The Manakin Architecture

1. DSpace architecture 2

2. Digital Repository Interface (DRI schema) 2

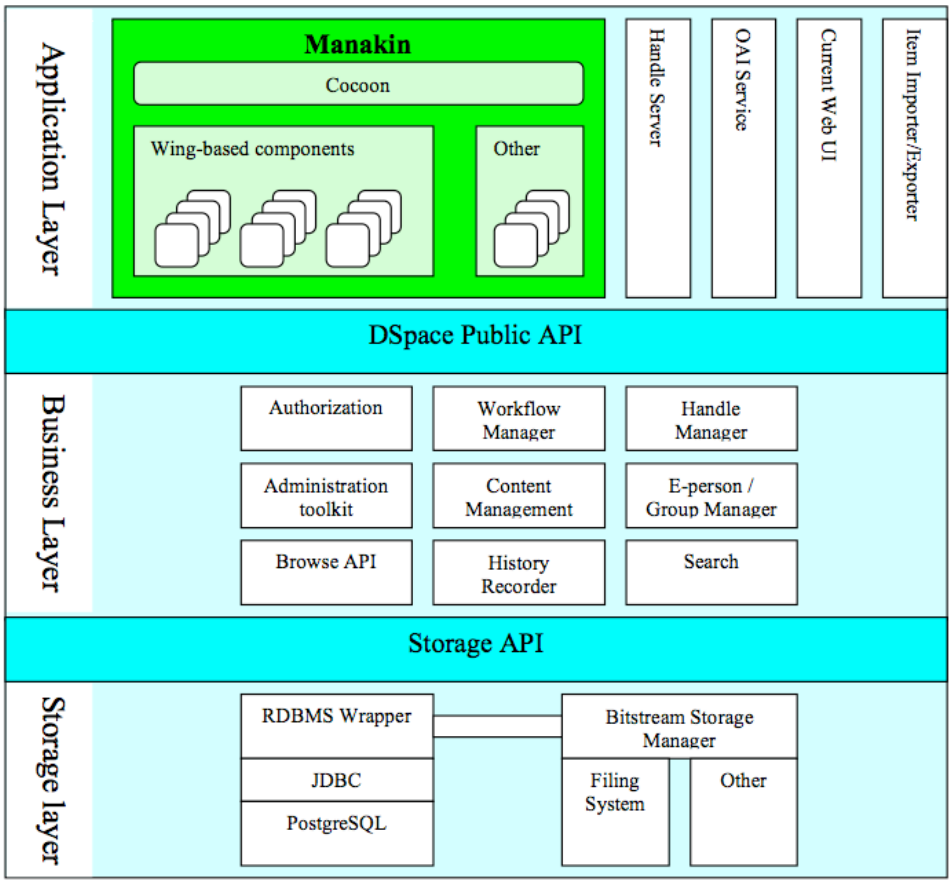
- Introduction 2
- Purpose 3
- DRI Structure 3
  - Important Elements 3
  - Full List of DRI Elements 5
- Localization and Internationalization 5

3. Architecture Overview 6

- Aspects 6
- Merging DRI Documents 7
- Themes 7
- Processing Order 8
- Configuration 8
  - xmlui.xconf 8
  - Configuring Aspects 9
  - Theme Transformer and Sitemap 11
- DSpace METS Metadata Representation 13
  - Default Sitemap Configuration 13
- DSpace sitemap 14

4. Reference 17

1. DSpace architecture



source: "Manakin Developer's Guide" Scott Phillips et al.[1]

The storage layer is responsible for physical storage of metadata and content. The business logic layer deals with managing the content of the archive, users of the archive (e-people), authorization, and workflow. The application layer contains components that communicate with the world outside of the individual DSpace installation, for example the Web user interface and the Open Archives Initiative protocol for metadata harvesting service.

The source code is organized to cohere very strictly to this three-layer architecture.
`org.dspace.app` → Application layer
`org.dspace` → Business logic layer (except storage and app)
`org.dspace.storage` → Storage layer

2. Digital Repository Interface (DRI schema)

Introduction

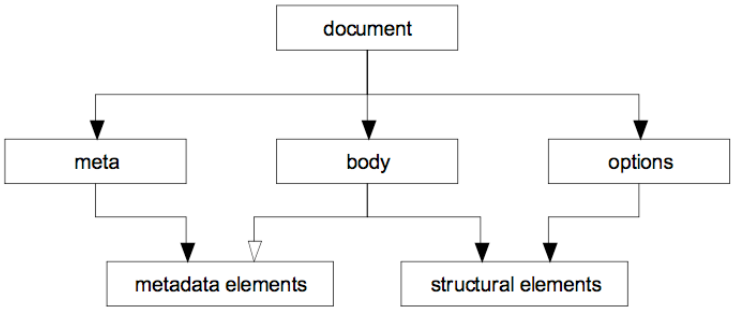
Digital Repository Interface (DRI) is a schema that governs the structure of a Manakin DSpace page when encoded as an XML Document. It determines what elements can be present in the Document and the relationship of those elements to each other. This reference document explains the purpose of DRI, provides a broad architectural overview, and explains common design patterns. The appendix includes a complete reference for elements used in the DRI Schema, a graphical representation of the element hierarchy, and a quick reference table of elements and attributes.

