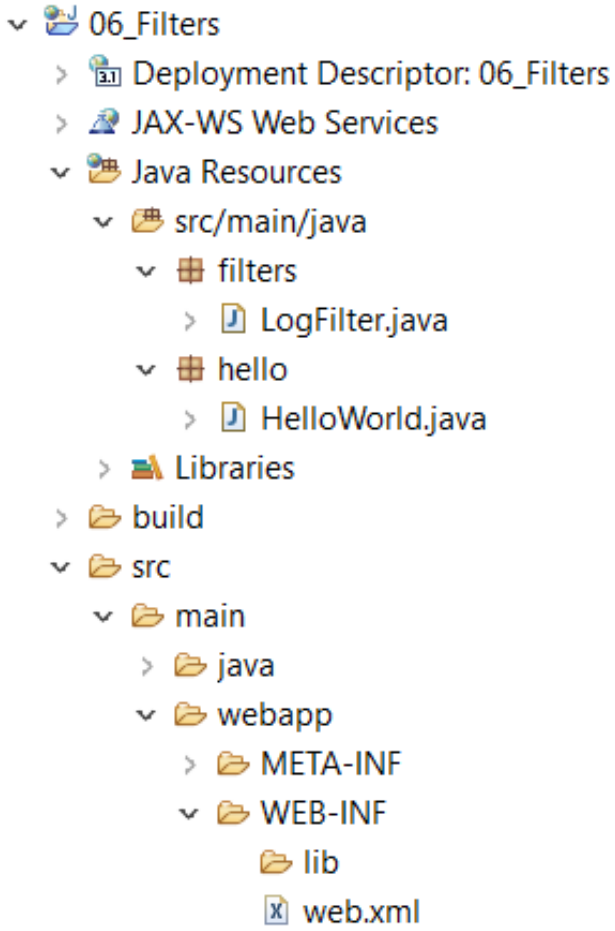
- **Servlet Filter Methods**
  - **`public void doFilter (ServletRequest, ServletResponse, FilterChain)`**
    - called by the container each time a request/response pair is passed through the chain due to a client request for a resource at the end of the chain
  - **`public void init(FilterConfig filterConfig)`**
    - called by the web container to indicate to a filter that it is being placed into service
  - **`public void destroy()`**
    - called by the web container to indicate to a filter that it is being taken out of service
  - **Filter mapping have to define in the `web.xml` file of the project**
    - automatically runs on every HTTP request when the specified URL pattern (/*) is called

Task8: Writing filters

- Source code: https://www.tutorialspoint.com/servlets/servlets-writing-filters.htm
- Copy the HelloWorld class from the 01_Hello

