Initialization

Problems with hard-coding values of variable

- The servlet member variables are usually fixed values. These do not change with respect to end user.
- While this is true, should we hard code information like database URL, user name and password?
- While developing the application we have one set of values for these, while in testing we have we will have another set of values, while in UAT we have yet another and finally we have actual values when we deploy.
- So at the minimum we need go to the specific location and to compile the code at least 4 times.
- Also things become complicated here because we are expecting all the people involved to be servlets savvy!.

Initialization

- After the servlet object is instantiated it is initialized. Container calls the **`init(ServletConfig c)`** method to do this.
- Initialization is provided so that a servlet can read persistent configuration data, initialize costly resources and perform other one-time activities.
- This configuration object allows the servlet to access name-value initialization parameters from the Web application’s configuration information.
- It also allows access to **`ServletContext`** object that is used to that describes the servlet’s runtime environment.

Scope of initial values in a web application

- The Initialization values can be either
  - Servlet specific: values that is accessible only to the servlet
    - For this `ServetConfig` interface is used
  - Application specific: values that is accessible all the servlet
    - For this `ServetContext` interface is used

Servlet specific values

- The values can be given either using Annotation or in web.xml.
- Annotation **`@WebInitParam`** can be used in Servlet class to provide the name and values for parameters.
- In web.xml:

```
<servlet>
<servlet-name>servlet-name</servlet-name>
<init-param>
  <param-name>param-name</param-name>
  <param-value>param-value</param-value>
</init-param>
…
</servlet>
```

`init()`

- Servlet instance variables are initialized in **`init()`**.
- The **`ServetConfig`** object is a configuration object used by a servlet container to pass information to a servlet during initialization. Most of the time this information will be required to initialize variables.
- There are two places where initialization can be done:
  - Override **`init(ServletConfig config)`**
  - Override **`init()and`** get **`config`** object using **`getServletConfig() method(of Servlet interface)`**

`ServletConfig`

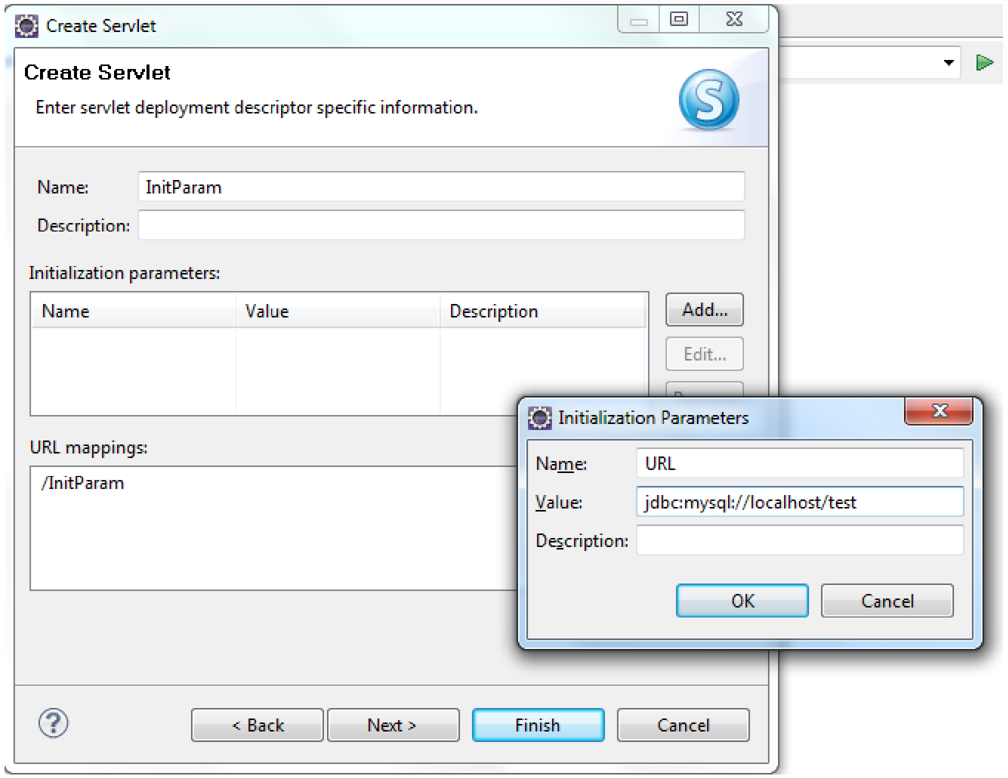
- This object is used to fetch the initialization parameters for individual servlets.
- This object is passed on by the application server to the servlet during initialization when **`init(ServletConfig)`** method is called.
- Getting **`ServletConfig`** object:

