CÔNG NGHỆ JAVA

Nguyễn Hữu Thể

Bài 2: Servlet

Nội dung

- **Servlet init()**
- **web.xml**
- **Annotation**
- **Forward**
- **Redirect**

Interface Servlet

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

VD: InitParamServlet.java

+ init(): Khởi tạo giá trị cho email

+ doGet(): Khởi tạo giá trị cho email2

```
package com.javatech.tutorial.servlet;
public class InitParamServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private String email;
	public InitParamServlet() { }
	@Override
	public void init(ServletConfig config) throws ServletException {
		super.init(config);
		// Lấy giá trị của tham số khởi tạo. Cấu hình trong web.xml.
		this.email = config.getInitParameter("email1");
	}
	@Override
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
	throws ServletException, IOException {
		// Lấy giá trị của tham số khởi tạo theo một cách khác.
		String email2 = this.getServletConfig().getInitParameter("email2");
		ServletOutputStream out = response.getOutputStream();
		out.println("<html>");
		out.println("<body>");
		out.println("<h3>Init Param</h3>");
		out.println("<p>email1 = " + this.email + "</p>");
		out.println("<p>email2 = " + email2 + "</p>");
		out.println("</body>");
		out.println("<html>");
	}
}
```

web.xml
Thiết lập thông tin cấu hình

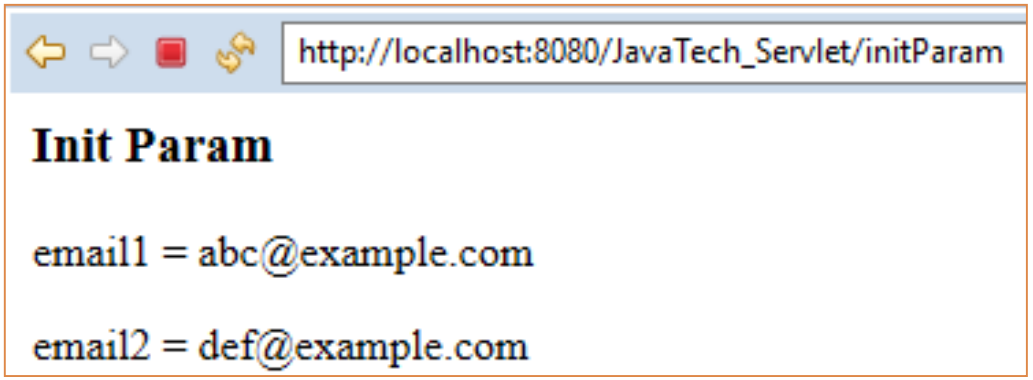
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
<display-name>JavaTech_Servlet</display-name>
 <servlet>
  <servlet-name>initParamServlet</servlet-name>
  <servlet-class>com.javatech.tutorial.servlet.InitParamServlet</servlet-class>

  <init-param>
   <param-name>email1</param-name>
   <param-value>abc@example.com</param-value>
  </init-param>

  <init-param>
   <param-name>email2</param-name>
   <param-value>def@example.com</param-value>
  </init-param>
 </servlet>

 <servlet-mapping>
  <servlet-name>initParamServlet</servlet-name>
  <url-pattern>/initParam</url-pattern>
 </servlet-mapping>

 <welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>
</web-app>
```



Sử dụng Annotation trong Servlet

@WebServlet

To declare a servlet.

@WebInitParam

To specify an initialization parameter.

@WebFilter

To declare a servlet filter.

@WebListener

To declare a WebListener

@WebServlet: VD1

```
@WebServlet("/HelloServlet")
public class HelloServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

