Blockathon2005Report

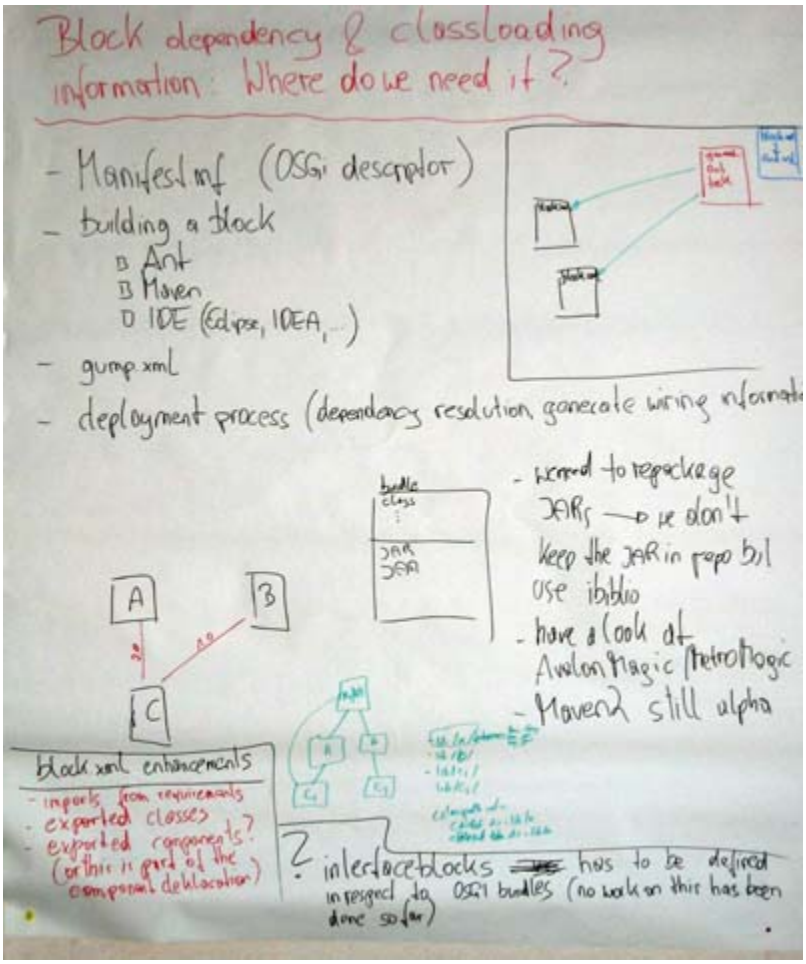
Info and reports about the ApacheCon 2005 [Blockathon].

See also osgi.

Tuesday, June 19th, evening

The template block has been hacked by Reinhard to work as an OSGI bundle, and Sylvain has been working on a component to bridge the OSGI and ECM++ component managers. Stay tuned...

Tuesday, June 19th, afternoon



Role of block.xml

Block metadata information will be used for the following purposes:

- Manifest information, when creating bundles
- Block build scripts for ant, Maven, Eclipse, IDEA, etc.
- Inclusion in gump.xml
- Deployement process: dependency resolution, wiring information

We have discussed several options for generating this metadata, and using our own block.xml seems to be the safest option for now. The other "views" on this metadata will be generated a build time.

Repackaging of external jars as OSGI bundles

It seems to make sense to repackage external jars (xalan, fop, for example) as individual OSGI bundles, based on the dependencies discovered at build time.

The cleanest solution seems to be the downloading of these external jars from existing repositories (ibiblio), using the appropriate ant tasks.

The downloaded jars will then be packaged as OSGI bundles for deployment, maybe exporting all the packages that they contain be default.

Compile-time access to interfaces defined in bundles

When compiling a bundle, there must be a way to get access to interfaces defined in other bundles that we depend on.

The brute force solution that comes to mind is:

- Download the bundle
- Explode the bundle jar and all jars that are found in it
- Delete all resulting class files, except the ones that the bundle exports
- Put all the remaining files in the classpath

But there is probably a better way, or existing tools in the OSGI projects.

Interface bundles

We should create individual bundles for groups of interfaces that represent contracts between the Cocoon core and the blocks.

To develop a sitemap block, for example, one should be able to take a single bundle containing the relevant interfaces, and only these (and the relevant documentation, see below).