[1] http://di.tamu.edu/projects/xmlui/resources/DevelopersGuide.pdf

Purpose

DRI is a schema that governs the structure of the XML Document. It determines the elements that can be present in the Document and the relationship of those elements to each other. Since all Manakin components produce XML Documents that adhere to the DRI schema, The XML Document serves as the abstraction layer. Two such components, Themes and Aspects, are essential to the workings of Manakin and are described briefly in this manual. Additionally, the Manakin Developers Guide provides a more detailed overview of Aspects and other Manakin components.

DRI Structure



Important Elements

Document

The `<document>` element is the root for all DRI pages and contains all other elements. It bears only one attribute, version, that contains the version number of the DRI system and the schema used to validate the produced document. At the time of writing the working version number is 1.1. However it is reasonable to expect that this number will be incremented when future changes are made to the schema.

Meta

The `<meta>` element is the top-level element under document and contains all metadata information about the page, the user that requested it, and the repository it is used with. It contains no structural elements, instead being the only container of metadata elements in a DRI Document. The metadata stored by the meta element is broken up into three major groups: `userMeta`, `pageMeta`, and `repositoryMeta`, each storing metadata information about their respective component. Please refer to the reference entries for more information about these elements.

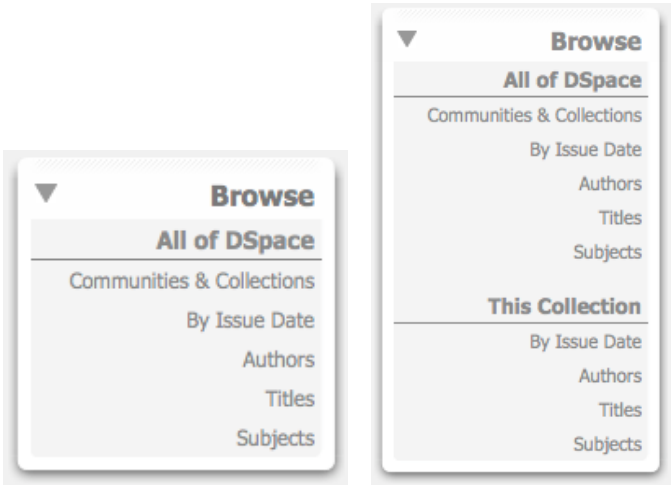
```
<meta>
<userMeta authenticated="no">
  <metadata element="identifier" qualifier="loginURL">/labs/login</metadata>
  <metadata element="language" qualifier="RFC3066">fr</metadata>
  <metadata element="language" qualifier="RFC3066">nl</metadata>
  <metadata element="language" qualifier="RFC3066">en_US</metadata>
  <metadata element="language" qualifier="RFC3066">en</metadata>
</userMeta>
<pageMeta>
  <metadata element="contextPath">/labs</metadata>
  <metadata element="request" qualifier="queryString"/>
  <metadata element="request" qualifier="scheme">http</metadata>
  <metadata element="request" qualifier="serverPort">80</metadata>
  <metadata element="request" qualifier="serverName">atmire.com</metadata>
  <metadata element="request" qualifier="URI"/>
  <metadata element="search" qualifier="simpleURL">/labs/search</metadata>
  <metadata element="search" qualifier="advancedURL">/labs/advanced-search</metadata>
  <metadata element="search" qualifier="queryField">query</metadata>
  <metadata element="page" qualifier="contactURL">/labs/contact</metadata>
```



```
  <metadata element="page" qualifier="feedbackURL">/labs/feedback</metadata>
  <metadata element="stylesheet" lang="infocon"
            qualifier="screen">lib/style-infocon.css</metadata>
</pageMeta>
<repositoryMeta>
  <repository repositoryID="123456789"
              url="/metadata/internal/repository/123456789/mets.xml"/>
</repositoryMeta>
</meta>
```