      public HelloServlet() {
        super();
      }

    protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello, Good afternoon!");
    }
}
```

VD2: @WebServlet, @WebInitParam

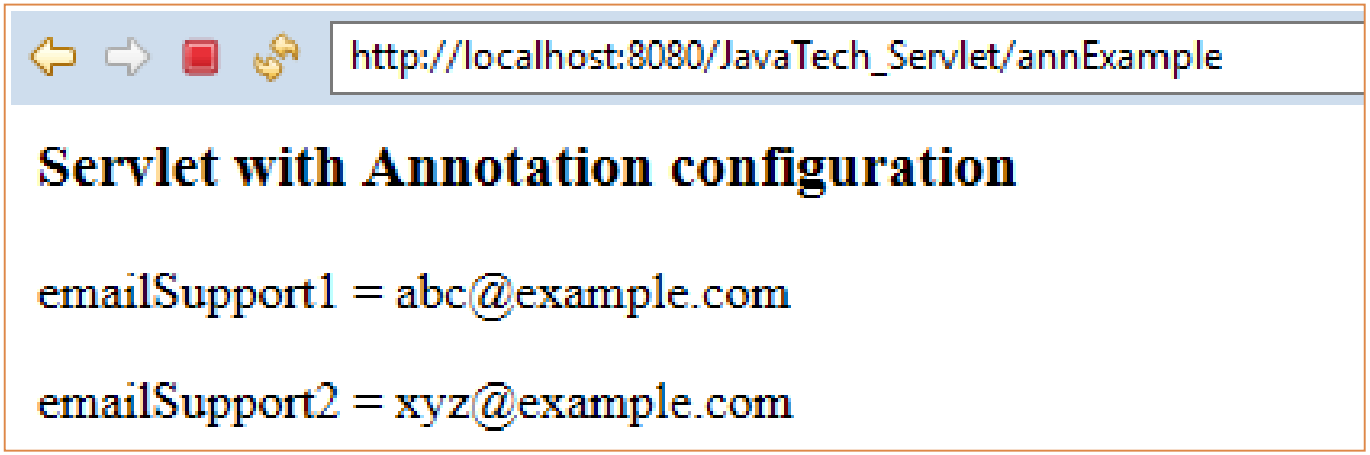
```
package com.javatech.tutorial.servlet;
// Có thể cấu hình một hoặc nhiều URL pattern để truy cập vào Servlet này.
@WebServlet(urlPatterns = { "/annotationExample", "/annExample" }, initParams = {
    @WebInitParam(name = "email1", value = "abc@example.com"),
    @WebInitParam(name = "email2", value = "xyz@example.com") })
public class AnnotationExampleServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        private String email1;
        public AnnotationExampleServlet() {  }
        @Override
        public void init(ServletConfig config) throws ServletException {
                super.init(config);
                this.email1 = config.getInitParameter("email1");
        }
        @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
                String email2 = this.getServletConfig().getInitParameter("email2");
                ServletOutputStream out = response.getOutputStream();
                out.println("<html>");
                out.println("<body>");
                out.println("<h3>Servlet with Annotation configuration</h3>");
                out.println("<p>emailSupport1 = " + this.email1 + "</p>");
                out.println("<p>emailSupport2 = " + email2 + "</p>");
                out.println("</body>");
                out.println("<html>");
        }
```

}

VD2: @WebServlet, @WebInitParam

http://localhost:8080/JavaTech_Servlet/annotationExample

http://localhost:8080/JavaTech_Servlet/annExample



url-pattern

- **url-pattern = "/any/*"**
  - So khớp với các path phía sau từ /any/
  - Ví dụ:

    http://localhost:8080/JavaTech_Servlet/any

    http://localhost:8080/JavaTech_Servlet/any/abc

    http://localhost:8080/JavaTech_Servlet/any/xyz/index.html
- **url-pattern = /**
  - Một **servlet** mặc định, **servlet** này sẽ được sử dụng để xử lý các request có đường dẫn không khớp với bất kỳ một **url-pattern** nào của các **Servlet** khác.
  - Ví dụ: đường dẫn không tồn tại sẽ gọi url-pattern này

    http://localhost:8080/JavaTech_Servlet/noname

Lấy thông tin Servlet Request

- Khi máy khách (trình duyệt) thực hiện yêu cầu GET, POST đối với Servlet,
  - Đối tượng HttpServletRequest giữ thông tin có giá trị về máy khách và chính yêu cầu đó.

❖ **Ví dụ**: lấy dữ liệu khác nhau từ một đối tượng Yêu cầu Servlet, như tiêu đề, tham số, đường dẫn và session

Forward (Chuyển tiếp)

- Chuyển tiếp yêu cầu tới một trang khác (hoặc một servlet khác).
- Địa chỉ trên trình duyệt của người dùng vẫn là đường dẫn của trang đầu tiên, nhưng nội dung của trang do trang được chuyển tiếp tới tạo ra.
- Trang được chuyển tiếp tới bắt buộc phải là một trang (hoặc Servlet) nằm trong cùng ứng dụng web.

forward(ServletRequest request, ServletResponse response)

VD:

```
RequestDispatcher rd =
request.getRequestDispatcher("pathToResource");
rd.forward(request, response);
```

Redirect (Chuyển hướng)

- Chuyển yêu cầu này tới một trang khác và kết thúc nhiệm vụ của nó.
- Trang được chuyển hướng tới có thể là trang trong ứng dụng, hoặc có thể là một trang bất kỳ.
- Địa chỉ trên trình duyệt của người dùng lúc này sẽ hiển thị đường dẫn của trang khác.

void *sendRedirect(String url)*

Forward() và SendRediret()

Forward()	**SendRediret()**
Khi chúng tôi sử dụng phương pháp chuyển tiếp yêu cầu là chuyển đến tài nguyên khác **trong cùng một máy chủ** để tiếp tục xử lý.	Trong trường hợp yêu cầu là sendRedirect là chuyển hướng đến tài nguyên khác như tên miền khác hoặc các đường dẫn là các máy chủ khác nhau để xử lý.
Nhìn bề ngoài chúng ta không thể thấy địa chỉ chuyển tiếp như vậy sẽ bảo mật hơn.	Trong thanh địa chỉ chúng tôi có thể thấy địa chỉ chuyển hướng mới.
Sử dụng phương pháp chuyển tiếp *forward()* thì tốc độ chuyên tiếp đến trang xử lý là nhanh hơn.	SendRedirect lnó chuyển hướng chậm hơn bởi vì một chuyến đi vòng thêm là cần thiết . yêu cầu mới được tạo ra cho nên tốc độ xử lý sẽ chậm hơn so với *forward*.
Khi sử dụng forward thì chúng ta có thể gửi dữ liệu sang một trang được chuyển tiếp đến bằng cách sử dụng request.setAttribute ()	Nhưng khi sử dụng sendRedirect nếu chúng ta muốn gửi dữ liệu đi thì phải lưu trữ các dữ liệu trong phiên làm việc hoặc đi cùng với URL.