How to Create Custom Error Pages

3 ways in creating custom error pages

1. web.xml

The standard webapp configuration file located in <webapp>/WEB-INF/web.xml can be used to map errors to specific URLs with the `<error-page>` element. This element creates a mapping between the `error-code` or `exception typ` to the `location` of a resource in the web application.

- `error-code` - integer value
- `exception-type` - fully qualified class name of a Java Exception type
- `location` - location of the resource in webapp relative to the root of the web application. Value should start with "**/**".

Error code example:

```
<error-page>
  <error-code>404</error-code>
  <location>/jspsnoop/ERROR/404</location>
</error-page>
```

Exception example:

```
<error-page>

<exception-type>java.io.IOException</exception-type>
  <location>/jspsnoop/IOException</location>
</error-page>
```

2. context file configuration.

Context files are nomrall located in <jetty.home>/contexts/?.xml (see ContextDeployer for more detail). Context files can be used to configure the default error handler provided for a context with more flexibility than is available with web.xml,
specifically with the support of error code ranges:

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay
Consulting//DTD Configure//EN"
"http://jetty.mortbay.org/configure.dtd">

<Configure
class="org.mortbay.jetty.webapp.WebAppContex
t">
  <Set name="contextPath">/test</Set>
  <Set name="war">
    <SystemProperty name="jetty.home"
default="."/>/webapps/test
  </Set>

  <!-- by Code -->
  <Get name="errorHandler">
    <Call name="addErrorPage">
      <Arg type="int">404</Arg>
      <Arg
type="String">/jspsnoop/ERROR/404</Arg>
    </Call>
  </Get>

  <!-- by Exception -->
  <Get name="errorHandler">
    <Call name="addErrorPage">
      <Arg>
        <Call class="java.lang.Class"
name="forName">
          <Arg
type="String">java.io.IOException</Arg>
```

```
        </Call>
      </Arg>
      <Arg
type="String">/jspsnoop/IOException</Arg>
    </Call>
  </Get>

  <!-- by Code Range -->
  <Get name="errorHandler">
    <Call name="addErrorPage">
      <Arg type="int">500</Arg>
      <Arg type="int">599</Arg>
      <Arg
type="String">/dump/errorCodeRangeMapping</A
rg>
```

```
    </Call>
  </Get>
</Configure>
```

3. Custom error handle class.

A context may be configured with a custom error handler class that extends ErrorHandler (for webapp contexts it must extend ErrorPageErrorHandler).

The following methods may be implemented to control the appearance of the error pages:

- public void handle(String target, HttpServletRequest request, HttpServletResponse response, int dispatch) throws IOException
- void handleErrorPage(HttpServletRequest request, Writer writer, int code, String message) throws IOException
- void writeErrorPage(HttpServletRequest request, Writer writer, int code, String message, boolean showStacks) throws IOException
- void writeErrorPageHead(HttpServletRequest request, Writer writer, int code, String message) throws IOException
- void writeErrorPageBody(HttpServletRequest request, Writer writer, int code, String message, boolean showStacks) throws IOException
- void writeErrorPageMessage(HttpServletRequest request, Writer writer, int code, String message,String uri) throws IOException
- void writeErrorPageStacks(HttpServletRequest request, Writer writer) throws IOException

The custom error handler may be set on the context via the API or via a context configuration file. For example a custom error handling class can be added to the javadoc context with:

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay
Consulting//DTD Configure//EN"
"http://jetty.mortbay.org/configure.dtd">

<Configure
class="org.mortbay.jetty.handler.ContextHand
ler">
  <Call class="org.mortbay.log.Log"
name="debug"><Arg>Configure
javadoc.xml</Arg></Call>
  <Set name="contextPath">/javadoc</Set>
  <Set name="resourceBase"><SystemProperty
name="jetty.home"
default="."/>/javadoc/</Set>

  <!-- Instantiate your own error handler
-->
  <Set name="errorHandler">
    <New
class="com.acme.handler.MyErrorHandler"/>
  </Set>

</Configure>
```

4. Server level 404 error

One may get a 'page not found' when a request is made to the server for a resource that is outside of any registered contexts. As an example, you have a domain name pointing to your public server IP yet no context is registered with jetty to serve pages for that domain. As a consequence, the server, by default, will give a listing of all contexts running on the server.

One of the quickest ways to avoid this behavior is to create a catch all context. Create a "root" web app mapped to the "/" url. Have the index.html redirect to whatever place with a header directive.