Options

The `<options>` element is another top-level element that contains all navigation and action options available to the user. The options are stored as items in list elements, broken up by the type of action they perform. Typical actions are: browsing, administration, or actions that are context dependent. Actions can contain sub-lists to such as the browse the repository versus browse this collection lists. The options element contains no metadata elements and can only make use of a small set of structural elements, namely the list element and its children.




```
<options>
<list n="browse" id="aspect.artifactbrowser.Navigation.list.browse">
<head>Browse</head>
       <list n="global" id="aspect.artifactbrowser.Navigation.list.global">
       <head>All of DSpace</head>
       <item>
              <xref target="/labs/community-list">Communities &amp; Collections</xref>
       </item>
       <item>
              <xref target="/labs/browse?type=dateissued">By Issue Date</xref>
       </item>
       <item>
              <xref target="/labs/browse?type=author">Authors</xref>
       </item>
       <item>
              <xref target="/labs/browse?type=title">Titles</xref>
       </item>
       <item>
              <xref target="/labs/browse?type=subject">Subjects</xref>
       </item>
       </list>
       <list n="context" id="aspect.artifactbrowser.Navigation.list.context"/>
</list>
</options>
```

Body

The last major top-level element is the `<body>` element. It contains all structural elements in a DRI Document, including the lists used by the options element. Structural elements are used to build a generic representation of a DSpace page. Any DSpace page can be represented with a combination of the

structural elements, which will in turn be transformed by the XSL templates into another format. This is the core mechanism that allows DSpace XML UI to apply uniform templates and styling rules to all DSpace pages and is the fundamental difference from the JSP approach.

```
<body>
<div rend="primary" n="comunity-browser"
     id="aspect.artifactbrowser.CommunityBrowser.div.comunity-browser">
  <head>Communities in DSpace</head>
  <p>Select a community to browse its collections.</p>
  <referenceSet rend="hierarchy" type="summaryList" n="community-browser"
           id="aspect.artifactbrowser.CommunityBrowser.referenceSet.community-browser">
    <reference repositoryID="123456789" type="DSpace Community"
               url="/metadata/handle/123456789/6080/mets.xml">
      <referenceSet type="summaryList">
        <reference repositoryID="123456789" type="DSpace Collection"
                   url="/metadata/handle/123456789/6484/mets.xml"/>
        <reference repositoryID="123456789" type="DSpace Collection"
                   url="/metadata/handle/123456789/6485/mets.xml"/>
        <reference repositoryID="123456789" type="DSpace Collection"
                   url="/metadata/handle/123456789/7415/mets.xml"/>
      </referenceSet>
    </reference>
  </referenceSet>
</div>
</body>
```

Full List of DRI Elements

Below is a full list of DRI Elements. For a detailed description of all the elements please refer to “DRI Schema Reference” [2].

BODY, cell, div, DOCUMENT, field, figure, head, help, hi, includeSet, instance, item, label, list, META, metadata, object, objectInclude, objectMeta, OPTIONS, p, pageMeta, params, repository, repositoryMeta, row, table, trail, userMeta, value, xref.

Localization and Internationalization

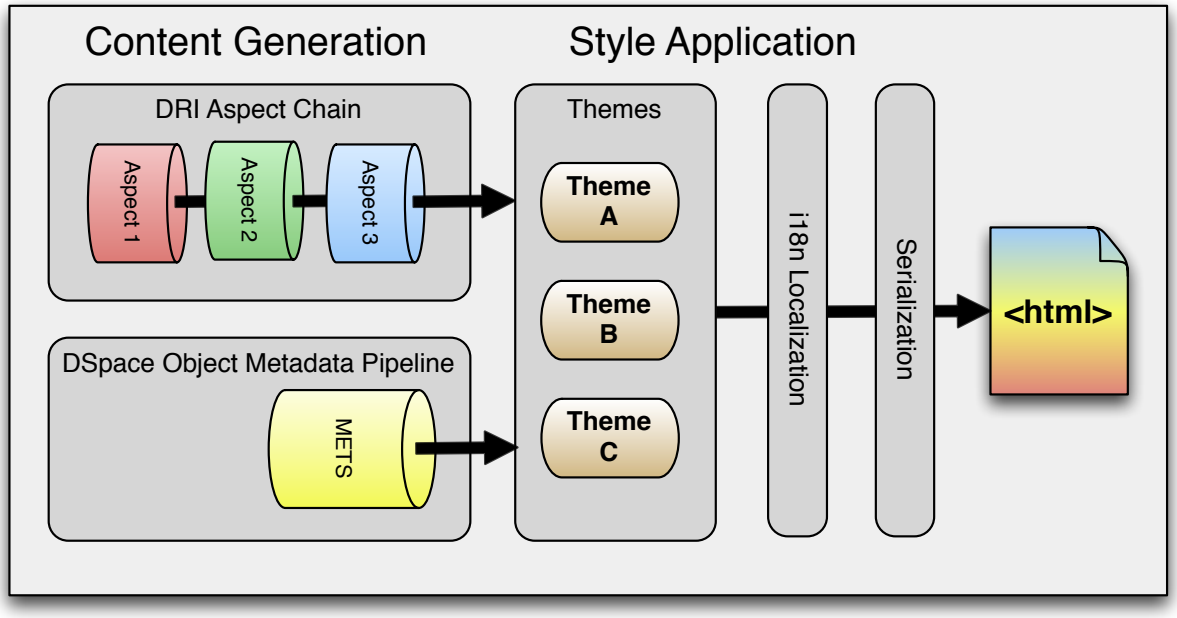
Internationalization is a very important component of the DRI system. It allows content to be offered in other languages based on user’s browser locale and conditioned upon availability of translations, as well as present dates and currency in a localized manner. There are two types of translated content: content stored and displayed by DSpace itself, and content introduced by the DRI styling process in the XSL transformations. Both types are handled by Cocoon’s i18n transformer without regard to their origin.