Servlet Filter Mapping in the web.xml:

```
<filter>
<filter-name>LogFilter</filter-name>
<filter-class>filters.LogFilter</filter-class>
<init-param>
<param-name>test-param</param-name>
<param-value>Initialization Parameter</param-value>
</init-param>
</filter>


<filter-mapping>
<filter-name>LogFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```



Task9: Exception Handling

- Source code: https://www.tutorialspoint.com/servlets/servlets-exception-handling.htm
- When a servlet throws an exception, the web container searches the configurations in web.xml
- You would have to use the error-page element in web.xml to specify the invocation of servlets in response to certain exceptions or HTTP status codes
  - The servlet ErrorHandler is defined in usual way as any other servlet and configured in web.xml
  - If there is any error with status code either 404 (Not Found) or 403 (Forbidden), then ErrorHandler servlet would be called
  - If the web application throws either ServletException or IOException, then the web container invokes the /ErrorHandler servlet
  - You can define different Error Handlers to handle different type of errors or exceptions. Above example is very much generic and hope it serve the purpose to explain you the basic concept

Task10: Cookies Handling

Servlets

- Source code: https://www.tutorialspoint.com/servlets/servlets-cookies-handling.htm
- **Cookies are text files stored on the client computer** and they are kept for various information tracking purpose
- **Small piece of data that a server sends to a user's web browser and is stored on the user's device**

- **There are three steps involved in identifying returning users:**
  - **server script sends a set of cookies to the browser**. For example name, age, or identification number etc.
  - **browser stores this information on local machine for future use**
  - **when next time browser sends any request to web server then it sends those cookies information to the server** and server uses that information to identify the user

- Cookies are used to store information such as user preferences, session data, or tracking information, enabling a website to remember the user across visits or track behavior for personalized experiences

Task10: Cookies Handling -setting

Servlets

- Cookies are usually set in an HTTP header

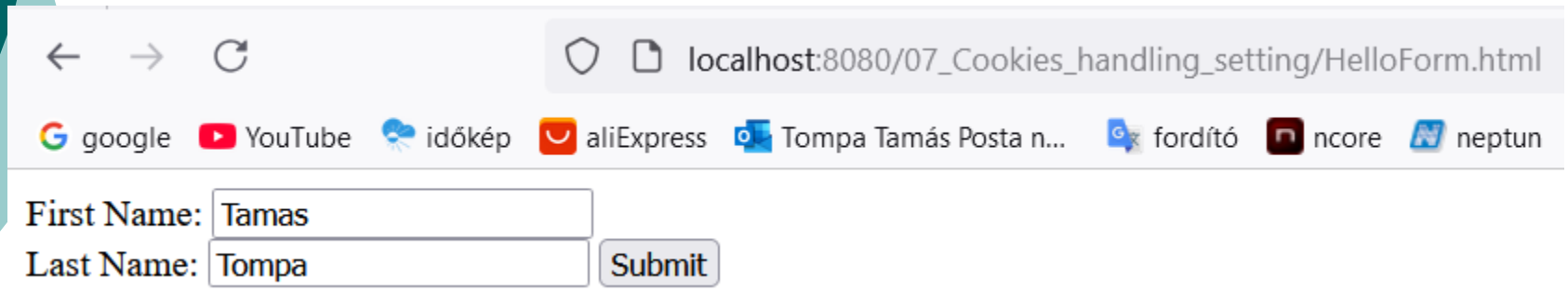
```
HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name = xyz; expires = Friday, 04-Feb-07 22:03:38
GMT;
path = /; domain = tutorialspoint.com
Connection: close
Content-Type: text/html
```

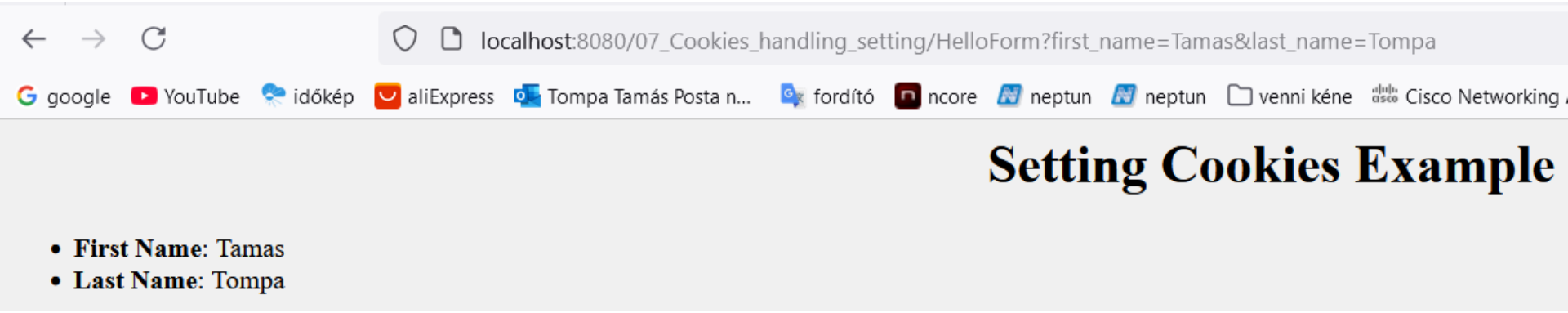
- Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded. The expires field is an instruction to the browser to "forget" the cookie after the given time and date
- If the browser is configured to store cookies, it will then keep this information until the expiry date

