BeginnerAdvancedWebappOrganisation

Organising your development (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 a cleanly separated development environment with minimal modifications of the existing Cocoon infrastructure. You will end up with a hierarchy of folders all completely under your control with only one modification in the Cocoon standard distribution.

Your basic skills

- You have basic knowledge about XML
- You know how to access a command line shell to your OS
- I assume, you also have read BeginnerSimpleWebappOrganisation.

Technical prerequisites

- You need a cleanly installed version of Cocoon.

If you don't have access to Cocoon right now, you can just install it on your local computer. Please refer to BeginnerInstallation for further information.

Links to other information sources

...

Separating your work folder from Cocoon

If you want to take out the whole work folder from the $cocoon-root, this can be done by specifying a mountpoint similar to what we described in part I of BeginnerSimpleWebappOrganisation. you will add a new pipeline within the $cocoon_root/sitemap.xmap file. Here is what you will have to do:

- Move to the $cocoon_root folder
- Edit the original sitemap.xmap in $cocoon_root:
  - Search for the line with <map:pipelines> . This is the start tag of the pipelines bracket near line 530.
  - Add following code snippet right AFTER the starttag <pipelines> and save your modifications:

```
<map:pipeline>
  <map:match pattern="work/**">
    <map:mount check-reload="yes"
               reload-method="synchron"
               src="/work/sitemap.xmap"
               uri-prefix="work"/>
  </map:match>
</map:pipeline>
```

- Now the src for the mount is an absolute path. By this you can place your actual workfolder at an arbitrary place within your filesystem. Create your work folder exactly at the location you have specified in the src attribute and proceed as described in Part II above. In the example snippet above i have choosen the location "/work" (see the src-attribute of the <map:mount> tag) although the folder name is arbitrary.

Now you have cleanly separated your work from the cocoon distribution. Your "work" folder may be located anywhere in your file system. Your URL's to your work will look like this:

http://localhost:8080/cocoon/work/index.html

Setting up a generic mount point outside of Cocoon

If you happen to need more than one external work folder, you can set up a generic sitemap similar to Part I above. I assume, you want to put some cocoon applications into subfolders of /work, e.g. /work/coconuts and /work/cocobolts. Then you want to adress these applications separately with URL's like

```
http://localhost:8080/cocoon/work/coconuts
http://localhost:8080/cocoon/work/cocobolts
```

Also this can be achieved with a slight modification of your pipeline. Go again to $cocoon_root/sitemap.xmap and modify your pipeline as follow:

```
<map:pipeline>
  <map:match pattern="work/*/**">
    <map:mount check-reload="yes"
               reload-method="synchron"
               src="/work/{1}/sitemap.xmap"
               uri-prefix="work/{1}"/>
  </map:match>
</map:pipeline>
```

Note how this sitemap now resembles the very first code snippet for the cocoon internal "mount" points. We only had to specify an absolute path and we are done ...

Hiding /cocoon from the URL

You may want to get rid of the /cocoon within the URLs to your cocoon application. I.e. instead of typing the URL:

http://mycompany:8080/cocoon/myapp

you want the URL:

http://mycompany/

to trigger cocoon to send an html index page back to the browser. This issue is worthwhile to create a separate page. Please look at BeginnerDefaultContext or CocoonEasyInstallation for further reading.

---

page metadata

- AUTHOR: Dabbous

- AUTHOR-CONTACT: hussayn.dabbous@saxess.deBR
- REVIEWED-BY: DerekH (minor editing)

- REVIEWER-CONTACT:BR