When the Content Generation process produces a DRI Document, some of the textual content may be marked up with i18n elements to signify that translations are available for that content. During the Style Application process, the Theme can also introduce new textual content, marking it up with i18n tags. As a result, after the Theme’s XSL templates are applied to the DRI Document, the final output consists of a DSpace page marked up in the chosen display format (like XHTML) with i18n elements from both DSpace and XSL content. This final document is sent through Cocoon’s i18n transformer that translates the marked up text.

[2] http://di.tamu.edu/projects/xmlui/schemaReference#Schema%20Overview

3. Architecture Overview

The following diagram provides the reader with an overview of the Manakin Architecture and its components.



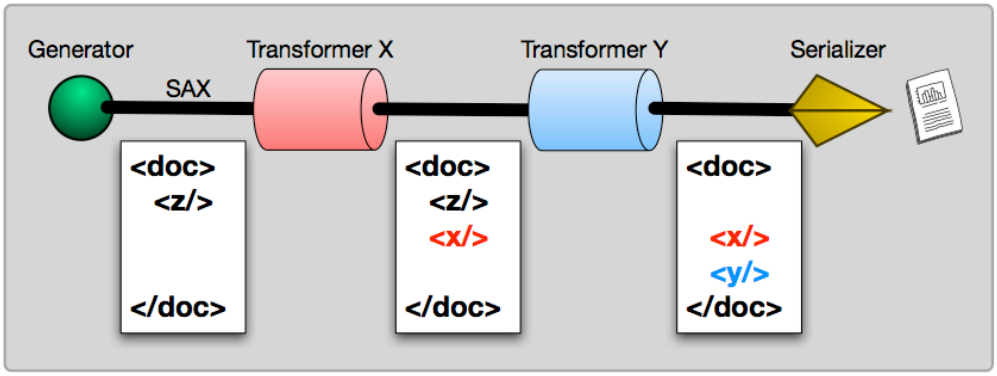
inspired by: "Manakin Developer's Guide" Scott Phillips et al.[3]

Aspects

In Aspect-Oriented Programming (AOP) programs are broken down into distinct parts that overlap as little as possible. These parts are called aspects, and woven together to form the program. 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. In addition to modifying the in-process document for display, Aspects execute the code that determines how the page is displayed.



[3] http://di.tamu.edu/projects/xmlui/resources/DevelopersGuide.pdf

Merging DRI Documents

Having described the structure of the DRI Document and having introduced aspects, it remains to explain how merging two DRI Documents into one occurs. In Manakin every Aspect is responsible for adding different functionality to a DSpace page. Since every instance of a page has to be a complete DRI Document, each Aspect is faced with the task of merging the Document it generated with the ones generated by previously executed Aspects. For this reason rules exist that describe which elements can be merged together and what happens to their data and child elements in the process.

When merging two DRI Documents, one is considered to be the main document, and the other a feeder document that is added in. The three top level containers (`meta`, `body` and `options`) of both documents are then individually analyzed and merged. In the case of the options and meta elements, the children tags are taken individually as well and treated differently from their siblings.

The `body` elements are the easiest to merge: their respective `div` children are preserved along with their ordering and are grouped together under one element. Thus, the new `body` tag will contain all the divs of the main document followed by all the divs of the feeder. However, if two divs have the same `n` and `rend` attributes, those divs will be merged into one. The resulting div will bear the `id`, `n`, and `rend` attributes of the main document's div and contain all the divs of the main document followed by all the divs of the feeder. This process continues recursively until all the divs have been merged.

