

Customizing Manakin (pt. 2)

1. Introduction

| | |
|--------------------------------|---|
| HTML / CSS style Tier | 2 |
| XML / XSL Theme Tier | 2 |
| Java / Cocoon development Tier | 2 |

2. Java / Cocoon development Tier

| | |
|------------------------------|---|
| Aspect Oriented Programming | 2 |
| Manakin Aspects | 2 |
| Example Shopping Cart Aspect | 4 |

3. Exercises

| | |
|---|---|
| Exercise 1: Enabling / Disabling existing aspects | 5 |
| Exercise 2: Editing existing aspects | 5 |
| Exercise 3: Adding a new page to an aspect | 5 |

1. Introduction

The component architecture of Manakin consists of three tiers:

- HTML / CSS style Tier
- XML / XSL Theme Tier
- Java / Cocoon development Tier

As the tiers descend, progressively more skill and experience are required of someone to be productive at that tier.

HTML / CSS style Tier

Requires only basic XHTML knowledge and can change the style of how information is presented to the user across an entire Theme.

XML / XSL Theme Tier

Requires XML and XSLT knowledge, but no Java experience. At this tier; information currently being displayed by DSpace can be further processed before being displayed to the user. Major site-wide presentation, structural, and limited computational changes may be made to an entire Theme, or an individual page.

Java / Cocoon development Tier

Requires Java development and Cocoon expertise and is able to perform any functional customization to DSpace using the supplied component architecture.

This session will focus on the Java / Cocoon development tier.

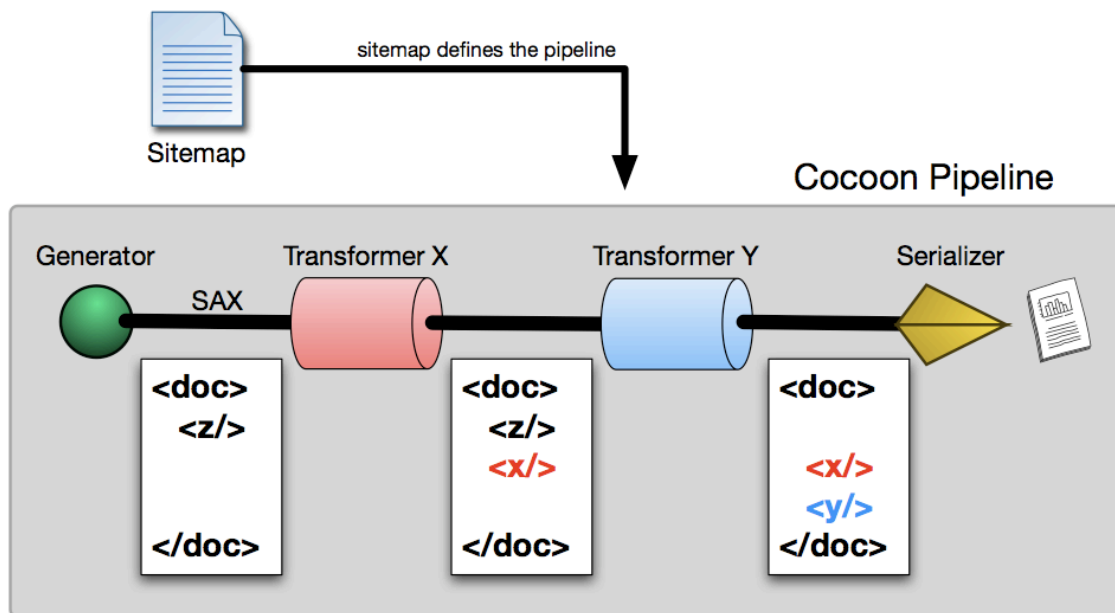
2. Java / Cocoon development Tier

Aspect Oriented Programming

Aspect-oriented programming (AOP) is a programming paradigm that increases modularity by allowing the separation of cross-cutting concerns, forming a basis for aspect-oriented software development. This entails breaking down a program into distinct parts (cohesive areas of functionality). Logging is a common example of a crosscutting concern because a logging strategy necessarily affects every single logged part of the system. Logging thereby crosscuts all logged classes and methods. In Aspect-Oriented Programming programs are broken down into distinct parts that overlap as little as possible. In Manakin these parts are called aspects, and woven together to form the program.

