

JfastCGI 2.0

Reference documentation

Summary

JfastCGI 2.0.....	1
Description :.....	1
Installing the library in your webapp :	1
Pooling of fastgci connections :	1
Portlet support :	1
Spring integration :.....	2
Running the simple way : the FastCGIServlet.....	2
Using several connections for the same app.....	2
Doing fancy things : writing a custom handler :	3
Using Spring instead of servlets.....	4
Using portlets :	4

Description :

JfastCGI allows a servlet container to act as a fastCGI gateway that serves content generated by a fastCGI producer. It gives an original and reliable way for using servlet containers such as tomcat for serving PHP pages, or any fastcgi application.

Installing the library in your webapp :

Copy jFastCGI-1.1.jar to the /WEB-INF/lib directory of your web project.

You also need to have commons-logging.jar in your application's classpath.

The following jars are optional, only necessary for certain features :

Pooling of fastgci connections :

commons-pool-1.4.jar

Portlet support :

portlet.jar (FastCGIPortlet is compatible with jsr170)

Spring integration :

spring-2.5.6 or higher.

Running the simple way : the FastCGIServlet

In most case, if you don't want to use pooling between several fastcgi servers, you just have to declare the fastCGI Servlet.

You have to tell it the tcp connection that have to be made to the external fastcgi program :

```
<servlet>
  <servlet-name>FastCGI</servlet-name>
  <servlet-class>net.jr.fastcgi.FastCGIServlet</servlet-class>
  <init-param>
    <param-name>server-address</param-name>
    <param-value>localhost:6666</param-value>
  </init-param>
</servlet>
```

If you, for example, use JfastCGI with php, just map the servlet with the php files by adding the <servlet-mapping> configuration in the web.xml :

```
<servlet-mapping>
  <servlet-name>FastCGI</servlet-name>
  <url-pattern>*.php</url-pattern>
</servlet-mapping>
```

The servlet may optionally may be responsible of the fastcgi process and start it for you if necessary. Just add the "start-executable" init parameter :

```
<servlet>
  <servlet-name>FastCGI</servlet-name>
  <servlet-class>net.jr.fastcgi.FastCGIServlet</servlet-class>
  <init-param>
    <param-name>server-address</param-name>
    <param-value>localhost:6666</param-value>
  </init-param>
  <init-param>
    <param-name>start-executable</param-name>
    <param-value>c:/wamp/bin/php/php5.2.6/php-cgi.exe -b6666</param-value>
  </init-param>
</servlet>
```

Using several connections for the same app

If the fastcgi application accepts it, several host/port couples may be used to serve the pages. The FastCGI servlet will choose one of the provided address, and use the connection for serving the page.

This configuration is done by providing a comma-separated list of addresses to the servlet configuration :

```
<servlet>
  <servlet-name>FastCGI</servlet-name>
  <servlet-class>net.jr.fastcgi.FastCGIServlet</servlet-class>
  <init-param>
    <param-name>cluster-addresses</param-name>
    <param-value>localhost:6666;host1:6666;host2:6666</param-value>
  </init-param>
</servlet>
```

by default, the fastcgi connection that will be used to build a response is chosen randomly when a request arrives.

Doing fancy things : writing a custom handler :

You may provide a more complex algorithm for creating and keeping the fastcgi tcp connections, the only thing you need to do is writing your own class, that implements `net.jr.fastcgi.ConnectionFactory` :

```
/**
 * interface that any service that can create / destroy connections to a fastcgi
 * provider should implement.
 *
 * @author jrialland
 *
 */
public interface ConnectionFactory {

    /**
     * Called when a connection is needed.
     *
     * @return
     */
    public Socket getConnection();

    /**
     * Called when a connection is released (not needed anymore)
     *
     * Note : it doesn't mean that the socket should be closed at all, but
     * notifies that this connection is no more
     * needed for a particular request.
     * For example, a pooling system could use this method to mark connection
     * as "useable" for another request.
     *
     * @param socket
     */
    public void releaseConnection(Socket socket);
}
```

When it's all done, just tell the servlet :

```
<servlet>
  <servlet-name>FastCGI</servlet-name>
  <servlet-class>net.jr.fastcgi.FastCGIServlet</servlet-class>
```

```

        <init-param>
            <param-name>connection-factory</param-name>
            <param-value>com.mycompany.fascgi.MyFancyConnectionFactory</param-value>
        </init-param>
    </servlet>

```

Using Spring instead of servlets

pre-requisite :

You know how to use Spring, and in particular Spring's [DispatcherServlet](#)

If you use spring, you may register a handler instead of the servlet, in particular if you use a complex configuration.

The configuration is done through spring :

```

<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <value>
            **.php=fastCGIRequestHandler
        </value>
    </property>
</bean>

<bean id="fastCGIRequestHandler" class="net.jr.fastcgi.spring.RequestHandler">
    <property name="connectionFactory" ref="connectionFactory" />
</bean>

<bean id="connectionFactory"
class="net.jr.fastcgi.impl.SingleConnectionFactory">
    <constructor-arg value="localhost:9763"/>
</bean>

```

Note that the SingleConnectionFactory can be replaced with a PooledConnectionFactory, or any class that implements net.jr.fastcgi.ConnectionFactory.

Using portlets :

The configuration works just like it does with servlet. Just declare the portlet in your portlet.xml,

```

<portlet>
    <portlet-name>sample_fastci_app</portlet-name>
    <display-name>Sample FastCGI Application</display-name>
    <portlet-class>net.jr.fastcgi.FastCGIPortlet</portlet-class>
    <init-param>
        <name>server-address</name>
        <value>localhost:6666</value>
    </init-param>
    <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>view</portlet-mode>
    </supports>
</portlet>

```

```
</supports>
<portlet-info>
  <title>My portlet</title>
</portlet-info>
</portlet>
```