Course: Web Application Development
Lab 3 - Introduction to Servlet Programming

Content:
- Introduction to Java Servlet Technology
- How to create a servlet and run on Eclipse IDE
- Practices and Exercises

Duration: 3 hours+

Part 1: Introduction to Servlet and Servlet Life Cycle.

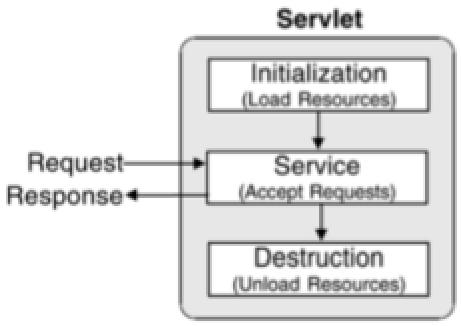
- **Recall: What is Servlet?**
    - A *servlet* is a Java programming language class that is used to extend the capabilities of servers that host applications access via a request-response programming model.
    - Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.
- **How Java support Servlet Programming**
    - The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines servlet life-cycle methods.
    - The **HttpServlet** class provides methods, such as **doGet** and **doPost**, for handling HTTP-specific services.
    - When implementing a generic service, you can use or extend the **GenericServlet** class provided with the Java Servlet API.
- **Tomcat Servlet/JSP container**
    - ***Tomcat*** can act as a stand-alone Web server and also as a servlet/JSP engine for other Web servers. When you download the Tomcat server, you really get a number of packages. **Catalina** and **Jasper** are the names of the servlet and JSP containers**.**
    - Tomcat by itself is a web server. This means that you can use Tomcat to service HTTP requests for servlets, as well as static files (HTML, image files, and so on). In practice, however, since it is faster for non-servlet, non-JSP requests, Tomcat normally is used as a module with another more robust web server, such as Apache web server or Microsoft Internet Information Server (IIS).
    - Tomcat is not a J2EE application server. However, as J2EE app servers must themselves contain a servlet container to support the servlet/JSP APIs, J2EE

app servers can embed Tomcat into their code to provide support for the Servlet and JSP APIs. One example of just such an application server is the popular open source JBoss J2EE app server (http://www.jboss.org/).

- **Servlet Life Cycle**
  - The javax.servlet.Servlet interface defines the methods that all servlets must implement and, among others, three methods that are known as **life-cycle methods**:

```
public void init(ServletConfig config) throws ServletException

public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException

public void destroy()
```

  - These life-cycle methods are each called at separate times during the life span of a servlet, from the initial creation to the moment it's removed from service and destroyed. These methods are called in the following order:



1. When the servlet is constructed, it is initialized with the init() method.

2. Any requests from clients are handled initially by the service() method before delegating to the doXxx() methods in the case of an HttpServlet. The service() method is responsible for processing the request and returning the response.

3. When the servlet needs to be removed from service, it's destroyed with the destroy() method, then garbage collected and finalized. When the container decides to take a servlet out of service, it first ensures that any service() method calls have been completed.

  - The init method is called by the servlet container after the servlet class has been instantiated. The servlet container calls this method exactly once to indicate to the servlet that the servlet is being placed into service. The init method is important also because the servlet container passes a *ServletConfig* object, which contains the configuration values stated in the web.xml file for this application.
  - The service method is called by the servlet container after the servlet's init method to allow the servlet to respond to a request. The servlet container passes a ServletRequest object and the ServletResponse object. The

