InterceptedFlowscript

(this document is adapted from the message posted by stefanomazzocchi on the cocoon development mailing list)

While writing flowscripts, you realize how many things can be applied to many of the various flowscript functions that are called by the sitemap. In Linotype, I ended up using the ability that javascript gives you of using functions as native objects and then invoquing them by passing a modified argument vector.

It came to me that what I did with linotype was a really-poor-man interception mechanism. Interception is the ability to "intercept" a method call and execute some code before or after the execution of the intercepted method.

Interception is the simplest form of the aspect orientation.

Adding interception mechanism to the flowscript requires three changes, two in the FOM, one (more important!) in the javascript syntax:

- the addition of a function call to the "cocoon" object, which indicates that intercepting function calls should be imported.

```
cocoon.apply("blah.js");
```

where "blah.js" contains the intercepting functions.

- the addition of a function call to context that continues the intercepted invocation:

```
 ...
 continueExecution(arguments);
 ...
```

- the introduction of wildcars for function names.

```
function get*() {
  ...
}
```

the above seems rather accademic, so allow me to write an example where the use of aspect-oriented flowscript would shine.

Something that can be modelled very well as an aspect is authentication/authorization (aka login) because it's not something that is part of the webapp (at least, it shouldn't be!) but it's something that is "wrapped" around it to allow people to access it without the proper authorization.

With interception, it can be implemented as such

```
login.js

   var user;
   var password;
   var role;

   function *() {
     var userManager = cocoon.getComponent("UserManager");
     while (true) {
       sendPageAndWait("login", { user : user, password : password });
       user = cocoon.request.user;
       password = cocoon.request.password;
       if (userManager.isValid(user, password)) {
          role = userManager.getRole(user);
          continueExecution(arguments);
          break;
       }
     }
```

then, in your flowscript, you simply have to call the "apply" function to apply the aspects to your flowscript

```
yourstuff.js

    cocoon.apply("login.js");

    function whatever(blah) {
      doSomething(blah);
    }
```