SERVER COMPONENTS

I. Parameter (Tham số)

1. Form Parameter

Một FORM được định nghĩa trong 1 Web Page bắt đầu từ tag <form> và kết thúc bằng </from>.

Ta sử dụng INPUT TAG để nhập trực tiếp dữ liệu.

Cấu trúc của INPUT TAG

<input type = "…" value = "…" name = "…"/>

name cung cấp Parameter Name

value cung cấp Parameter Value

Vì vậy, mỗi Form Parameter bản chất là một cặp name – value.

2. ServletContext Initialization Parameters

Là một Parameter được khởi tạo để cung cấp thông tin khởi đầu cho Servlet.

Thiết lập bên trong web.xml.

```
<web-app>
<context-param>
<param-name>  parName  </param-name>
<param-value>  parValue  </param-value
</context-param>
…
```

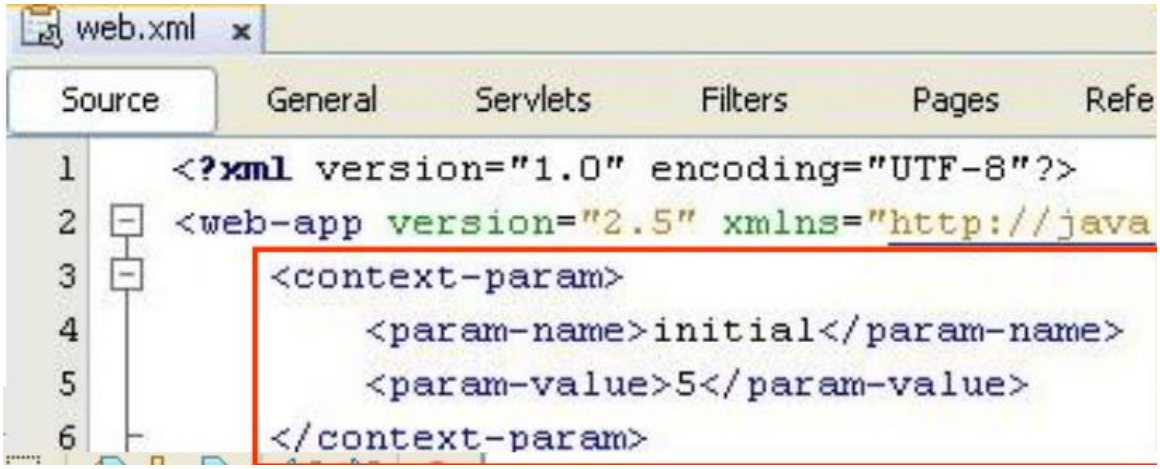
</web-app>

Bằng cách này, bằng cách set lại parName, parValue thì ta sẽ có những giá trị ban đầu, ta có thể lấy các giá trị này bằng cách