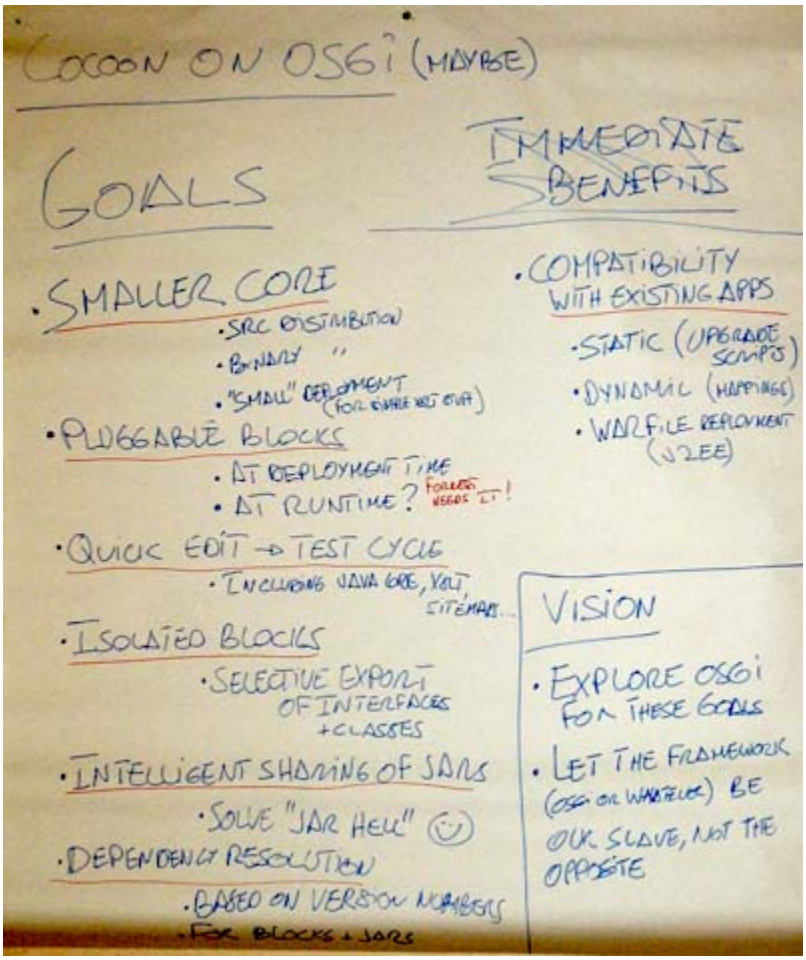
Documentation in bundles

The OSGI spec provides for documentation in the OSGI-OPT directory of a bundle, this would be a nice way of making the bundles/blocks standalone components.

Tuesday July 19th, noon

Agreement on a list of high-level goals which are independent from OSGI.

Posted to the dev@ list, see http://marc.theaimsgroup.com/?l=xml-cocoon-dev&m=112177131522853&w=2



(the crossed out "immediate benefits" don't mean there are no immediate benefits - we crossed this section out thinking that the benefits are evident, and we wanted to have lunch at that point 😉

Tuesday July 19th, morning

Talk about blocks management led by Daniel.

Tasks for today

- Upayavira: making Cocoon OSGI bundles run with Oscar 2.0alpha
- Sylvain: will work on the ECM++ stuff (core container level)
- Daniel: will work on the ECM++ stuff (block level)
- Daniel: will work on a talk about sitemap block integration
- Reinhard: make another block an OSGI bundle (probably the XML templating block)
- Reinhard: trigger the discussion about block deployment (especially we should find usecases; how to we)
- Bertrand: will trigger a discussion about Block infrastructure in general (how to package blocks? e.g. have a

Some package names need to be changed

The OSGI mechanisms for import/export of classes are based on package names, a bundle must export a complete package, it cannot export a single class.

This means we have a problem with our current package names, for example many blocks have classes in o.a.c.generation or o.a.c.transformation, that need to be exported. The package names will need to be changed, for example to o.a.c.blocks.html.HtmlTransformer.

ECM-based blocks

Existing blocks using the ECM++ component manager internally will need to make their services available to the OSGI framework. This can be done when the ECM component manager is being setup, maybe by creating an ECM event listener which "relays" service availability info to OSGI.

We might want to make only some ECM components available as OSGI services.

Scenario:

1. Bundle is loaded by OSGI
2. BundleActivator initializes ECM ComponentManager, uses a Listener to learn about ECM Services being registered and registers (some of) them as OSGI services

ECM++ bundle

As several blocks will be using ECM++ it makes sense to have an ECM++ bundle to make this shared code available.