  `ServletConfig getServletConfig()` method in **`Servlet interface`**
- Methods to get init parameters in **`ServletConfig`**

  `String getInitParameter(String name)`

  `Enumeration getInitParameterNames()`
- **`Since GenericServlet`** class implements also **`ServletConfig interface it`** provides the same method as the above.

Example: Using annotation



Create Servlet

Create Servlet

Enter servlet deployment descriptor specific information.

Name: InitParam

Description:

Initialization parameters:

Name	Value	Description

URL mappings:

/InitParam

Initialization Parameters

Name: URL

Value: jdbc:mysql://localhost/test

Description:

- While creating Servlet, after entering Servlet name, package name move to next tab.
- Click on Add button in the Initialization parameter sections and add a name value pair.
- Move to the next tab and select init() and doGet() method .

Servlet code generated

- The servlet code generated has

```
@WebServlet(
urlPatterns = { "/InitParam" },
initParams = {
@WebInitParam(name = "url", value =
  "jdbc:mysql://localhost/test")
})
```

- To get code added in `init()` method:

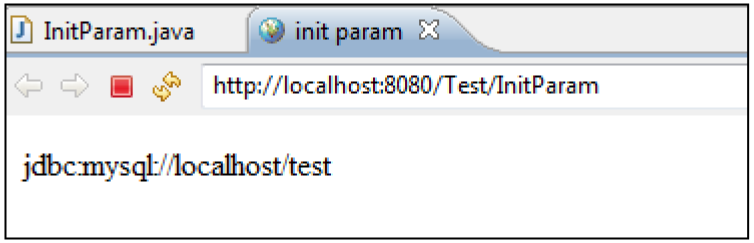
```
url=config.getInitParameter("url");
```

- Code **`doGet()`** method to get the URL value:

```
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws
ServletException, IOException {
response.setContentType("text/html");

java.io.PrintWriter out = response.getWriter();

out.println("<html><head><title>init param
</title></head>");
out.println("<body>"+ url+"</body></html>");
}
```



```
jdbc:mysql://localhost/test
```

Application specific parameters

- It is more likely that some of configuration values are more application specific than servlet specific.
- For instance database related configuration information usually should be made available to all the components of the web application.
- The ServletContext interface defines a servlet’s view of the Web application within which the servlet is running.
- A ServletContext is rooted at a known path within a Web server.
- These name-value pairs can be given only in web.xml.

`ServletContext`

- This object is global to all servlets and jsps in an application context.
- It is used to set and get variables that are global application variables.
- It is also used by the servlet to get info about the servlet engine this servlet is running and info about other servlets.
- Getting ServletContext
- **`ServletContext getServletContext()`** method of **`HttpServlet`**
- Methods of **`ServletContext`**

```
String getInitParameter(String name)
Enumeration getInitParameterNames()
```

