web.xml Servlet Configuration

For a Java servlet to be accessible from a browser, you must tell the servlet container what servlets to deploy, and what URL's to map the servlets to. This is done in the web.xml file of your Java web application.

Configuring and Mapping a Servlet

To configure a servlet in the web.xml file, you write this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

  <servlet>
    <servlet-name>controlServlet</servlet-name>
    <servlet-class>com.jenkov.butterfly.ControlServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>controlServlet</servlet-name>
    <url-pattern>*.html</url-pattern>
  </servlet-mapping>
</web-app>
```

First you configure the servlet. This is done using the `<servlet>` element. Here you give the servlet a name, and writes the class name of the servlet.

Second, you map the servlet to a URL or URL pattern. This is done in the `<servlet-mapping>` element. In the above example, all URL's ending in `.html` are sent to the servlet.

Other possible servlet URL mappings are:

```
/myServlet

/myServlet.do

/myServlet*
```

The * is a wild card, meaning any text. As you can see, you can either map a servlet to a single, specific URL, or to a pattern of URL's, using a wild card (*). What you will use depends on what the servlet does.

Servlet Init Parameters

You can pass parameters to a servlet from the web.xml file. The init parameters of a servlet can only be accessed by that servlet. Here is how you configure them in the web.xml file:

```
<servlet>
    <servlet-name>controlServlet</servlet-name>
    <servlet-class>com.jenkov.butterfly.ControlServlet</servlet-class>

        <init-param>
            <param-name>myParam</param-name>
            <param-value>paramValue</param-value>
        </init-param>
</servlet>
```

Here is how you read the init parameters from inside your servlet - in the servlets `init()` method:

```
public class SimpleServlet extends GenericServlet {

  protected String myParam = null;

  public void init(ServletConfig servletConfig) throws ServletException{
    this.myParam = servletConfig.getInitParameter("myParam");
  }

  public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {

    response.getWriter().write("<html><body>myParam = " +
            this.myParam + "</body></html>");
  }
}
```

A servlets `init()` method is called when the servlet container loads the servlet for the first time. No one can access the servlet until the servlet has been loaded, and the `init()` method has been called successfully.

Servlet Load-on-Startup

The `<servlet>` element has a subelement called `<load-on-startup>` which you can use to control when the servlet container should load the servlet. If you do not specify a `<load-on-startup>` element, the servlet container will typically load your servlet when the first request arrives for it.

By setting a `<load-on-startup>` element, you can tell the servlet container to load the servlet as soon as the servlet container starts. Remember, the servlets `init()` method is called when the servlet is loaded.

Here is an example `<load-on-startup>` configuration:

```
<servlet>
    <servlet-name>controlServlet</servlet-name>
    <servlet-class>com.jenkov.webui.ControlServlet</servlet-class>
    <init-param><param-name>container.script.static</param-name>
            <param-value>/WEB-INF/container.script</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

The number inside the **`<load-on-startup>1</load-on-startup>`** element tells the servlet container in what sequence the servlets should be loaded. The lower numbers are loaded first. If the value is negative, or unspecified, the servlet container can load the servlet at any time.

Context Parameters

You can also set some context parameters which can be read from all servlets in your application. Here is how you configure a context parameter:

```
<context-param>
    <param-name>myParam</param-name>
    <param-value>the value</param-value>
</context-param>
```

Here is how you access the parameter from inside an `HttpServlet` subclass:

```
String myContextParam =
        request.getServletContext()
               .getInitParameter("myParam");
```