# Jsp Configuration

## Jsp Configuration

There are many configuration parameters for the jsp engine. Some parameters only affect pre-compilation, and some affect runtime recompilation checking. Parameters also differ between the 2.0 and 2.1 release of the jsp engine. This page lists the configuration parameters, their meaning and their default settings.

> **Be Careful**
>
> For all of the parameter names below, if its not working, then try using all lower case instead of camel case, as the implementation is quite inconsistent in its parameter naming strategy.
>
> Also be aware that for JDK1.6, the default for "keepgenerated" is "false", therefore .java files will only be left after compilation if there is a compilation error.

### Jsp 2.1

| init param | Description | Default | webdefault.xml |
|---|---|---|---|
| development | If `development=true`, recompilation checks are made on each request. See also `modificationTestInterval` | TRUE | - |
| fork | Should Ant fork its java compiles of JSP pages. | TRUE | FALSE |
| keepgenerated | Do you want to keep the generated Java files around? | FALSE | - |
| saveByteCode | If class files are generated as byte arrays, should they be saved to disk at the end of compilations? | FALSE | - |
| trimSpaces | Should white spaces between directives or actions be trimmed? | FALSE | - |
| enablePooling | Determines whether tag handler pooling is enabled | TRUE | - |
| mappedFile | Support for mapped Files. Generates a servlet that has a print statement per line of the jsp file. | TRUE | - |

| | | | |
|---|---|---|---|
| sendErrorToClient | If false, stack traces etc are sent to std error instead of the clients browser | FALSE | - |
| classdebuginfo | Include debugging info in class file | TRUE | - |
| checkInterval | Interval in seconds between background recompile checks. Only relevant if `development=false` | 0 | - |
| suppressSmap | Generation of SMAP info for JSR45 debugging | FALSE | - |
| dumpSmap | Dump SMAP JSR45 info to a file | FALSE | - |
| genStrAsCharArray | Option for generating Strings. | FALSE | - |
| genStrAsByteArray | Option for generating Strings | TRUE | - |
| defaultBufferNone | | FALSE | - |
| errorOnUseBeanInvalidClassAttribute | | FALSE | - |
| scratchDir | Directory where servlets are generated. Set by Jetty according to the work dir settings for the webapp. | - | - |
| compiler | Determined at runtime. For Jetty is the eclipse jdt compiler. | | |
| compilerTargetVM | Target vm to compile for. | 1.5 | - |
| compilerSourceVM | Sets source compliance level for the jdt compiler. | 1.5 | - |
| javaEncoding | Pass through the encoding to use for the compilation | UTF8 | - |

| | | | |
|---|---|---|---|
| modificationTestInterval | If `development=true`, interval between recompilation checks, triggered by a request | 0 | - |
| xpoweredBy | Generate an X-Powered-By response header | FALSE | FALSE |
| usePrecompiled/use-precompiled | | FALSE | - |
| validating/enableTldValidation | Whether or not to validate tag files against the schema | FALSE | |
| reload-interval | If `reload-interval=0`, then no runtime checking of jsps, otherwise sets the checking interval for both `development=true` and `development=false` | | |
| initial-capacity/initialCapacity | The initial capacity of the hash maps mapping the name of the jsp to class and jsp file | | |

Much confusion generally ensues about the `development`, `checkInterval` and `modificationTestInterval` parameters and jsp runtime recompilation. Here is a factoring out of the various options:

1. Check the jsp files for possible recompilation on every request:

```
<init-param>
        <param-name>development</param-name>
        <param-value>true</param-value>
</init-param>
```

2. Only check approximately every N seconds, where a request will trigger the time-lapse calculation. This example checks every 60 seconds:

```
<init-param>
        <param-name>development</param-name>
        <param-value>true</param-value>
</init-param>
<init-param>
        <param-name>modificationTestInterval</param-name>
        <param-value>60</param-value>
</init-param>
```

3. Do no checking whatsoever, but still compile the jsp on the very first hit. (Note: this "reload-interval" parameter is a short hand for a "development=false" and "checkInterval=0" combination):

```
<init-param>
        <param-name>reload-interval</param-name>
        <param-value>-1</param-value>
</init-param>
```

4. Don't do any request-time checking, but instead start a background thread to do checks every N seconds (in the example below, every 60 sec):

```
<init-param>
        <param-name>development</param-name>
        <param-value>false</param-value>
</init-param>
<init-param>
        <param-name>checkInterval</param-name>
        <param-value>60</param-value>
</init-param>
```

## Modifying the Configuration

There are several options for modifying the Jasper JSP servlet configuration. Which is most suitable will depend on your requirements.

### Override webdefault.xml

You can make a copy of the webdefault.xml shipped with jetty, apply your changes and use it instead of the shipped version. The example below shows
how to do this when using the maven jetty plugin.

**"Overiding webdefault.xml"**

```
<plugin>
   <groupId>org.mortbay.jetty</groupId>
   <artifactId>jetty-maven-plugin</artifactId>
   <configuration>
   <webAppConfig>

<defaultsDescriptor>src/main/resources/webdefault.xml</defaultsD
escriptor>
   </webAppConfig>
   <connectors>
   ..
```

If you're using jetty standalone, and you want to change the jsp settings for just one or a few of your webapps, then copy the `$JETTY_HOME/etc/webdefault.xml` file somewhere, modify it, and then use a context xml file to set this file as the webdefaults for your webapp. Here's a snippet:

```
<Configure
class="org.eclipse.jetty.webapp.WebAppContex
t">

  <Set name="contextPath">/foo</Set>
  <Set name="war"><SystemProperty
name="jetty.home"
default="."/>/webapps/foobar.war</Set>
  <Set
name="defaultsDescriptor">/home/smith/dev/we
bdefault.xml</Set>

</Configure>
```

If you want to change the jsp settings for all of the webapps, then edit the `$JETTY_HOME/etc/webdefaults.xml` file directly instead.

### Configure JSP Servlet in web.xml

Another option is to add an entry for the Jasper JSP servlet to the *web.xml* file of your webapp.

You can use the entry in webdefault.xml as a starting point.

**"Configuring JSP Servlet in web.xml"**

```
<servlet id="jsp">
    <servlet-name>jsp</servlet-name>

<servlet-class>org.apache.jasper.servlet.JspServlet</servlet-cla
ss>
    <init-param>
        <param-name>logVerbosityLevel</param-name>
        <param-value>DEBUG</param-value>
    </init-param>
    <init-param>
        <param-name>fork</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>keepgenerated</param-name>
        <param-value>true</param-value>
    </init-param>
    ...

    <load-on-startup>0</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>jsp</servlet-name>
    <url-pattern>*.jsp</url-pattern>
    <url-pattern>*.jspf</url-pattern>
    <url-pattern>*.jspx</url-pattern>
    <url-pattern>*.xsp</url-pattern>
    <url-pattern>*.JSP</url-pattern>
    <url-pattern>*.JSPF</url-pattern>
    <url-pattern>*.JSPX</url-pattern>
    <url-pattern>*.XSP</url-pattern>
  </servlet-mapping>

  <servlet id="my-servlet">
    <servlet-name>myServlet</servlet-name>
    <servlet-class>com.acme.servlet.MyServlet</servlet-class>
     ...
```