Manakin Aspects

Manakin Aspects are the arrangement of Cocoon components (transformers, actions, matchers, etc) that implement a new set of coupled features for the system. These Aspects combine to form all the features of Manakin. Each of the system's Aspects are "chained together", so that for each page the system generates, every Aspect is given the chance to add its own content into the page. Aspect chaining allows new features to be overlaid onto an existing system while eliminating the need to patch or merge files, because all Aspects are kept structurally separate.



Aspects are implemented as Cocoon sub-sitemaps composed of components which may query the DSpace API, possibly changing the state of DSpace. They take a DRI document as input, incorporate their features into the document, and pass the modified DRI document along to the next Aspect; this use of DRI documents as both input and output is what makes Aspect chaining possible. In addition to modifying the in-process document for display, Aspects execute the code that determines how the page is displayed.

The order in which Aspects are executed is determined by the Aspect configuration file, `xmlui.xconf`. Aspects are packaged together with their source code, sitemaps, unique translations for the particular Aspect, and any other resources that may be required. This packaging enables portability of Aspects; they can be copied from one DSpace installation to another with minimal conflicts.

The standard installation of Manakin includes five aspects:

- **Artifact Browser**

- The ArtifactBrowser Aspect is responsible for browsing communities, collections, items, and bitstreams, viewing an individual item, and searching the repository.

- **E-person**

- The E-person Aspect is responsible for logging in, logging out, new user registration, forgotten passwords, editing profiles, and changing passwords.

- **Submission**

- The Submission Aspect is responsible for submitting new items to DSpace, the workflow process, and finally ingesting the new items into the DSpace repository.

- **Administrative**

- The Administrative Aspect is responsible for administrating DSpace, such as creating, modifying, and removing all communities, collections, e-persons, groups, registries, and authorizations.

- **XMLTest**

- The XMLTest Aspect is included as a guide to show how to extend Manakin for local customizations without modifying the standard Manakin Aspects.

Example Shopping Cart Aspect

This example is based on an example from the “Manakin Developer’s Guide” by Scott Phillips, Cody Green, John Leggett, Alexey Maslov, Adam Mikeal, Brian Surratt ¹.

To gain a fuller understanding of Aspects’ interaction with Manakin, consider the example of creating an Aspect to add a shopping cart to DSpace. This new shopping cart should provide several new features to the system, including the ability to add an item to the user’s cart, the ability to manage the cart (adding or removing items), and the ability to purchase and grant access to the item(s).

This new Aspect and all associated files, including the source code, configuration, cocoon sitemap, and any static resources, would be contained in its own directory. When this new Aspect is added to the `xmlui.xconf` configuration file, any page served by Manakin will be passed through it, and depending upon the URL of the request, one of the following cases will occur: specific existing pages could be modified, content could be added to all pages, or entirely new pages could be created.

- The first case, modifying existing pages, is encountered when a standard item display page is being generated. The new Aspect would likely add a “Buy This Item” button to this page. This new button is added to the page content already generated by the standard `ArtifactBrowser` Aspect. In this case, the shopping cart is extending one of the standard Manakin Aspects.
- The second case, adding content to all pages, is encountered because once the user has added an item to their shopping cart and is continuing to browse the site, they will need the ability to navigate back to their cart. Since this possibility could occur anywhere within the site, the Aspect needs to add a “View Cart” link to all pages.
- The third case, creating new pages, will occur when the user wants to checkout or review their cart’s contents. After the user clicks the “View Cart” link, they are taken to a new page and shown the contents of their shopping cart. This page did not exist in the system before the Aspect was added; it’s entirely new content generated by the shopping cart Aspect. This case handles the actions that need to be performed on the server, such as adding or removing items from the cart, purchasing the cart, and finally enabling access to the content.