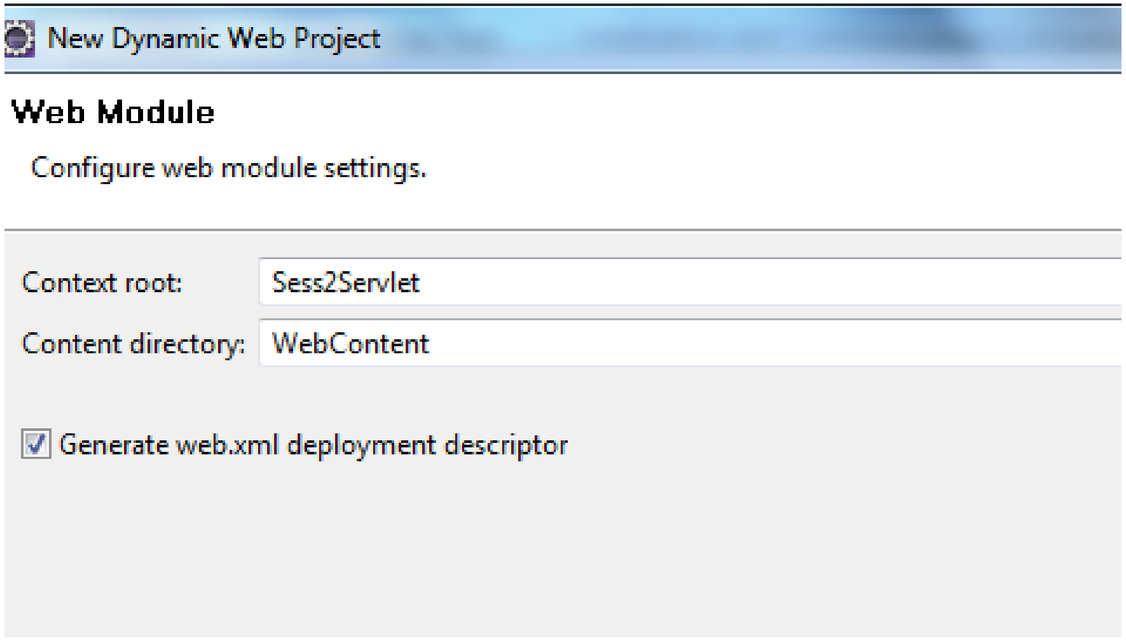
Context parameters in web.xml

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

In any servlet in of the current application:

```
ServletContext ctx=getServletContext();
String value=ctx.getInitParameter(“param-name”);
```

A web module with web.xml in eclipse



New Dynamic Web Project

Web Module

Configure web module settings.

