BeginnerDefaultPort

Eliminate the port from the URL(cookbook)

- TARGET-AUDIENCE: ***beginner*** advanced expert
- COCOON-RELEASES: 2.0.3, 2.0.4
- DOCUMENT-STATUS: ***draft*** reviewed released

What you will get from this page

I will show you how you can setup tomcat and apache in order to serve your cocoon application transparently from your webserver. I will use the jk comunication protocol, because this protocol is the default used in tomcat-4.1.* and it is supported down to ???

Your basic skills

- basic knowledge about apache administration
- basic knowledge about tomcat administration

Technical prerequisites

- You have read the page BeginnerDefaultContext
- You have enabled your webserver to use the mod_jk module.

If you don't know, whether your webserver is setup with mod_jk please take a look at CocoonAndApache and AddMod_jk(in German) While the first link focusses on technical issues, the secnd link is more a cookbook approach, but at the moment it is completely written in german langage. I will translate this page and add it to the CocoonCompetenceCenter in a few days.

Links to other information sources

...

The trivial solution

If you don't bother with peformance ad you don't want to make things too complex at the beginning you can simply run tomcat on port 80 instead of the default port 8080. If you do this, you can access tomcat as if it where the default HTTP server. I.e. the URL `http://mycompany` would already be served by tomcat and if you already followed the approach in BeginnerDefaultContext you are done.

But this is not the recommended approach, because

1. tomcat is said to be far less performant as the apache-webserer for standard file serving.
2. most probably your webserver/intranet server already is up and running on the default port, so you don't even have the chance to get hands on port 80 for our tomcat.
3. If port 80 is not occupied you can use it for now, but you definitely cut off the possibility to run your webserver under the now occupied default port.

Because all of this you may need to read on ...

Another trivial (and sufficient) solution: Using Apache's mod_proxy

For those who do not want to take the effort of using Apache's mod_jk as described in the next section (at least I could not get it to work) I recommend another solution. Using mod_proxy: It is really simple and pretty much does job, as far as I understand it (see also CocoonAndApache).

Simply add the following to lines to you Apache 'conf' file.
`=== ===ProxyPass / http://localhost:8080/`
`=== ===ProxyPassReverse / http://localhost:8080/`

As a result, Apache will forward all requests for `/` to `localhost:8080` and vice versa.
If you think doing it the proxy_way is not appropriate for your needs go ahead to the next section and use mod_jk.

Setting up a comunication between tomcat and apache

The elimination of the port from the URL is more tricky, because now the Webserver comes into play. I assume, your webserver is already running under the standard HTTP port 80. So that your company site can be accessed via `http://mycompany`

Now what you have to do is to setup a comunication line between the webserver and tomcat. You can read the very comprehensive doc about this issue from apache. If you prefer to just set it up and don't bother, go on with this text. But be warned. I won't tell you anything about the concepts. If you fail, you have to go to the official docs!

Preparing tomcat

- Go to the `$tomcatroot/conf` folder and edit server.xml
- Search for the definition of the Ajp12 Connector.
- Add a new connector using following snippet:

```
<Connector className="org.apache.tomcat.service.PoolTcpConnector">
  <Parameter name="handler"
             value="org.apache.tomcat.service.connector.Ajp13ConnectionHandler"/>
  <Parameter name="port" value="8090"/>
</Connector>
```

  The value of the port is arbitrary. You only need to enshure, the port is not used elsewhere on your computer.

create `mod_jk.conf` and `worker.properties`

You can let tomcat create these files for you. I prefer to give you two sample files here, which you can cut/paste into your environment. These files are furnitured to be used with cocoon. Both files will be placed into your $apacheroot/conf folder:

sample mod_jk.conf

The following snippet is the smallest possible definition. Please check all pathes and modify them for your environment:

```
<IfModule mod_jk.c>
  JkWorkersFile /etc/httpd/conf/workers.properties
  JkLogFile     /var/log/httpd/mod_jk.log
  JkLogLevel error
  JkMount /cocoon   cocoon
  JkMount /cocoon/  cocoon
  JkMount /cocoon/*  cocoon
</IfModule>
```

Note, that with this definition all requests to /cocoon will be served fully from cocoon. You may want to add rules to serve static content from the server instead. If you want to know, how this works, please have a look at ???

sample worker.properties Please check all pathes and modify them for your environment:

```
workers.tomcat_home=/opt/tomcat/tomcat-4.1.18
workers.java_home=/opt/java/java-1.3.1
worker.list=cocoon
worker.cocoon.port=8090
worker.cocoon.host=localhost
worker.cocoon.type=ajp13
```

Note: You can define as many workers as you like. But don't forget to place them into the comma separated property `worker.list`. Now you can mount different URL-spaces to different tomcat servers...

what to do if things go wrong

1.If your webserver does not come up, check the error logs of your webserver. You will get hints to what may go wrong.
1.If you get unexpected results while accessing your cocoon via your webserver, check that you really have specified the corect JkMount points.
1.If you get tons of error messages concerning illegal protocol errors, check that you reallyhave connected the to the tcp connector of your tomcat. Common error is to connect tp the HTTP-connector instead.
1.If still things go worng, you need to consult the apache and tomcat docs.

---

page metadata

- AUTHOR:DabbousBR
- AUTHOR-CONTACT: hussayn.dabbous@saxess.deBR
- REVIEWED-BY:BR
- REVIEWER-CONTACT:BR