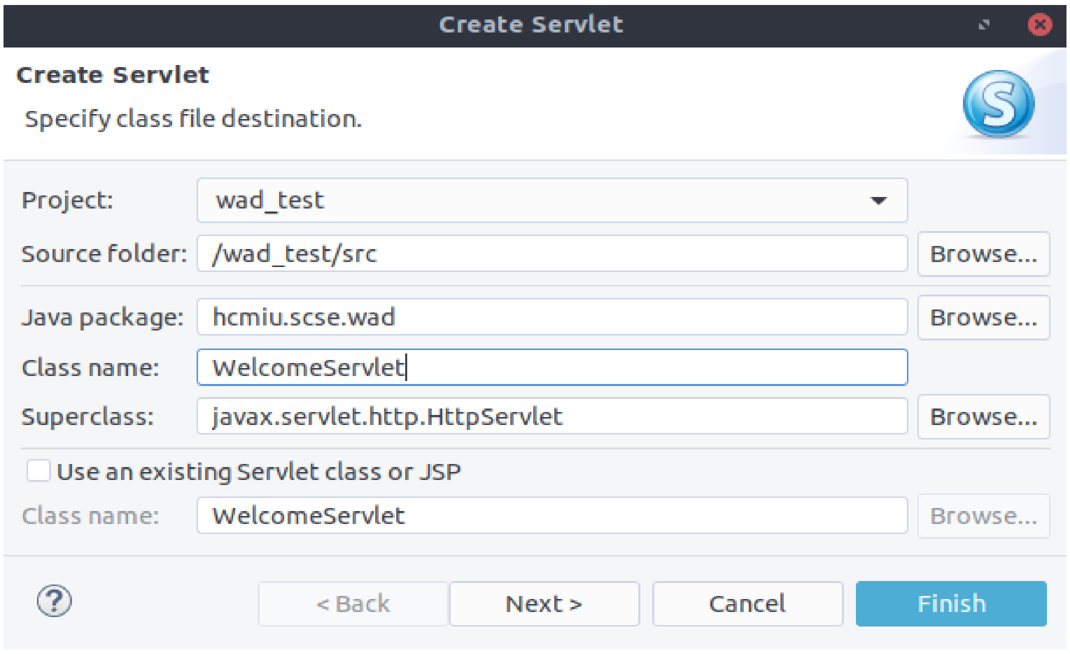
ServletRequest object contains the client's request and the ServletResponse contains the servlet's response.

- The servlet container calls the destroy method before removing a servlet instance from service. This normally happens when the servlet container is shut down or the servlet container needs some free memory.

*(More information: refer from textbook: **Core Servlets and Java Server Pages**)*

Part 2: How to create a servlet and run on Eclipse IDE

- Create new **Dynamic Web Project** (if you don't have one).
- Right click on your project (on **Project Explorer** panel), **New** → **Servlet**


Create Servlet

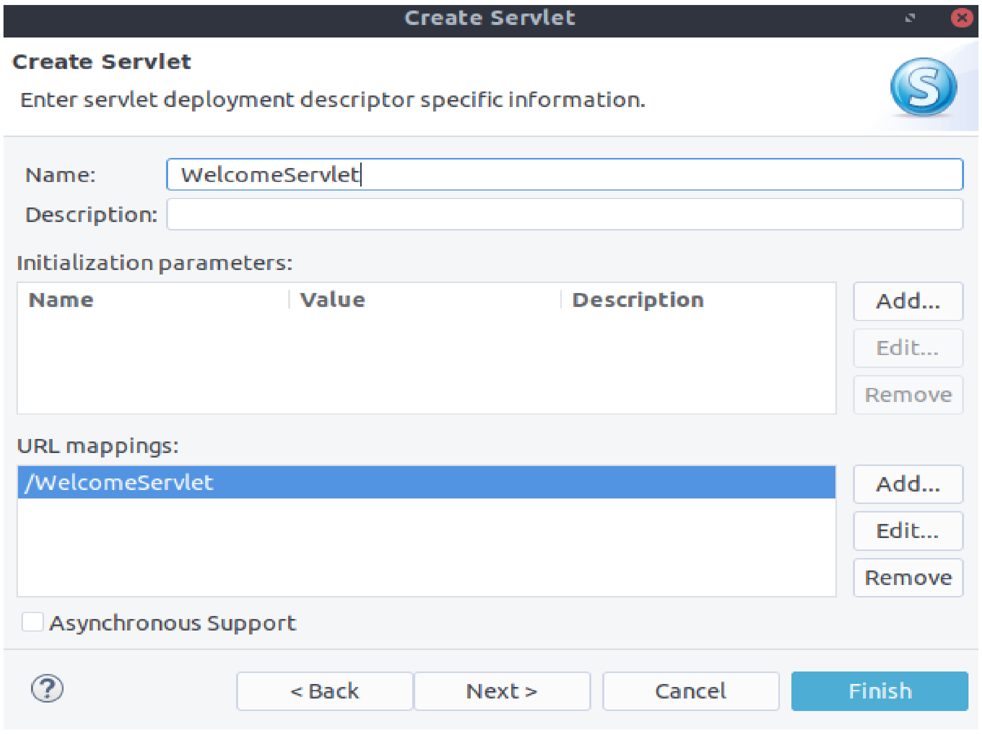
Create Servlet
Specify class file destination.