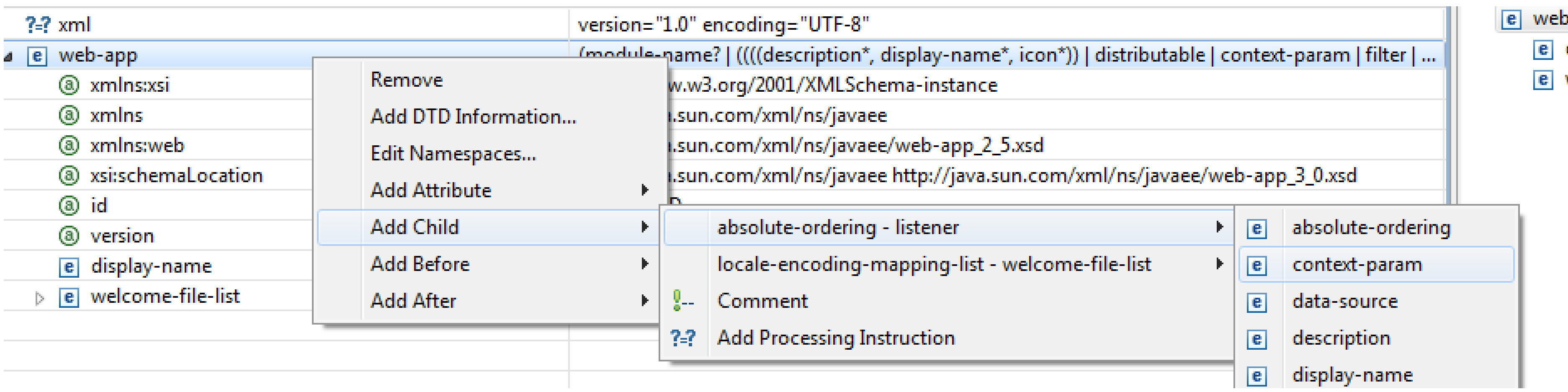
Context root: Sess2Servlet

Content directory: WebContent

[✓] Generate web.xml deployment descriptor

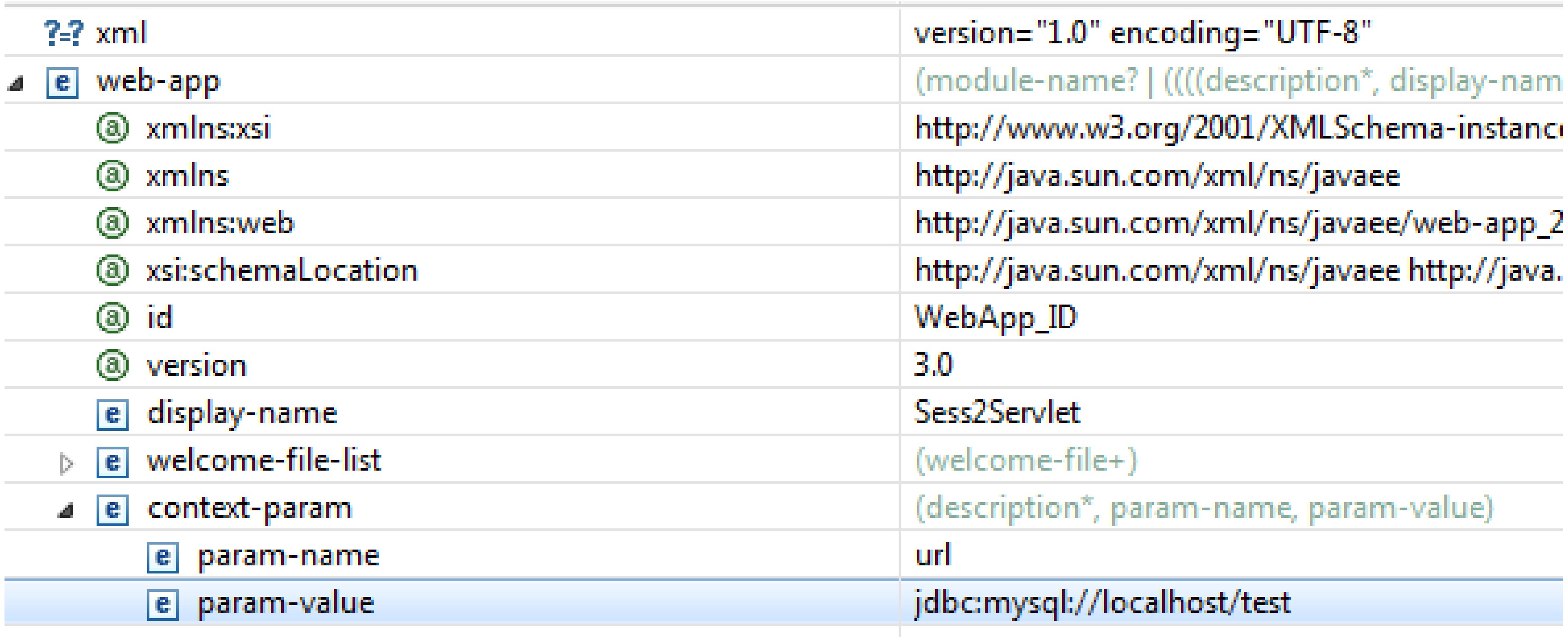
- Create a new dynamic web project.
- After specifying application name etc in the first tab, move on and click next.
- Select the checkbox “Generate web.xml deployment descriptor” and then click finish.