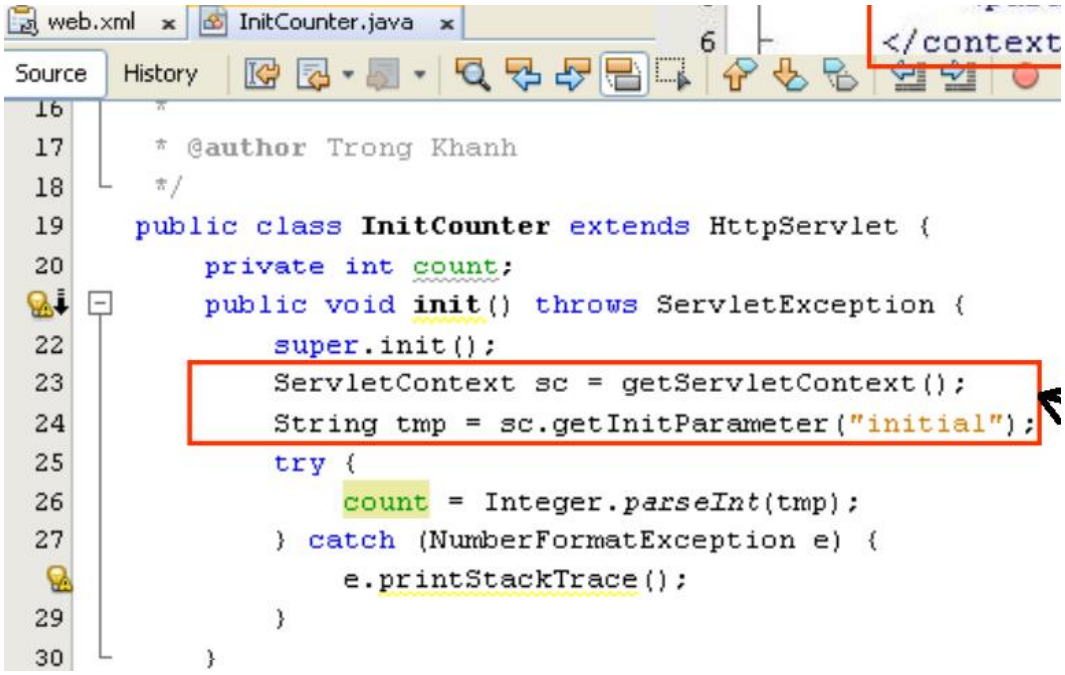
ServletContext sc = getServletContext();

String var = sc.getInitParameter("parName");

Bằng cách này, ta đã có thể lấy được giá trị của Context Initialization Parameter, và lưu vào biến String var.

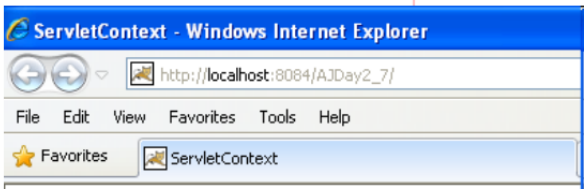


Hình 1: Khởi tạo Parameter "Name: initial – Value: 5"



Hình 2: Tạo Class con InitCounter của HttpServlet và khởi tạo biến count;

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        ...
        out.println("<body>");
        out.println("<h1>The ServletContext-Init Demo</h1>");
        count++;
        out.println("The web is accessed in " + count + "times");
        ...
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```



Hình 3: Sử dụng biến Count bên trong processRequest;

3. ServletConfig Initialization Parameters

3.1. The ServletConfig Interface

- Để thiết lập các Parameter trong suốt quá trình cài đặt, thì
- Servlet dùng 1 Object được gọi là ServletConfig Interface.

Điều chỉnh Servlet trước khi xử lý dữ liệu.

Methods	Descriptions
getServletName	- **public String getServletName()** - Searches the configuration information and retrieves name of the servlet instance - String servletName = getServletName();
getInitParameter	- **public String getInitParameter (String name)** - Retrieves the value of the initialisation parameter - Returns null if the specified parameter does not exist - String password = getInitParameter(”password”);
getServletContext	- **public ServletContext getServletContext()** - returns a ServletContext object used by the servlet to interact with its container. - ServletContext ctx = getServletContext();

Thiết lập bên trong web.xml.

```
<servlet>
<servlet-name> servletName </servlet-name>
<servlet-class> servletClass </servlet-
class>
<init-param>
<param-name> parName </param-name>
<param-value> parValue </param-value>
</init-param>
</servlet>
```

Bằng cách này, bằng cách set lại parName, parValue thì ta sẽ có những giá trị ban đầu, ta có thể lấy các giá trị này bằng cách

```
ServletConfig sc = getServletConfig();

String var = sc.getInitParameter(“parName");
```

Bằng cách này, ta đã có thể lấy được giá trị của Config Initialization Parameter, và lưu vào biến String var.