Project: wad_test
Source folder: /wad_test/src
Java package: hcmiu.scse.wad
Class name: WelcomeServlet
Superclass: javax.servlet.http.HttpServlet
☐ Use an existing Servlet class or JSP
Class name: WelcomeServlet

****Note:**
- **Project:** name of your created project.
- **Source folder:** the root folder where new Servlet file will be saved to. Default is the **src** folder in your project.
- **Java package:** the sub folders/paths to your Servlet file. Mainly used for source code management. In this case, I use *hcmiu.scse.wad* for my Java package (my new Servlet file will be saved to *wad_test/src/hcmiu/scse/wad/*)
- **Class name:** The name of your Servlet
- **Supperclass:** The parent class of your Servlet, It should be *javax.servlet.http.HttpServlet* by default.

- At this point, you can choose **Finish** to create your new Servlet or **Next** for advanced options.

- If you select **Next**, you will be asked to enter servlet deployment descriptor specific information.



Create Servlet

Enter servlet deployment descriptor specific information.

Name: WelcomeServlet

Description:

Initialization parameters:

Name	Value	Description

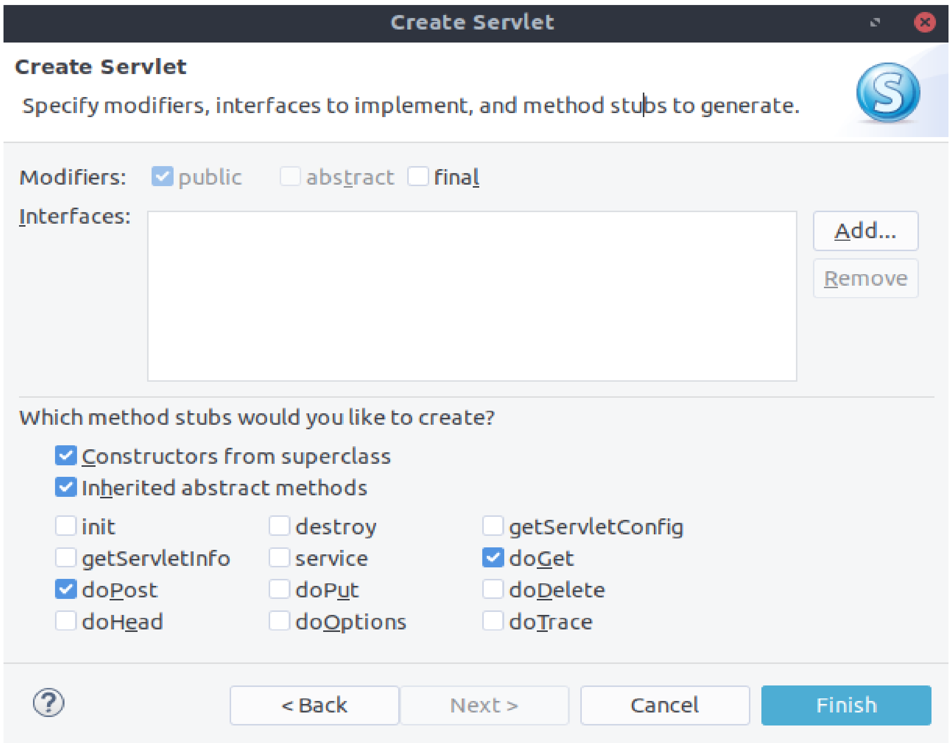
URL mappings:

/WelcomeServlet

☐ Asynchronous Support

**Note:
- **Description**: you may provide a short description for your Servlet.
- **Initialization Parameters:** Some parameters used in your Servlet.
- **URL mappings:** the url point to Servlet. The default value is the name of your Servlet. The complete URL will be *http://localhost:{port}/{NameOfYourProject}/{YourServletName}* .
Example: http://localhost:8080/wad_test/WelcomeServlet

- Choose **Finish** to create new Servlet or **Next** for other options.
- If you select **Next**, you will be asked to specify modifiers, interfaces, and method stubs.



Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: ☑ public ☐ abstract ☐ final

Interfaces:

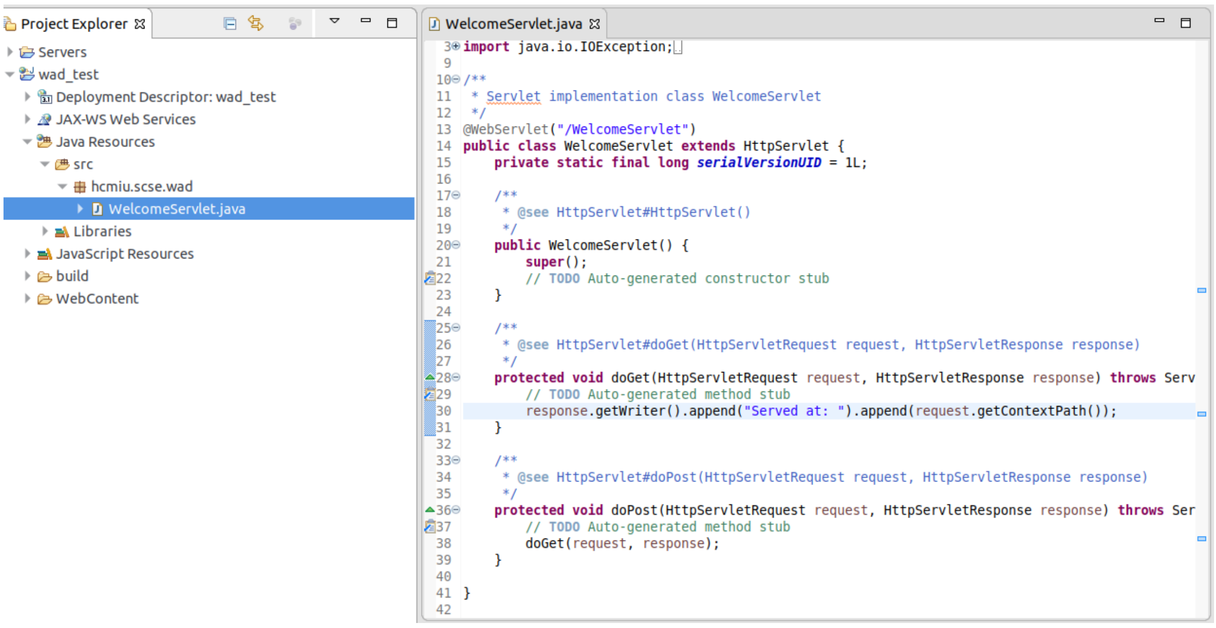
Which method stubs would you like to create?

☑ Constructors from superclass
☑ Inherited abstract methods

☐ init	☐ destroy	☐ getServletConfig
☐ getServletInfo	☐ service	☑ doGet
☑ doPost	☐ doPut	☐ doDelete
☐ doHead	☐ doOptions	☐ doTrace

**Note:
- **Interfaces:** define the interface you want your new Servlet implement.
- **Method Stubs:** selecting the functions you want your Servlet override from parent classes. By default, *doPost* and *doGet* are selected. In some cases (especially your course project), you may need other methods (such as *init, destroy, doPut, doDelete*) as well.

