Lab Exercise 2- CRUD API Example in Java (Servlets) using Tomcat

Objective:

CRUD API Example in Java (Servlets) using **Eclipse + Tomcat**.

This is the **simplest possible implementation** without using Spring Boot — perfect for labs, interviews, or basic hands-on API training.

We will implement CRUD for a resource:

Resource Name: User

Each user has:

- id
- name
- email

All data will be stored **in-memory (HashMap)** for simplicity.

Project Structure

```
SimpleCRUDAPI/
 └── src/
      └── com.api/
            └── UserServlet.java
 └── WebContent/
      └── WEB-INF/
            └── web.xml
```

1. Create the User Servlet (CRUD API)

Name: **UserServlet**

Package: **com.api**

Replace code with the following:

UserServlet.java

```
package com.api;

import java.io.BufferedReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/users")
public class UserServlet extends HttpServlet {

   private static final long serialVersionUID = 1L;

   // Simple in-memory storage
   private Map<Integer, String> users = new HashMap<>();

   // Utility: read full JSON body from request
   private String readBody(HttpServletRequest request) throws IOException {
      BufferedReader br = request.getReader();
```

```
    StringBuilder sb = new StringBuilder();
    String line;

    while ((line = br.readLine()) != null) {
        sb.append(line);
    }

    return sb.toString();
}

// Utility: simple JSON parser (string-based)
private Map<String, String> parseJson(String json) {
    Map<String, String> map = new HashMap<>();

    json = json.trim();

    // Remove { }
    json = json.substring(1, json.length() - 1).trim();

    // Split key-value pairs
    String[] pairs = json.split(",");

    for (String pair : pairs) {
        String[] keyValue = pair.split(":");

        if (keyValue.length == 2) {
            String key = keyValue[0].replace("\"", "").trim();
            String value = keyValue[1].replace("\"", "").trim();

            map.put(key, value);
        }
    }
```

```
        return map;
    }

    // CREATE (POST) - JSON
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String body = readBody(request);

        Map<String, String> json = parseJson(body);

        int id = Integer.parseInt(json.get("id"));
        String name = json.get("name");
        String email = json.get("email");

        String userJson = "{ \"id\": " + id + ", \"name\": \"" + name + "\", \"email\": \"" +
email + "\" }";

        users.put(id, userJson);

        response.setContentType("application/json");
        response.getWriter().write("{\"message\": \"User created successfully\"}");
    }

    // READ (GET)
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        int id = Integer.parseInt(request.getParameter("id"));
```

```
        response.setContentType("application/json");

        if (users.containsKey(id)) {
            response.getWriter().write(users.get(id));
        } else {
            response.getWriter().write("{\"error\": \"User not found\"}");
        }
    }

    // UPDATE (PUT) - JSON
    @Override
    protected void doPut(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        int id = Integer.parseInt(request.getParameter("id"));

        if (!users.containsKey(id)) {
            response.setContentType("application/json");
            response.getWriter().write("{\"error\": \"User not found\"}");
            return;
        }

        String body = readBody(request);

        Map<String, String> json = parseJson(body);

        String name = json.get("name");
        String email = json.get("email");

        String updatedJson = "{ \"id\": " + id + ", \"name\": \"" + name + "\", \"email\":
\"" + email + "\" }";
```

```
        users.put(id, updatedJson);

        response.setContentType("application/json");
        response.getWriter().write("{\"message\": \"User updated successfully\"}");
    }

    // DELETE (DELETE)
    @Override
    protected void doDelete(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

        int id = Integer.parseInt(request.getParameter("id"));

        response.setContentType("application/json");

        if (users.containsKey(id)) {
            users.remove(id);
            response.getWriter().write("{\"message\": \"User deleted successfully\"}");
        } else {
            response.getWriter().write("{\"error\": \"User not found\"}");
        }
    }
}
```

2. Add Servlet Mapping in web.xml (Optional)

If not using annotations, add:

```
<servlet>
   <servlet-name>UserServlet</servlet-name>
   <servlet-class>com.api.UserServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>UserServlet</servlet-name>
   <url-pattern>/users</url-pattern>
</servlet-mapping>
```

----

3. Deploy and Run on Tomcat

In Eclipse:

Right-click project → **Run As** → **Run on Server**

Choose Tomcat 9 or 10.

---

4. Test the CRUD API (Using Postman or Browser)

---

1. CREATE (POST)

```
POST http://localhost:8080/SimpleCRUDAPI/users
```

Body → JSON

```
{
  "id": "1",
  "name": "John",
  "email": "john@example.com"
}
```

Response:

```
{ "message": "User created successfully" }
```

---

2. READ (GET)

```
GET http://localhost:8080/SimpleCRUDAPI/users?id=1
```

Response:

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com"
}
```

---

3. UPDATE (PUT)

```
PUT http://localhost:8080/SimpleCRUDAPI/users?id=1
```

Body:

```
{
  "name": "Updated Name",
  "email": "updated@example.com"
}
```

Response:

```
{ "message": "User updated successfully" }
```

---

4. DELETE (DELETE)

```
DELETE http://localhost:8080/SimpleCRUDAPI/users?id=1
```

Response:

```
{ "message": "User deleted successfully" }
```