II. Scope

Xác định độ lâu của một vùng nhớ mà có sẵn trong Servlet Context.

Có 3 loại Scope:

1. Request Scope
   - Tồn tại từ HTTP Request chạm Container cho đến khi Servlet (Controller) chuyển HTTP Response.
   - Thư viện: javax.servlet.ServletRequest
   - Dữ liệu trên Request Scope thực chất là từ FormBean (Form Data), và sẽ làm mới lại mỗi khi nhận Request khác từ cùng hoặc khác User.
2. Session Scope

- Tồn tại từ khi 1 Web Browser được thành lập cho đến khi nó timeout hoặc bị đóng.
- Các giá trị có thể được truy cập từ nhiều Request khác nhau của cùng 1 Client (người dùng).
- Có thể được chia sẻ từ các tab khác nhau, miễn là cùng Client (Người dùng).
- Thư viện: javax.sevlet.HttpSession

3. Context Scope
    - Bắt đầu từ khi ứng dụng được DEPLOY cho đến khi WEB CONTAINER sụp xuống.
    - Các tham số, giá trị bên trong Context Scope sẵn sàng cho tất cả Request.

Có thể hiểu nôm na Scope chính là vùng nhớ của Server, và tùy vào loại Scope mà thời gian tồn tại khác nhau.

III. Web Container Model

1. Life Cycle của Servlet

LIFE CYCLE của Servlet gồm 3 hàm

- init(): Phương thức Servlet.init() được gọi từ Web Container để ám chỉ rằng, các tài nguyên của Servlet đã được cài đặt thành công và nó sẵn sàng cung cấp dịch vụ.
- service(): Phương thức Servlet.service() của Servlet được cầu khẩn để thông báo cho Servlet về những yêu cầu của Client (người dùng).
- destroy(): Phương thức được gọi ra để báo là thời gian hoạt động (lifetime) của Servlet đã kết thúc.

2. Servlet Container

- Là một COMPLIER, là một phần mềm có thể chạy
- Là TRUNG GIAN của Web Server và Servlet.
- Chạy, thiết lập và điều khiển Servlet.

+ Khi 1 Request đến, Container sẽ map (đơn ánh) Request Message và Servlet, biên dịch Request (và tạo ra Request Object) và gửi đến Servlet (hiểu là Request là từ Web Server mà nhờ cho Container gửi về cho Servlet).

+ Servlet xử lý Request và tạo ra Response Value. Container tạo ra Response Object, gán Response Value cho, và gửi trả về cho Web Server để nó biên dịch lại thành Response Message.

- Được thiết kế để xử lý nhiều Request Message 1 lúc.
- Có thể giữ bất cứ số lượng Servlets, Filters và Listeners.
- Cả Container và Objects bên trong đều là ĐA LUỒNG (Multi-Threading, nghĩa là xử lý song song).

3. Servlet Context

Được xem như là 1 Memory Segment (Bộ phận vùng nhớ) rằng mà của Container mà:

- Nằm ngoài Container, nhưng mà lại có mối quan hệ mật thiết với Container.
- Gom tất cả phương thức (method) rằng mà cần cho một vài ứng dụng Web cụ thể.
- Lưu trữ một vài đối tượng trong Server Side rằng toàn bộ thành phần của Web Server có thể truy cập.

- Tồn tại từ khi ứng dụng được deploy cho đến khi nó mất đi.

Container sử dụng ServletContext chí cốt để

- Gom các thành phần có liên quan.
- Chia sẻ dữ liệu dễ dàng hơn (vì là bộ nhớ dùng chung).
- Cung cấp các dịch vụ để ứng dụng Web có thể tương tác với Container.

Mỗi Servlet Context thường chỉ tương tác với 1 Web Server.

webapps

\day1

\WEB-INF

web.xml

\day2

intro.html

\WEB-INF

web.xml

Đây là 2 Context riêng của 2 Web (day1 và day2), trong đó day2 là 1 html tĩnh intro.html.