- Click **Finish** to complete all reconfiguration and create your new Servlet.

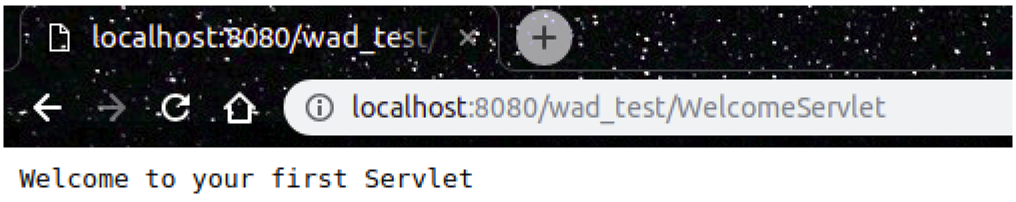


```
import java.io.IOException;

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

    /**
     * @see HttpServlet#HttpServlet()
     */
    public WelcomeServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Serv
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws Ser
        // TODO Auto-generated method stub
        doGet(request, response);
    }

}
```

- Try to modify your **doGet** function to print out something. For example:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.getWriter().println("Welcome to your first Servlet");
}
```

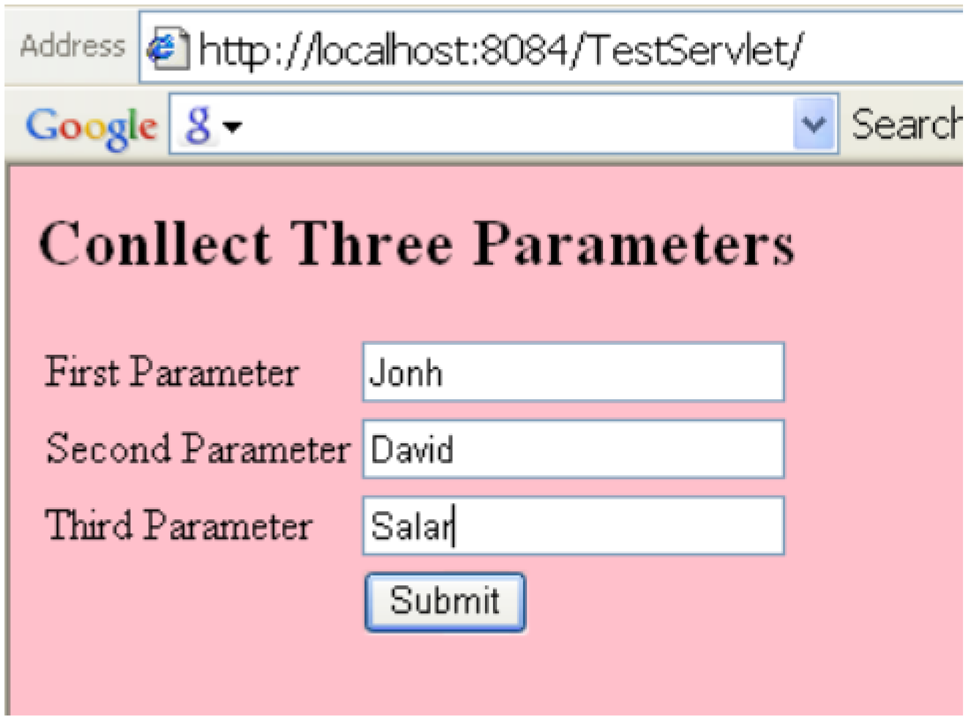
- Run your project and access to the Servlet (by URL) to see the result:



```
Welcome to your first Servlet
```

Part 3: Practices and Exercises.

Exercise1: design a form (ThreeParams.JSP) as below:

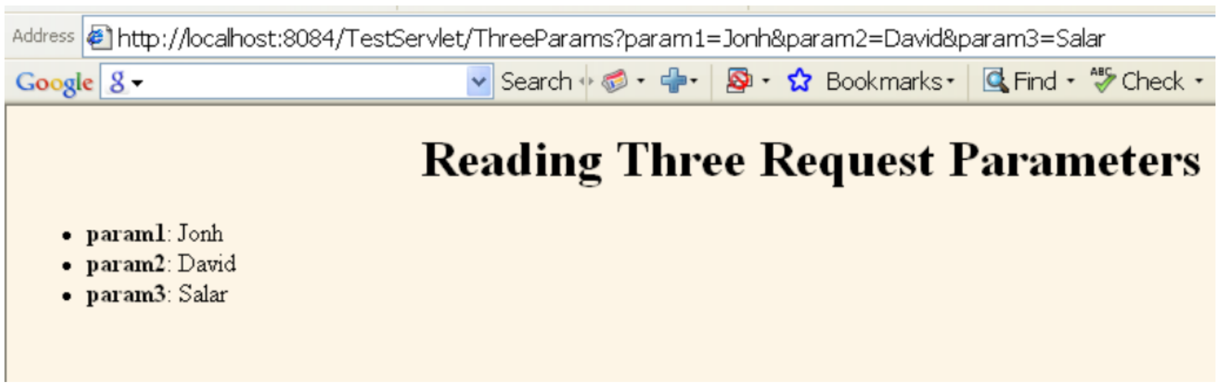


Address http://localhost:8084/TestServlet/

Conllect Three Parameters

First Parameter	Jonh
Second Parameter	David
Third Parameter	Salar

After inputting three values in the textbox -> click Submit button, it will call the servlet GetThreeParam.java

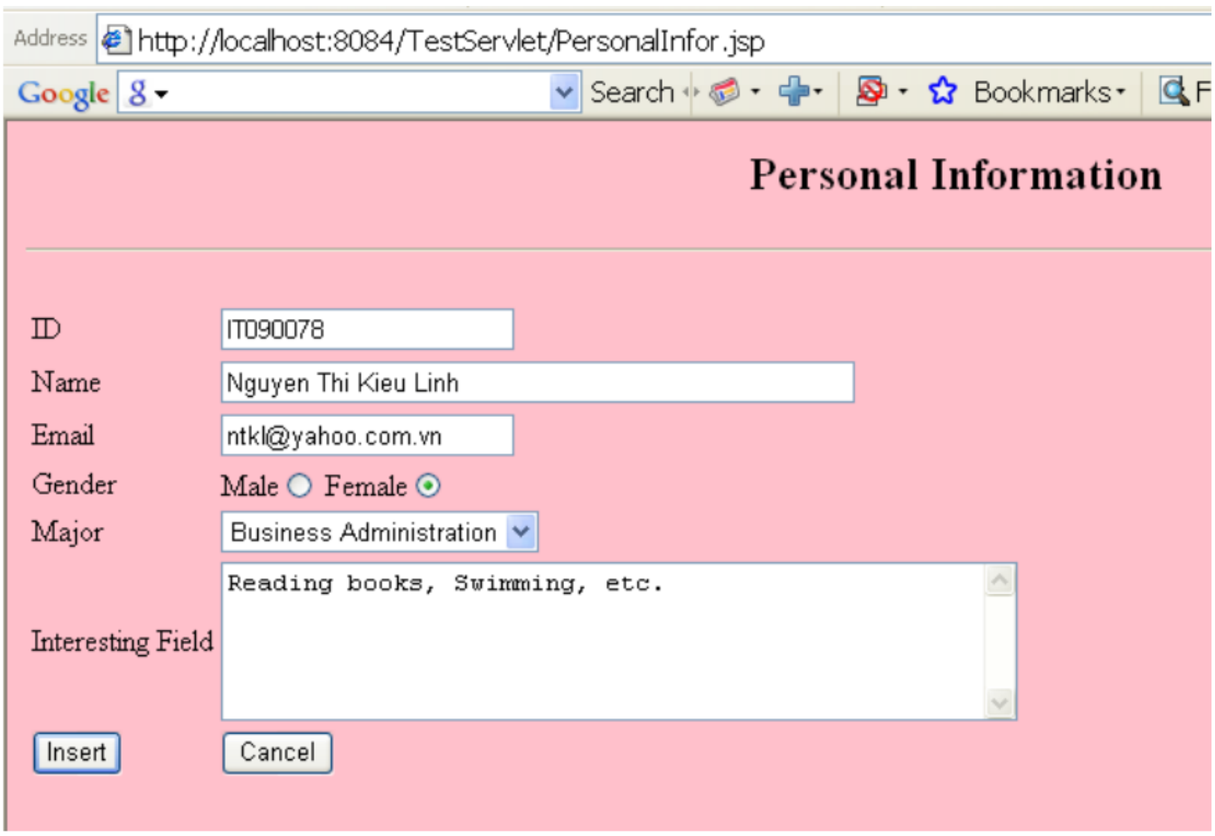


Address http://localhost:8084/TestServlet/ThreeParams?param1=Jonh&param2=David&param3=Salar

Reading Three Request Parameters

- **param1**: Jonh