Blocks using other component managers

In the same way, blocks using Spring, hivemind or whatever other component manager can make some of their components available as OSGI services.

The same pattern (component manager initialization - service instantiation listener - OSGI service registration) seems to be applicable in these cases.

Monday July 18th - evening

The Cocoon trunk is running under OSGI! See http://marc.theaimsgroup.com/?l=xml-cocoon-dev&m=112170290723962&w=2

On the picture you see Reinhard in front and Daniel behind him - these guys have been making it happen this afternoon!

Monday July 18th - morning

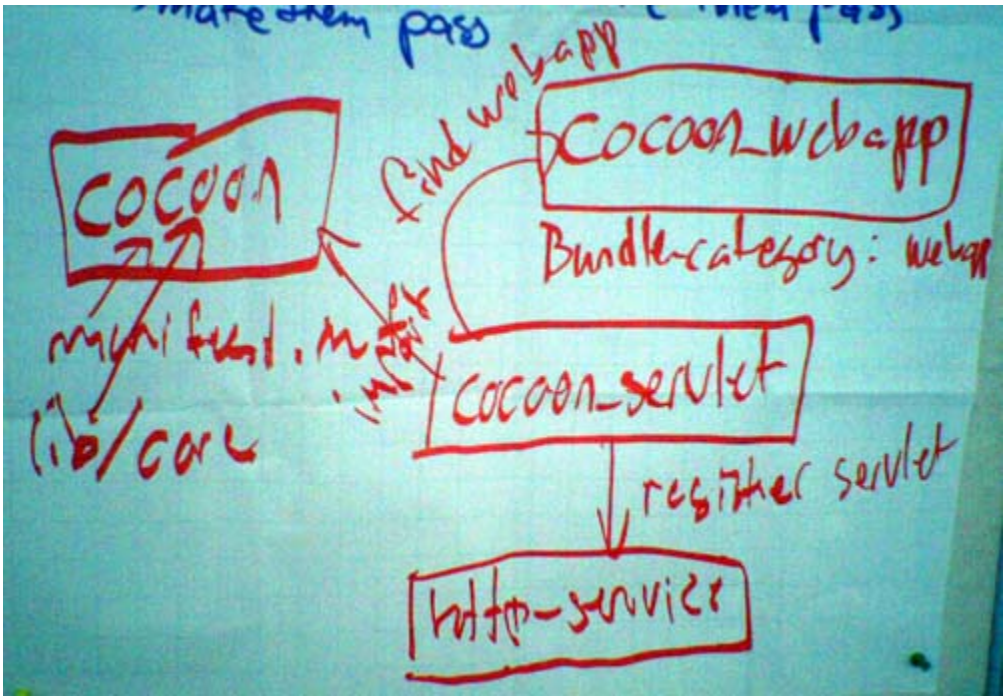
Daniel talks about the current state and next steps.

In the meantime, Sylvain is finishing the first release of Lepido to showcase it during Gianugo's Cocoon tutorial

Current state of the Cocoon OSGI stuff

The whiteboard/osgi code is being integrated into the trunk, to make it more convenient to work on.

Currently the OSGI-related code consists of three bundles:



The cocoon bundle:

- Contains all the code from src/java and all the libraries from lib/core
- Currently all packages are exported (to make it easier to work now) via an OSGI manifest file
- Does not include an OSGI Activator class

The webapp bundle:

- Contains all the files from src/webapp
- Bundle-category: webapp
- Passive bundle, only contains resources

The cocoon-servlet bundle:

When loaded:

- Imports some classes from the Cocoon bundle (via an Activator class)
- Gets resources from the webapp bundle, via the OSGI bundle: protocol
- Creates the Cocoon Servlet
- Registers the Servlet with the OSGI HttpService

Main TODOs:

- Package some or all libraries in their own bundles
- Directory listing (e.g. for DirectoryGenerator) doesn't work in OSGI currently, throws Exceptions. Need to be converted to warnings at least, to avoid the need for patching the xconf files to run under OSGI (See http://marc.theaimsgroup.com/?l=xml-cocoon-dev&m=112167452021559&w=2) .

Next steps

1. Get the OSGI-based Cocoon Servlet to work in the trunk
2. Remove the need to patch the xconf files
3. Create a bundle for one existing block or create an "OSGI test" block
4. Demonstrate the dynamic loading of the block bundle via an URL