IV. Attribute

Để hiểu hơn, ta tiến hành so sánh Attribute và Parameters

	Attribute	Parameter

Định nghĩa	Là thông tin bên trong ứng dụng Web.	Là những thông tin đi vào ứng dụng Web.
Scope	Chúng có thể được định nghĩa hoặc truy cập bởi bất kì Scope	Request Scope
Kiểu dữ liệu	Object	String

Container sử dụng Attribute để:

- Cung cấp thông tin để hiểu lệnh: Nó cung cấp APIs (Giao diện lập trình ứng dụng) rằng cung cấp thông tin cho Container.
- Giữ thông tin để truy cập sau.

Developer có thể truy cập vào các Attribute bằng Attribute's Name.

Methods	Descriptions
getAttribute	- **public Object getAttribute(String name)** - returns the value of the name attribute as Object - Ex: String user = (String)servletContext.getAttribute("USER");
setAttribute	- **public void setAttribute(String name, Object obj)** - Binds an object to a given attribute name in the scope - Replace the attribute with new attribute, if the name specified is already used - servletContext.setAttribute("USER", "Aptech");
removeAttribute	- **public void removeAttribute(String name)** - Removes the name attributes - Ex: servletContext.removeAttribute("USER");
getAttributeNames	- **public Enumeration getAttributeNames()** - Returns an Enumeration containing the name of available attributes. Returns an empty if no attributes exist.

V. Request Dispatcher

Request Dispatcher là một INTERFACE, việc thực hiện xác định một đối tượng có thể gửi yêu cầu tới bất kỳ tài nguyên nào (như HTML, Image, JSP, Servlet) trên Server.

Ở đây, đối tượng RequestDispatcher rs đã được khởi tạo và sẽ gửi yêu cầu tới "hello.html". Bằng câu lệnh rs.forward() thì sẽ nó gửi nội dung bên trong () của forward() tới cho "hello.html".

Ở đây, cậu lệnh để lấy 1 RequestDispatcher là getRequestDispatcher();

forward	- **Redirect** the **output** to another servlet - **Forward** the **request** to another Servlet to process the client request. - **Ex**: RequestDispatcher rd = request.getRequestDipatcher("home.jsp"); rd.forward(request, response);
include	- **Include** the **content** of another servlet into the current output stream - Include the **output** of another Servlet to process the client request - **Ex** RequestDispatcher rd = request.getRequestDipatcher("home.jsp"); rd.include (request, response);

Hiểu là forward thì gửi đi tới home.jsp, trong khi include thì lại lấy nó từ home.jsp.

VI. DTO

- Là 1 Java Class
- Chứa thuộc tính, Getter, Setter
- Đối tượng của DTO thì map 1:1 với 1 hàng trong Database
- Classname: TableDTO
- Package name: table
- Imps: Serializable (để chuyển Object thành Stream)
- Được xem như là Constructor cơ bản

Minh họa DTO:

```
package sample.registration;
import ...
/**
 *
 * @author kieukhanh
 */
public class RegistrationDTO implements Serializable {
    private String username;
    private String password;
    private String lastname;
    private boolean role;
    public RegistrationDTO() {...2 lines }
    public RegistrationDTO(String username, String password,
            String lastname, boolean role) {...6 lines }
    /**...3 lines */
    public String getUsername() {...3 lines }
    /**...3 lines */
    public void setUsername(String username) {...3 lines }
    /**...3 lines */
    public String getPassword() {...3 lines }
    /**...3 lines */
    public void setPassword(String password) {...3 lines }
    /**...3 lines */
    public String getLastname() {...3 lines }
    /**...3 lines */
    public void setLastname(String lastname) {...3 lines }
    /**...3 lines */
    public boolean isRole() {...3 lines }
    /**...3 lines */
    public void setRole(boolean role) {...3 lines }
```