- **param2**: David
- **param3**: Salar

Exercise 2: Design form Personal Information (PersonalInfor.jsp)

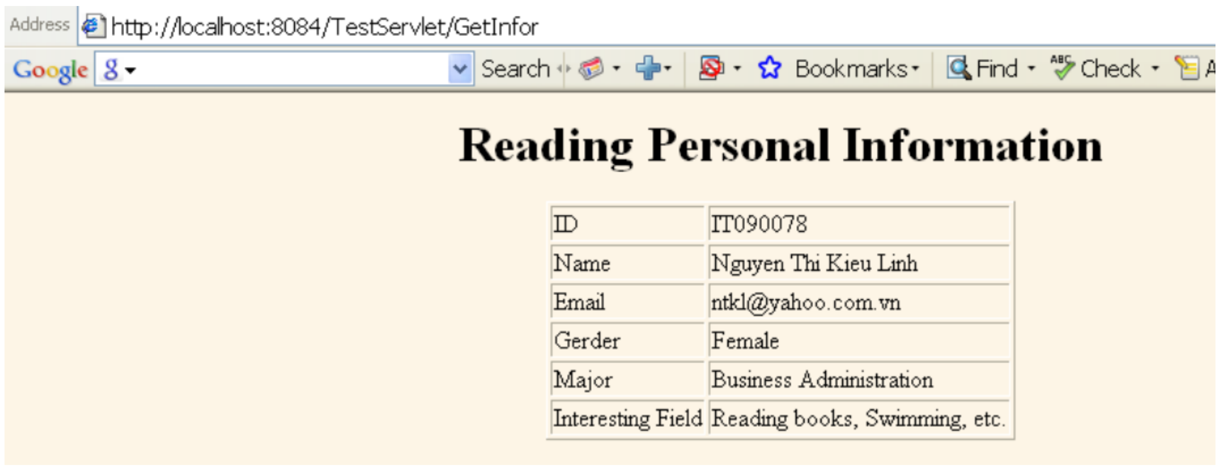


http://localhost:8084/TestServlet/PersonalInfor.jsp

Personal Information

ID	IT090078
Name	Nguyen Thi Kieu Linh
Email	ntkl@yahoo.com.vn
Gender	Male ○ [✓ Female]
Major	Business Administration
Interesting Field	Reading books, Swimming, etc.

The result from GetInfor.java

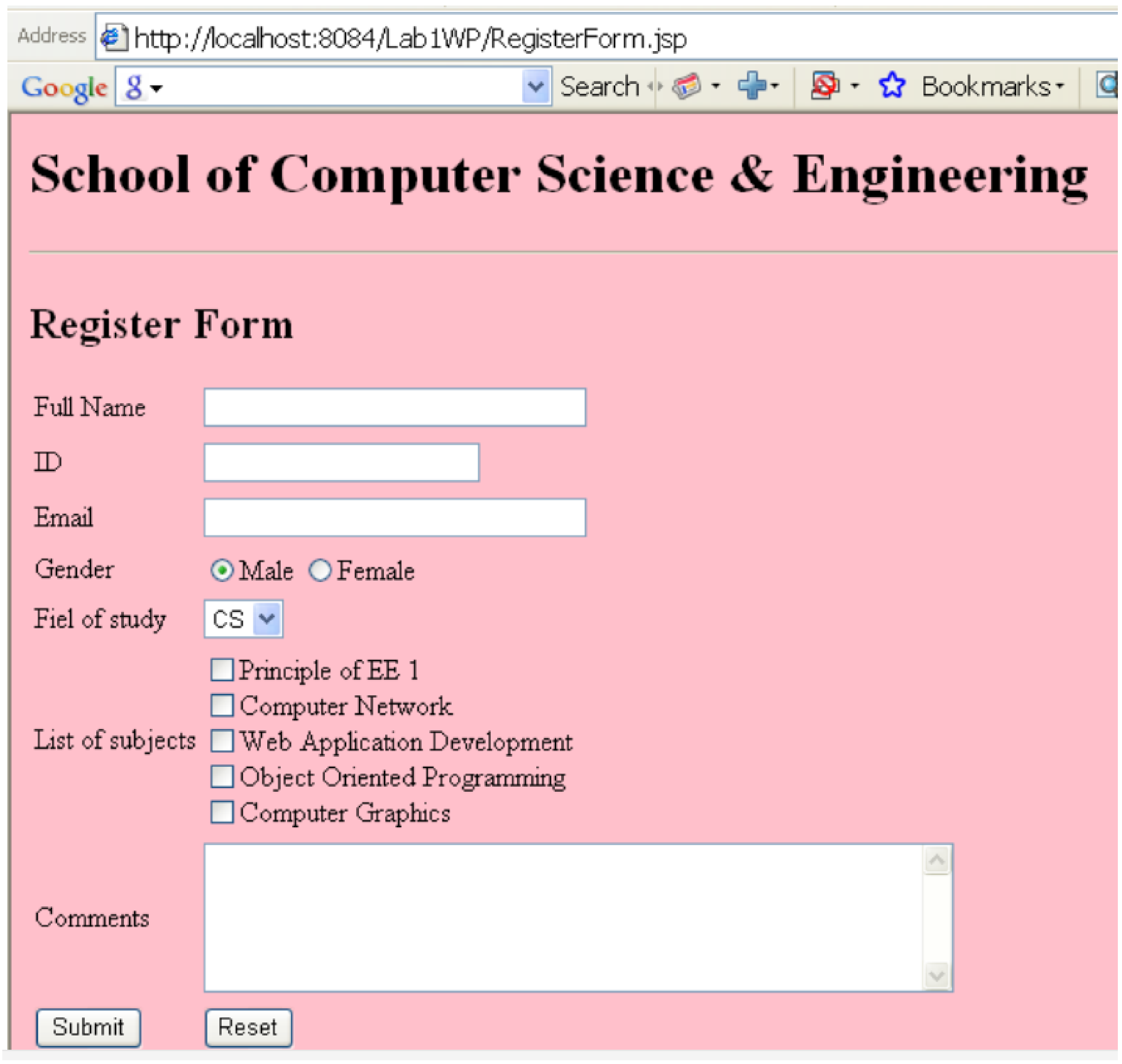


http://localhost:8084/TestServlet/GetInfor

Reading Personal Information

ID	IT090078
Name	Nguyen Thi Kieu Linh
Email	ntkl@yahoo.com.vn
Gerder	Female
Major	Business Administration
Interesting Field	Reading books, Swimming, etc.

Exercise 3: Design a register form, use Servlet to get all information when the user clicks the Submit button, and put it into a new page.



Address http://localhost:8084/Lab1WP/RegisterForm.jsp

School of Computer Science & Engineering

Register Form

Full Name	
ID	
Email	
Gender	[✓ Male] Female
Fiel of study	CS
List of subjects	☐ Principle of EE 1 ☐ Computer Network ☐ Web Application Development ☐ Object Oriented Programming ☐ Computer Graphics
Comments	
Submit	Reset

The result show parameter Servlet (RegisterCourse.java)