```
Cookie cookie = new Cookie("key","value");
```

Task10: Cookies Handling -setting

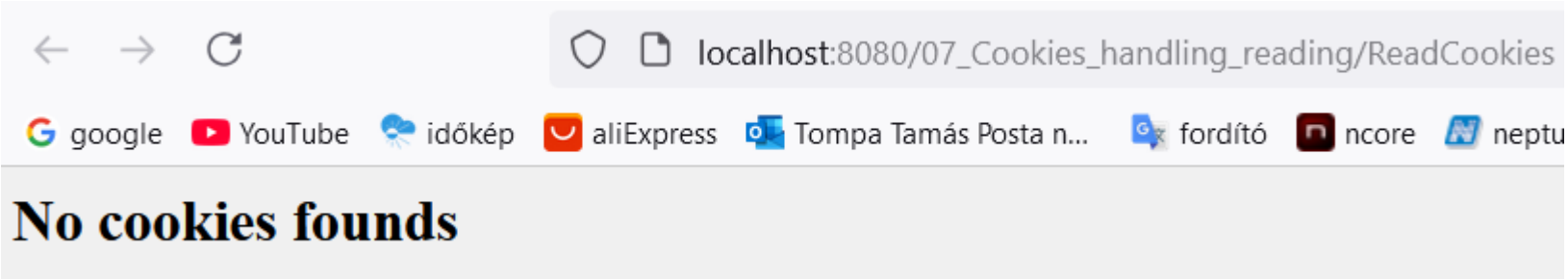
Servlets





Task10: Cookies Handling -reading

- Source code: https://www.tutorialspoint.com/servlets/servlets-cookies-handling.htm



Task10: Cookies Handling -reading

Servlets

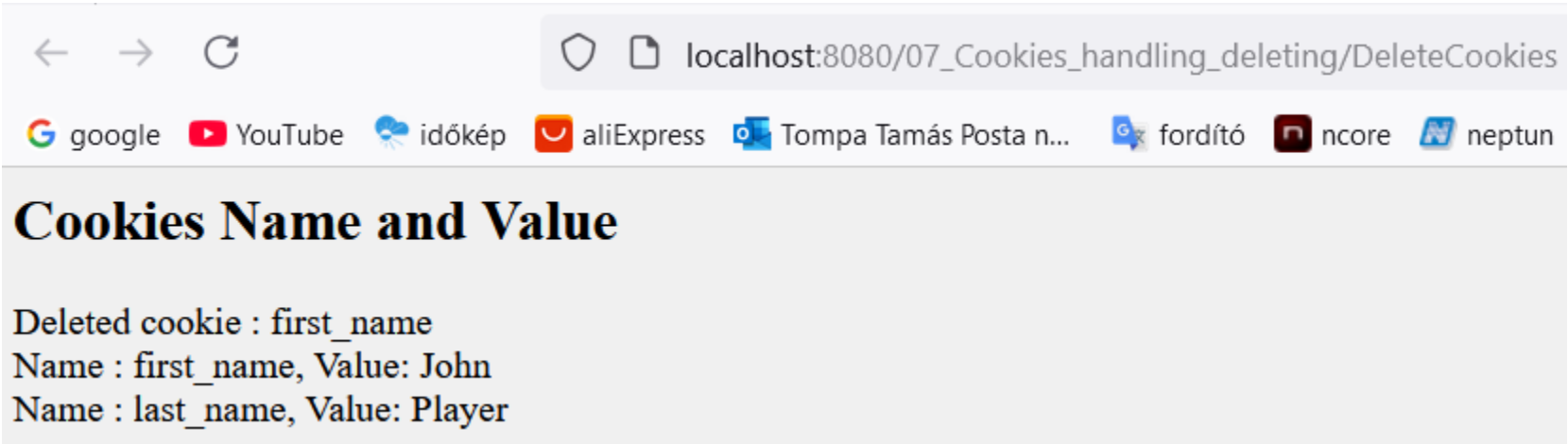
- Source code: https://www.tutorialspoint.com/servlets/servlets-cookies-handling.htm

```
// Create cookies
Cookie firstName = new Cookie("first_name", "John");
Cookie lastName = new Cookie("last_name", "Player");