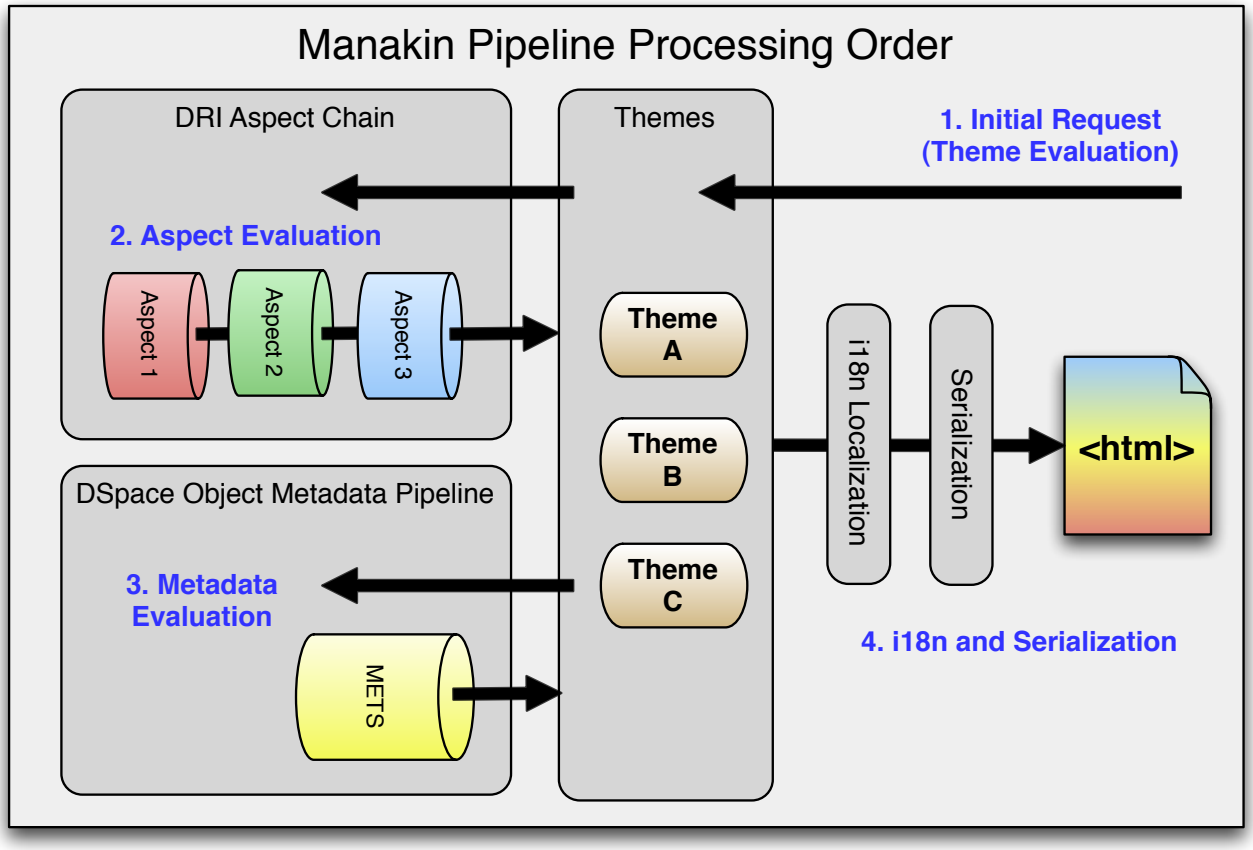
Merging the `options` elements is somewhat different. First, list elements under `options` of both documents are compared with each other. Those unique to either document are simply added under the new options element. In case of duplicates, that is list elements that belong to both documents and have the same `n` attribute, the two lists will be merged into one. The new `list` element will consist of the main document's `head` element, followed label-item pairs from the main document, and then finally the label-item pairs of the feeder, provided they are different from those of the main.

Finally, the meta elements are merged much like the elements under body. The three children of `meta`, `userMeta`, `pageMeta`, and `objectMeta` are individually merged, adding the contents of the feeder after the contents of the main.

Themes

Manakin Themes stylize the content generated by Manakin to a display format suitable for the user. The format typically is XHTML. Themes are implemented as XSL stylesheets applied in multiple stages to a pipeline's contents. The stylesheets, along with any static resources that may be required by the theme such as images, multimedia, CSS stylesheets, help documents, and translations, are all packaged together. Themes may be configured to apply to all pages in one or more community or collection. They can also be configured to apply to a singe arbitrary page.