The shopping cart Aspect only deals with the features necessary to implement a shopping cart, so when the Aspect is turned off, there would be no indication of the shopping cart anywhere: no dead links, no dead pages, etc. However, when the Aspect is turned on, the links to the shopping cart, the “Buy This Item” button, and checkout pages would all be present without necessitating the merging or patching of any files.

¹ <http://di.tamu.edu/projects/xmlui/resources/DevelopersGuide.pdf>

3. Exercises

Exercise 1: Enabling / Disabling existing aspects

In the `xmlui.xconf` file, enable the XMLTest aspect, and look at the changes in the navigation, and the new pages which have been created. These pages are especially useful for theming your repository.

Disable the Submission aspect, and notice that not only the “Submissions” navigation has disappeared. Also the collection homepage doesn’t contain a link to “Submit a new item to this collection” when logged in anymore.

Don’t forget to re-enable the Submission aspect afterwards.

Exercise 2: Editing existing aspects

Find the sitemap related to the submission aspect, and edit it to remove the “Submit a new item to this collection” handler in order to disable the page behind this link. This should still display the link, but you should receive a “Page not found” error when clicking the link.

Now also remove the link to “Submit a new item to this collection” from the collection homepage so it cannot be clicked anymore.

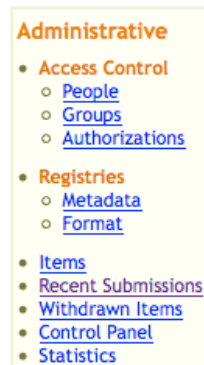
Exercise 3: Adding a new page to an aspect

This is a more advanced exercise where you need to add a new page called “Recent Submissions” to the DSpace Administrative aspect. First of all you will need to add a “Recent Submissions link to the Administrative navigation as shown in the image to the right.

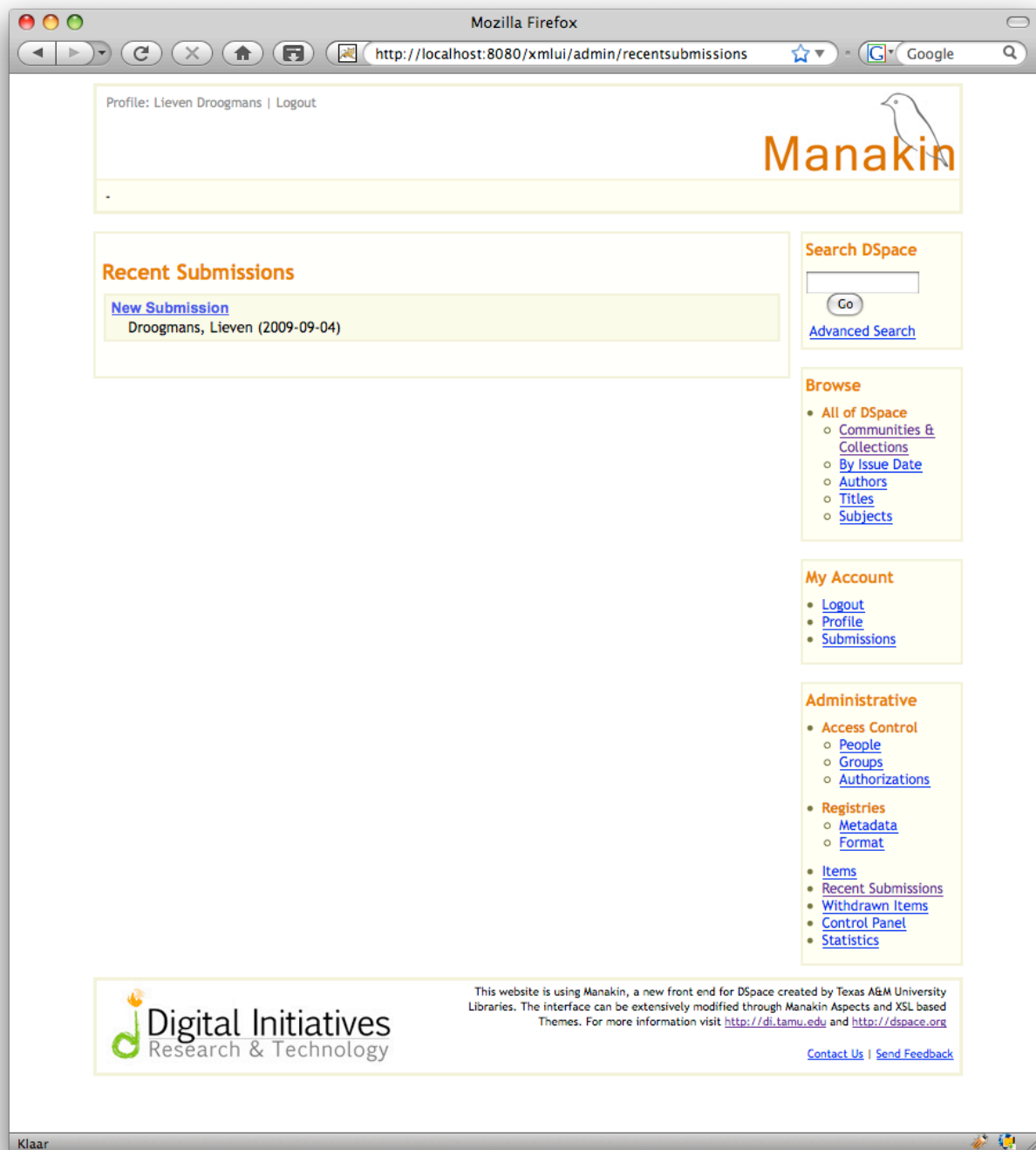
The “Recent Submissions” link should point to a new page containing the recent submission in the repository. If the page appears to be empty at first, please make sure you have submitted at least one item.

