BlocksUseCases

Dream on! What would you like Blocks to be? – BertrandDelacretaz

Feel free to add more use-cases to this page, or to comment on existing ones!

Blocks Use Case #1: apt-get like installation of new functionality

1. User heard about a new FunkySerializer Block for Cocoon and wants to install it.
2. User navigates to the BlocksInstaller.html page (standard part of the minimal Cocoon installation).
3. User enters the name of the *FunkySerializer* block on the BlocksInstaller form, and requests installation by submitting the form.
4. Cocoon retrieves the FunkySerializer information page from the download site, and shows it to the user, asking for confirmation before installing the Block.
5. Cocoon retrieves the FunkySerializer block archive, lets FunkySerializer ask a few questions through a couple of web forms, updates its configuration (jars, xconf, sitemaps, etc.) and activates the newly downloaded Block.
6. The BlocksInstaller page also allows the user to find out which versions of which Blocks are currently installed, and to update or remove them as needed.

I'm not saying I will implement this tomorrow, but if Blocks could make this possible it would be great! – BertrandDelacretaz

Comments

- On cocoon-dev, Carsten Ziegeler says: *...I think, what you describe is more a 'module feature' and not so much the 'big blocks concept' - although your wish list should be possible with the blocks concept...*

Blocks Use Case #2: Skin/Webmail example

Excerpted from Giacomo Pati's cocoon-dev message, full text available here. COB means 'Cocoon Block' below.

...Consider the following:

1. There is a COB interface defined about skinning application data

of defined XSchemas (navigations, menus, content) by transformations (i.e. XSLT) into something else (i.e. HTML)

2. There is a COB interface defined about accessing mail messages which generates defined XSchemas of mail messages and mail folders from mail stores like IMAP, POP, mboxes or "write your own"

3. There is a COB which offers an WebMail application which depends on

- a skinning COB
- a mail message accessing COB

Now, when someone likes to deploy the WebMailCOB it uses the Cocoon COB management application which asks for a URI to the COB to deploy. As a COB will have meta data inside, the management console will be able to ask for configuration data for the COB and also knows of unresolved dependancies to other COBs. So the deployer of the WebMailCOB will be asked where the dependant skin and access COBs are. Now, the management console could offer implementations for them from a well known site like cob.apache.org (URL will be configurable of course) which have been registered there. Now you can select the Christmas skin COB and the IMAP access COB from the list and configure them to have your WebMail ready and running.

And now you will be free to use other skin COBs in terms of minutes or you can write your own to achieve "corporate identity". You have your mails stored in a database and there is no COB for it? Write your own and have them be served by the WebMailCOB.

(...continues, see full text from above link)

Blocks Use Case #3:

I want use a block to convert a file, which is used for my own components. For example in my vision, I have a Chaperon block, which offers a parser transformer, which parses a text within a xml document, and a text generator, which produce a SAX stream.

I also want to offer with a Chaperon block a pipeline, which converts a grammar as text format to xml version of the grammar.

myblock.block

```
<map:transform type="parser" src="chaperon:/mygrammar.txt"/>
```

chaperon.block

```
<map:pipeline>
 <map:match pattern="*.txt">
  [transform txt to xml]
 </map:match>
</map:pipeline>
```

But the problem is that the chaperon block need access to the files in myblock to transform them. Is there any possible way to do this??!

Stephan Michels.

Blocks Use Case #4:

*This might be waiting for **you** to write it!*