Processing Order



In Manakin, the theme (and more specifically, its pipeline) acts as an initial request controller. The first pipeline activated when a request is processed, is the theme pipeline. Once activated, matchers present in the theme sitemaps and `xmlui.xconf` determine which theme will be applied. The applied theme then calls out to the DRI Aspect Chain.

Processing within the DRI Aspect chain again evaluates the request for specific matching criteria as each aspect is processed. Each aspect provides its own `sitemap.xmap` file with a pipeline with a specialized "AspectGenerator", cocoon matchers, and for each specific mapping, an implementation of an "AspectTransformer" responsible for properly processing DRI SAX events generated within the Aspect Pipeline.

Configuration

xmlui.xconf

<aspects>

This section configures the Aspect "chain". An Aspect provides a set of coupled features for the system. All Aspects are chained together such that together they form the complete DSpace website. This is where the chain is defined, the order in which each aspect is declared determines its order in the chain. Aspects at the top are invoked first.

The `<aspect>` element has two attributes, `name` & `path`. The name is used to identify the Aspect, while the path determines the directory. The path attribute should be listed exactly as it is found in the `dspace-xmlui-api/src/main/resources/aspects/` directory followed by a slash.

<themes>

This section configures which Theme should apply to a particular URL. Themes stylize an abstract DRI document (generated by the Aspect chain from above) and produce XHTML (or possibly another format) for display to the user. Each theme rule is processed in the order that it is listed below, the first rule that matches is the theme that is applied.

The `<theme>` element has several attributes including: `name`, `id`, `regex`, `handle`, and `path`. The `name` attribute is used to identify the theme, while the `path` determines the directory. The `path` attribute should be listed exactly as it is found in the `dspace-xmlui-webapp/src/main/webapp/themes/` directory. Both the `regex` and `handle` attributes determine if the theme rule matches the URL.

Keep in mind that the order of `<theme>` elements matters in the case of overlapping matching rules. For example, a theme rule with a very broad matching rule (like `regex=".*"`, which represents all pages) will override a more specific theme declaration (like `handle="1234/23"`, which might represents a DSpace item) if placed before it.

Finally, theme application also "cascades" down to pages derived from the one that the theme directly applies to. Thus, a theme applied to a specific community will also apply to that community's collections and their respective items.

Example

```
<xmlui>
    <aspects>
        <aspect name="Artifact Browser" path="resource://aspects/ArtifactBrowser/" />
        <aspect name="Administration" path="resource://aspects/Administrative/" />
        <aspect name="E-Person" path="resource://aspects/EPerson/" />
        <aspect name="Submission and Workflow" path="resource://aspects/Submission/" />
    </aspects>
    <themes>
        <theme name="Test Theme 1" handle="123456789/1" path="theme1/"/>
        <theme name="Test Theme 2" regex="community-list" path="theme2/"/>
        <theme name="Default Reference Theme" regex=".*" path="Reference/" />
    </themes>
</xmlui>
```

Configuring Aspects

Sitemaps

An example sitemap can be found at

`dspace-xmlui/dspace-xmlui-api/src/main/resources/aspects/XMLTest/sitemap.xmap`

Part of this sitemap is displayed here:


```
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
   <map:components>
      <map:transformers>
         <map:transformer name="Navigation" src="org.dspace.app.xmlui.aspect.xmltest.Navigation" />
         <map:transformer name="HTMLTest" src="org.dspace.app.xmlui.aspect.xmltest.HTMLTest" />
      </map:transformers>
   </map:components>
   <map:pipelines>
      <map:pipeline>
         <map:generate />
         <map:transform type="Navigation" />
         <map:match pattern="xmltest/HTML">
            <map:transform type="HTMLTest" />
         </map:match>
         <map:serialize type="xml" />
      </map:pipeline>
   </map:pipelines>
</map:sitemap>
```

The sitemap contains a definition of the transformer Components:
- `org.dspace.app.xmlui.aspect.xmltest.Navigation`
- `org.dspace.app.xmlui.aspect.xmltest.HTMLTest`

These are java classes generating DRI.

The Pipeline contains the following sections:
- Generate: Default Generator Applied to capture SAX events from a previous Aspect
- Navigation Transformer: A transformer applied to all requests, which adds the links in the sidebar.
- HTMLTest Transformer: A transformer applied only to the request URI: http://localhost:8080/xmlui/xmltest/HTML which will create the DRI required to render the contents of the page.
- Serialization: allows the processing to be handed to the next Aspect in the Pipeline.

Matchers within the aspect sitemaps use REGEX patterns to determine whether the transformer should be applied. These patterns are compared to the part of URL after the contextpath. In the example here, http://localhost:8080/xmlui/xmltest/HTML, the contextpath is /xmlui and “xmltest/HTML” is the part used in the matcher. More complicated matchers are based on java code. An example is

```
<map:match type="HandleTypeMatcher" pattern="collection">
```

which will check whether the currently displayed handle is a collection.