The Recent submissions page should look as the screenshot below (if using the Reference Theme).

Hint: Java code to get the recent submissions.



```
private java.util.List<BrowseItem> getRecentlySubmittedItems() throws SQLException
{
    String source = ConfigurationManager.getProperty("recent.submissions.sort-option");
    BrowserScope scope = new BrowserScope(context);
    scope.setResultsPerPage(RECENT_SUBMISSIONS);
    try
    {
        scope.setBrowseIndex(BrowseIndex.getItemBrowseIndex());
        for (SortOption so : SortOption.getSortOptions())
        {
            if (so.getName().equals(source))
            {
                scope.setSortBy(so.getNumber());
                scope.setOrder(SortOption.DESENDING);
            }
        }
        BrowseEngine be = new BrowseEngine(context);
        return be.browse(scope).getResults();
    }
    catch (SortException se) { log.error("Caught SortException", se); }
    catch (BrowseException bex) { log.error("Caught BrowseException", bex); }
    return new ArrayList<BrowseItem>();
}
```



TERMS OF USE

PLEASE READ THESE TERMS OF USE CAREFULLY BEFORE USING THESE COURSE MATERIALS. By using these course materials, you signify your assent to these terms of use. If you do not agree to these terms of use, please do not use these course materials.

RESTRICTIONS ON USE OF MATERIALS. These course materials are owned by @mire NV, Technologielaan 9, 3001 Heverlee (Belgium).

No components from these course materials owned, licensed or controlled by @mire NV may be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, except that you may download one copy of the materials on any single computer for your personal, non-commercial home use only, provided you keep intact all copyright and other proprietary notices.

Modification of the materials or use of the materials for any other purpose is a violation of @mire's copyright and other proprietary rights. The use of any such material on any other web site or networked computer environment is prohibited.

To request permission to reproduce materials,
call +32 2 888 29 56,
email info@atmire.com,
or write to @mire NV, Technologielaan 9, 3001 Heverlee, Belgium.