Adding context parameter in eclipse



?-? xml	version="1.0" encoding="UTF-8"
web-app	(module-name? | ((((description*, display-name*, icon*)) | distributable | context-param | filter | ...
xmlns:xsi	w.w3.org/2001/XMLSchema-instance
xmlns	.sun.com/xml/ns/javaee
xmlns:web	.sun.com/xml/ns/javaee/web-app_2_5.xsd
xsi:schemaLocation	.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd
id	
version	
display-name	
welcome-file-list	

- Double click on Deployment Decsriptor.
- It will open up a page similar to the above picture.
- Right click on the web-app and select context-param as shown.
- Change the param name and value to “url” and value to “jdbc:mysql://localhost/test” as shown in the next slide

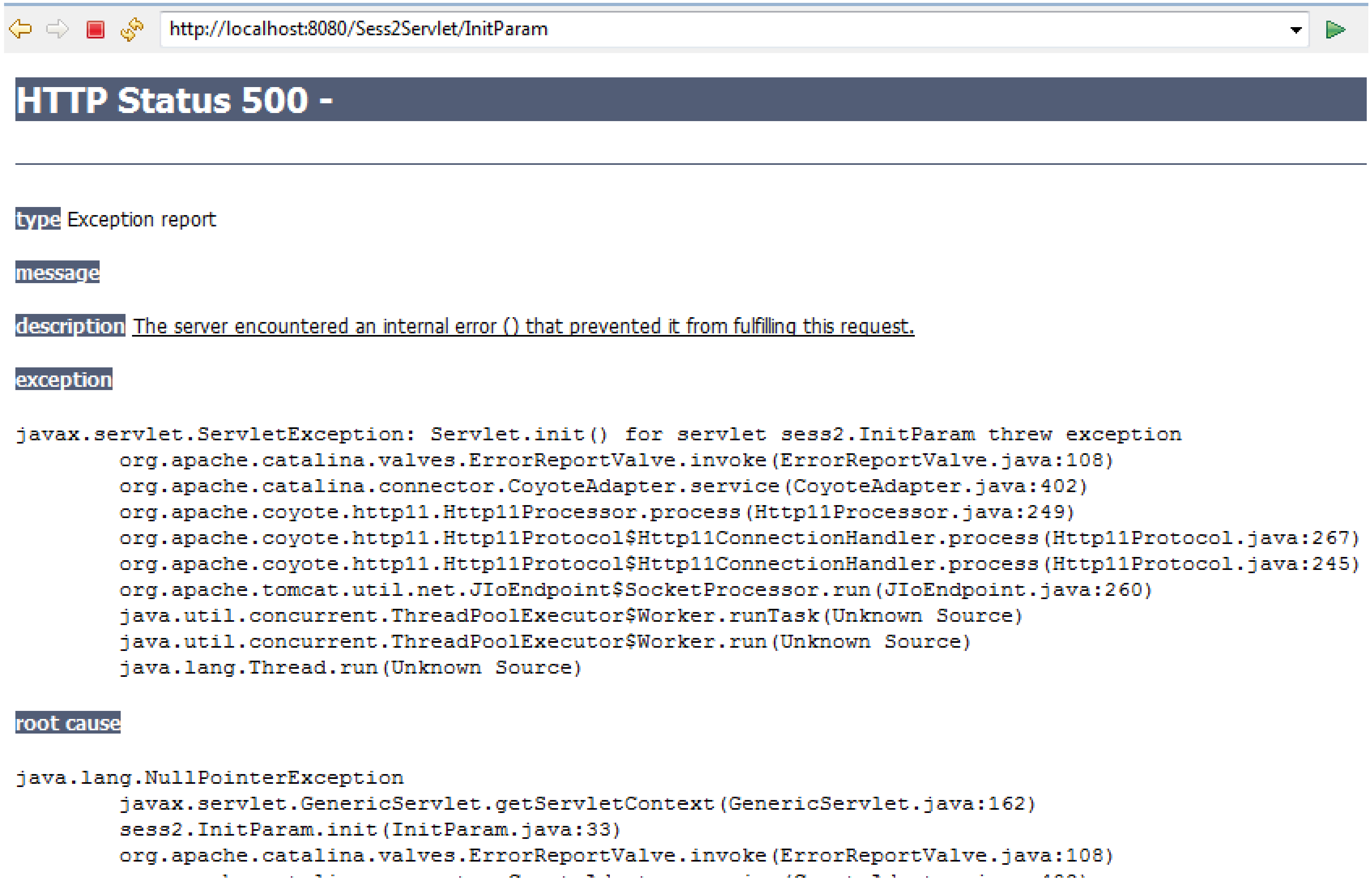


?-? xml	version="1.0" encoding="UTF-8"
◢ e web-app	(module-name? | ((((description*, display-nam
ⓐ xmlns:xsi	http://www.w3.org/2001/XMLSchema-instanc
ⓐ xmlns	http://java.sun.com/xml/ns/javaee
ⓐ xmlns:web	http://java.sun.com/xml/ns/javaee/web-app_2
ⓐ xsi:schemaLocation	http://java.sun.com/xml/ns/javaee http://java.
ⓐ id	WebApp_ID
ⓐ version	3.0
e display-name	Sess2Servlet
▷ e welcome-file-list	(welcome-file+)
◢ e context-param	(description*, param-name, param-value)
e param-name	url
e param-value	jdbc:mysql://localhost/test

Code in Servlet to get context parameter

```
 String url;
public void init(ServletConfig config) throws
   ServletException {
 url=getServletContext().getInitParameter("url");
}
protected void doGet(HttpServletRequest request,
   HttpServletResponse response) throws
   ServletException, IOException {
   response.setContentType("text/html");
   java.io.PrintWriter out = response.getWriter();
     out.println("<html><head><title>init param
   </title></head>");
     out.println("<body>"+ url+"</body></html>");
}
```

On Execution



http://localhost:8080/Sess2Servlet/InitParam

HTTP Status 500 -

type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

```
javax.servlet.ServletException: Servlet.init() for servlet sess2.InitParam threw exception
	org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:108)
	org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:402)
	org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:249)
	org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.process(Http11Protocol.java:267)
	org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.process(Http11Protocol.java:245)