Aspect Java Classes

Example classes from the above XMLTest aspect. These show the basic wiring of a sitemap to java classes.

Navigation

The `org.dspace.app.xmlui.aspect.xmltest.Navigation` class provides an example of utilizing the Wing framework to add Navigation Options to the Options section of the DRI document. An extract of this class is displayed here:

```
public class Navigation extends AbstractDSpaceTransformer
{
    public void addOptions(Options options) throws SAXException, WingException,
            UIException, SQLException, IOException, AuthorizeException
    {
        List test = options.addList("XMLTest");
        test.setHead("XML Test");
        test.addItemXref(contextPath + "/xmltest/HTML","HTML");
    }
}
```

This results in the DRI:


```
<list id="aspect.xmltest.Navigation.list.XMLTest" n="XMLTest">
<head>XML Test</head>
<item>
<xref target="/ntu/xmltest/HTML">HTML</xref>
</item>
</list>
```

HTMLTest

The `org.dspace.app.xmlui.aspect.xmltest.HTMLTest` class adds content to the body of a DRI document during aspect processing. This example also adds `pageMetadata` such as the Page Title and Breadcrumb Trail features.

```
public class HTMLTest extends AbstractDSpaceTransformer
{
        public void addPageMeta(PageMeta pageMeta) throws SAXException, WingException, UIException,
                        SQLException, IOException, AuthorizeException {
                pageMeta.addMetadata("title").addContent("HTML Test");
                pageMeta.addTrailLink(contextPath + "/", "DSpace Home");
                pageMeta.addTrail().addContent("HTML Test");
        }

        public void addBody(Body body) throws SAXException, WingException,
                        UIException, SQLException, IOException, AuthorizeException {
                Request request = ObjectModelHelper.getRequest(objectModel);

                String fragment =
                "<p>This is a test of manakin's ability to render HTML fragments.</p>";

                boolean blankLines = false;

                Division test = div.addDivision("html-test-sample");
                test.setHead("Rendered Sample");

                test.addSimpleHTMLFragment(blankLines, fragment);
        }
}
```

This results for the pagemeta in the DRI:

```
<metadata element="title">HTML Test</metadata>
<trail target="/ntu/">DSpace Home</trail>
<trail>HTML Test</trail>
```

and for the body:

```
<div id="aspect.xmltest.HTMLTest.div.html-test-sample" n="html-test-sample">
<head>Rendered Sample</head>
<p>
This is a test of manakin's ability to render HTML fragments.
</p>
</div>
```

Theme Transformer and Sitemap

- Selects the appropriate Theme to be applied based on configuration in xmlui.xconf
- Applies the i18n Transform to render localized text strings into a designated language.
- Serializes DRI Stream to XHTML

Basic Theme structure (example “template” theme)

Directory Resources:

A typical layout of a theme consists of:

- a directory containing all images related to the theme
- the css stylesheets required to design the pages
- an XSLT stylesheet transformer (e.g. `template.xsl`) containing all changes to the existing templates such as the header and footer
- and a sitemap.xmap file to match the theme directory to the css files and the xsl file

```
template/
   |-- images
   |   `-- logo.gif
   |-- lib
   |   |-- style-ie.css
   |   `-- style.css
   |-- sitemap.xmap
   `-- template.xsl
```

template.xsl

The XSLT stylesheet transformer will convert the DRI to XHTML. All themes import dri2xhtml.xsl as a start transformation to XHTML. Parts of dri2xhtml.xsl (or files it refers to such as structural.xsl) can be overridden in the theme’s xsl. An example of a customized footer is given here:

```
<xsl:stylesheet>

   <xsl:import href="../dri2xhtml.xsl" />
   <xsl:output indent="yes" />

   <!-- An example of an existing template copied from structural.xsl and overridden -->
   <xsl:template name="buildFooter">
      <div id="ds-footer">
         <div id="ds-footer-links">
            <a>
               <xsl:attribute name="href">
                        <xsl:value-of
                  select="/dri:document/dri:meta/dri:pageMeta/dri:metadata[@element='contextPath']
                              [not(@qualifier)]" />
                        <xsl:text>/contact</xsl:text>
                     </xsl:attribute>
               <i18n:text>xmlui.dri2xhtml.structural.contact-link
               </i18n:text>
            </a>
            <xsl:text> | </xsl:text>
            <a>
               <xsl:attribute name="href">
                        <xsl:value-of
                  select="/dri:document/dri:meta/dri:pageMeta/dri:metadata[@element='contextPath']
                              [not(@qualifier)]" />
                        <xsl:text>/feedback</xsl:text>
                     </xsl:attribute>
               <i18n:text>xmlui.dri2xhtml.structural.feedback-link
               </i18n:text>
            </a>
         </div>
      </div>
   </xsl:template>
</xsl:stylesheet>
```


