2️⃣

Bài 3: SERVLET (JAVA WEB)

1. Tổng quan về Servlet

1.1. Servlet là gì?

Servlet là một lớp Java chạy trên **Servlet Container** (ví dụ: Apache Tomcat), dùng để:

- Nhận và xử lý các yêu cầu HTTP từ client (trình duyệt)
- Truy xuất và xử lý dữ liệu (CSDL, service, API)
- Tạo phản hồi (HTML, JSON, XML, ...) trả về client

📌 Servlet đóng vai trò **Controller** trong mô hình MVC.

1.2. Vị trí của Servlet trong mô hình MVC

Thành phần	Vai trò
Model	Xử lý dữ liệu, nghiệp vụ
Servlet (Controller)	Nhận request, xử lý logic, điều hướng
JSP (View)	Hiển thị dữ liệu

---

2. Servlet Container

Servlet không chạy độc lập mà cần **Servlet Container**:

- Apache Tomcat
- Jetty
- GlassFish

Nhiệm vụ của Servlet Container:

- Quản lý vòng đời Servlet
- Mapping URL → Servlet
- Quản lý request / response

- Quản lý session

---

3. Cấu hình Servlet

3.1. Cấu hình bằng Annotation

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
}
```

3.2. Cấu hình bằng web.xml

```
<servlet>
  <servlet-name>HelloServlet</servlet-name>
  <servlet-class>com.example.HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>HelloServlet</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

---

4. Vòng đời Servlet

1. **init()** – Khởi tạo (chạy 1 lần)
2. **service()** – Xử lý request
3. **destroy()** – Giải phóng tài nguyên

---

5. Xử lý HTTP GET

```
public class HelloServlet extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse r
esponse)
      throws IOException {
```

```
    String name = request.getParameter("name");
    response.setContentType("text/html");

    PrintWriter out = response.getWriter();
    out.println("<h1>Hello, " + name + "!</h1>");
  }
}
```

📎 URL ví dụ:

```
http://localhost:9999/hello?name=Hiep
```

---

6. Xử lý HTTP POST

```
public class LoginServlet extends HttpServlet {

  protected void doPost(HttpServletRequest request, HttpServletResponse r
esponse)
      throws IOException, ServletException {

    String username = request.getParameter("username");
    String password = request.getParameter("password");

    if ("admin".equals(username) && "123456".equals(password)) {
      HttpSession session = request.getSession();
      session.setAttribute("user", username);
      response.sendRedirect("welcome");
    } else {
      request.setAttribute("errorMessage", "Sai username hoặc password!");
      RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
      rd.forward(request, response);
    }
  }
}
```

---

7. Request & Response

HttpServletRequest

- `getParameter()`
- `getAttribute()` / `setAttribute()`
- `getSession()`

HttpServletResponse

- `setContentType()`
- `sendRedirect()`
- `getWriter()`

---

8. Session trong Servlet

```
HttpSession session = request.getSession();
session.setAttribute("user", "admin");
```

📌 Session dùng để lưu trạng thái đăng nhập.

---

9. HTTP Methods

Giới thiệu HTTP Methods HTTP Methods là các phương thức giao tiếp giữa **client** và **server**. Trong Java Web, các HTTP Methods được xử lý thông qua các phương thức tương ứng trong `HttpServlet`.

HTTP Method	Servlet Method
GET	doGet()
POST	doPost()
PUT	doPut()
DELETE	doDelete()
HEAD	doHead()
OPTIONS	doOptions()
TRACE	doTrace()

9.2. GET – Lấy dữ liệu

9.2.1. Mục đích

- Lấy dữ liệu từ server
- Không làm thay đổi dữ liệu

9.2.2. Ví dụ URL

```
GET /hello?name=Hiep
```

9.2.3. Servlet ví dụ

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {

  protected void doGet(HttpServletRequest request, HttpServletResponse r
esponse)
      throws IOException {

    String name = request.getParameter("name");
    response.setContentType("text/html");

    PrintWriter out = response.getWriter();
    out.println("<h1>Hello, " + name + "!</h1>");
  }
}
```

9.3. POST – Gửi dữ liệu

9.3.1. Mục đích

- Gửi dữ liệu từ form
- Thường dùng cho đăng nhập, đăng ký

9.3.2. Form HTML

```
<form action="login" method="post">
  Username: <input type="text" name="username"><br>
```

```
  Password: <input type="password" name="password"><br>
  <input type="submit" value="Login">
</form>
```

9.3.3. Servlet ví dụ

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {

  protected void doPost(HttpServletRequest request, HttpServletResponse r
esponse)
      throws IOException, ServletException {

    String username = request.getParameter("username");
    String password = request.getParameter("password");

    if ("admin".equals(username) && "123456".equals(password)) {
      response.getWriter().println("Login success!");
    } else {
      response.getWriter().println("Login failed!");
    }
  }
}
```

9.4. PUT – Cập nhật toàn bộ dữ liệu

9.4.1. Mục đích

- Cập nhật hoặc thay thế toàn bộ resource
- Idempotent

9.4.2. Request ví dụ

```
PUT /users
BODY:
id=1&name=Hiep&email=hiep@example.com
```

9.4.3. Servlet ví dụ

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

  protected void doPut(HttpServletRequest request, HttpServletResponse re
sponse)
      throws IOException {

    String id = request.getParameter("id");
    String name = request.getParameter("name");

    response.getWriter().println("Updated user " + id + " - " + name);
  }
}
```

9.5. PATCH – Cập nhật một phần dữ liệu

9.5.1. Mục đích

- Cập nhật một phần resource

9.5.2. Ví dụ

```
PATCH /users
BODY:
email=new@example.com
```

9.5.3. Servlet (xử lý chung trong service)

```
protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws IOException, ServletException {

  if ("PATCH".equalsIgnoreCase(req.getMethod())) {
    resp.getWriter().println("PATCH request received");
  } else {
    super.service(req, resp);
```

```
  }
}
```

9.6. DELETE – Xóa dữ liệu

9.6.1. Mục đích

- Xóa resource

9.6.2. Ví dụ

```
DELETE /users?id=1
```

9.6.3. Servlet ví dụ

```
protected void doDelete(HttpServletRequest request, HttpServletResponse
response)
    throws IOException {

  String id = request.getParameter("id");
  response.getWriter().println("Deleted user with id = " + id);
}
```

9.7. HEAD – Lấy header

9.7.1. Mục đích

- Kiểm tra resource
- Không trả body

9.7.2. Servlet ví dụ

```
protected void doHead(HttpServletRequest request, HttpServletResponse r
esponse)
    throws IOException {
```

```
  response.setHeader("X-App-Status", "Alive");
}
```

9.8. OPTIONS – Kiểm tra method được hỗ trợ

```
protected void doOptions(HttpServletRequest request, HttpServletRespons
e response)
    throws IOException {

  response.setHeader("Allow", "GET,POST,PUT,DELETE");
}
```

9.9. TRACE – Debug request

```
protected void doTrace(HttpServletRequest request, HttpServletResponse r
esponse)
    throws IOException {

  response.getWriter().println("TRACE method invoked");
}
```

Tổng kết

Method	Dùng khi
GET	Lấy dữ liệu
POST	Gửi / tạo mới
PUT	Cập nhật toàn bộ
PATCH	Cập nhật một phần
DELETE	Xóa
HEAD	Kiểm tra
OPTIONS	Kiểm tra method
TRACE	Debug