	org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:260)
	java.util.concurrent.ThreadPoolExecutor$Worker.runTask(Unknown Source)
	java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
	java.lang.Thread.run(Unknown Source)
```

root cause

```
java.lang.NullPointerException
	javax.servlet.GenericServlet.getServletContext(GenericServlet.java:162)
	sess2.InitParam.init(InitParam.java:33)
	org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:108)
```

Tell me why

- Why does the previous code throw **`NullPointerException`** ?
- Take a look at the **`init()`** method.

```
public void init(ServletConfig config) throws
ServletException {
 url=getServletContext().getInitParameter("url");
}
```

- The super class **`init()`** method that takes **`ServletConfig`** object initializes **`ServletContext`** object.
- Since we are not calling super class **`init()`**, **`getServletContext()`** returns **`null`** and hence an exception is thrown.
- How will you correct this problem?

Ways to get Context parameters in init()

1.
   ```
   public void init(ServletConfig config) throws
   ServletException {
   super.init(config);
   url=getServletContext().getInitParameter("url");
   }
   ```

2.
   ```
   public void init() throws ServletException {
   url=getServletContext().getInitParameter("url");
   }
   ```

   Life cycle method which is called first is **`init(ServletConfig config)`**. This method in turn calls **`init()`** method. Hence **`ServletContext`** object is available in **`init()`**.

Errors while initialization

- If **`init()`** method throws an runtime exception other than, then the **`service()`** method is not called.
- This also means that **`destroy()`** method is not called.
- So the cleanup code of **`destroy()`** method must be provided in the init() methods catch block in cases where there is a possibility of an exception being thrown.
- A new instance may be instantiated and initialized by the container after a failed initialization (except in case of **`UnavailableException`**)

UnavailableException

- The `UnavailableException` can be thrown from the init() method.
- This exception can indicate the min time the servlet will be unavailable.
- `UnavailableException(java.lang.String` msg, int seconds)
- In this case, the container waits for the period to pass before creating and initializing a new servlet instance.