// Set cookies to expire in 24 hours
firstName.setMaxAge(60 * 60 * 24);
lastName.setMaxAge(60 * 60 * 24);

// Add cookies to response
response.addCookie(firstName);
response.addCookie(lastName);
```

Task10: Cookies Handling -delete

- Source code: https://www.tutorialspoint.com/servlets/servlets-cookies-handling.htm

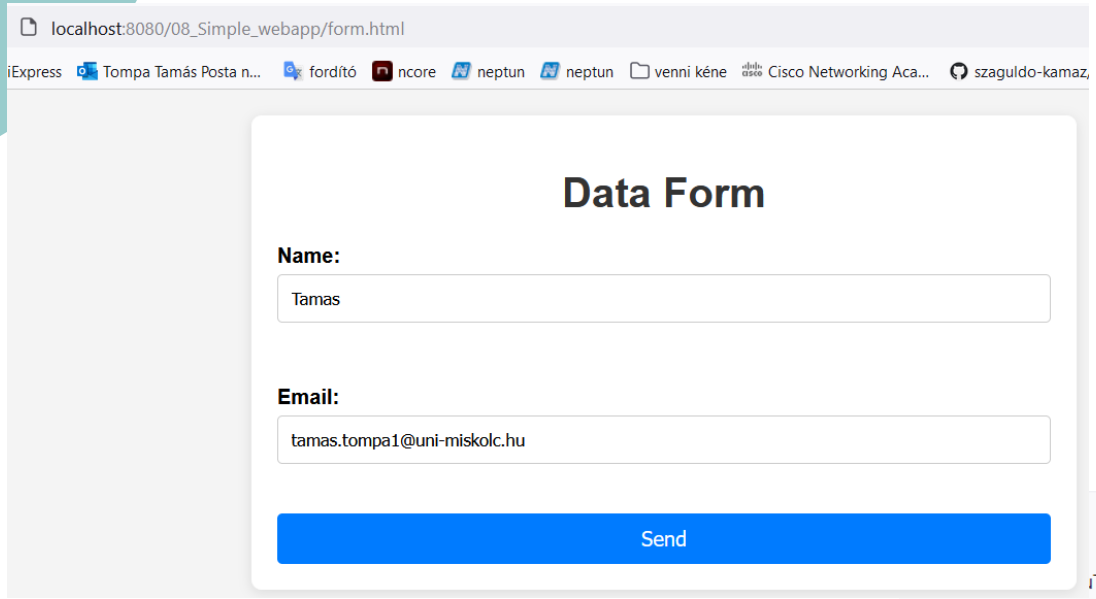
- **Read an already existing cookie** and store it in Cookie object
  - `cookies = request.getCookies();`
- **Set cookie age as zero** using setMaxAge() method to delete an existing cookie
  - `cookie.setMaxAge(0);`
- **Add** this cookie back **into response header**
  - `response.addCookie(cookie);`

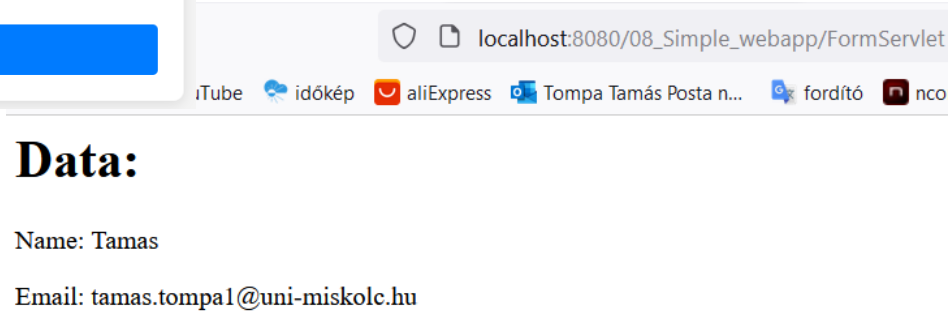


Task11: Create a simple webapp

Servlets

- Create a simple web application that contains
  - a form where the user can enter their name and email address
  - the servlet processes the form and displays the entered data on a new page





Thank you for your attention!