Common templates for the four metadata display “modes” [4]

- `SummaryList` = Summarized list of objects (Community/Collection listing, item browse/search)
- `SummaryView`= Summarized view of single object (Item page summary metadata)
- `DetailList`= Detailed list of objects Rarely used: (“Item appears in following collections”)
- `DetailView`= Detailed view of single object (Item’s full record page, collection/community pages)

DSpace METS Metadata Representation

Default Sitemap Configuration

METS XML representations are generated by one Cocoon Generator, “DSpaceMETSGenerator”

```
<map:generator name="DSpaceMETSGenerator"
src="org.dspace.app.xmlui.cocoon.DSpaceMETSGenerator"/>
```

The METS metadata pipeline is used for generating the metadata describing a DSpaceObject (e.g. an item). It is configured to match any patch prefixed with “metadata/handle/*/*/**” both internally and externally. METS representations are based on the DSpace METS SIP Profile. METS Content is Generated for any view of the following DSpaceObjects: Site, Community, Collection, Item. Views of these Items can be specific to that object: “Item Detail View”, “Collection Detail View”, or they can be aggregations of such objects, Search Result List, Browse Result List or Recently Submitted List.

Example METS serialization:

```
<mets:METS>
   <mets:dmdSec GROUPID="group_dmd_0" ID="dmd_1">
      <mets:mdWrap OTHERMDTYPE="DIM" MDTYPE="OTHER">
         <mets:xmlData>
            <dim:dim dspaceType="ITEM">
               <dim:field element="title" mdschema="dc" language="en">Earth map</dim:field>
               ...
            </dim:dim>
         </mets:xmlData>
      </mets:mdWrap>
   </mets:dmdSec>
   <mets:fileSec>
      <mets:fileGrp USE="CONTENT">
         <mets:file SIZE="7190373" GROUP_ID="group_file_8898"
            CHECKSUM="0fdb412dae2606682696a68f131d10ab" KIND="JPEG Image"
            MIMETYPE="image/jpeg" CHECKSUMTYPE="MD5" ID="file_8898">
            <mets:FLocat LOCTYPE="URL"
               xlink:href="/labs/bitstream/handle/123456789/7618/earth-map-huge.jpg?sequence=1"
               xlink:title="earth-map-huge.jpg" xlink:type="locator"
               xlink:label="Huge Map" />
         </mets:file>
      </mets:fileGrp>
      ...
   </mets:fileSec>
   <mets:structMap TYPE="LOGICAL" LABEL="DSpace">
      <mets:div TYPE="DSpace Item" DMDID="dmd_1">
         <mets:div TYPE="DSpace Content Bitstream" ID="div_2">
            <mets:fptr FILEID="file_8898" />
         </mets:div>
      </mets:div>
   </mets:structMap>
</mets:METS>
```

The following METS Example is for a DSpace Item and is comprised of three important sections:

[4] http://www.slideshare.net/tdonohue/making-dspace-15-your-own-customizations-via-overlays

- Descriptive Metadata (dmdSec): All Metadata describing the Item. In the theming layer, this view is utilized whenever a Summary View provides a Title, Description, Date, Author or Link in the rendering of the DSpaceObject in a List, or whenever a Detailed View is rendered.

```
<mets:dmdSec GROUPID="group_dmd_0" ID="dmd_1">
   <mets:mdWrap OTHERMDTYPE="DIM" MDTYPE="OTHER">
      <mets:xmlData>
         <dim:dim dspaceType="ITEM">
            <dim:field element="contributor" mdschema="dc" qualifier="author">
                   Smith, Donald Jr
            </dim:field>
            <dim:field element="date" mdschema="dc" qualifier="issued">
                   2009-02-20T15:32:52Z
            </dim:field>
            <dim:field element="identifier" mdschema="dc" qualifier="uri">
                   https://atmire.com/labs/handle/123456789/7618
            </dim:field>
            <dim:field element="subject" mdschema="dc">Earth map</dim:field>
            <dim:field element="title" mdschema="dc" language="en">Earth map</dim:field>
```

- File Section (fileSec): A description of all the provided file “Bitstreams” that are part of the Item. Each File Description provides a detailed manifest of attributes concerning the Bitstream file which are available to the theming layer for presentation. The “File Section” of a DSpace Item is divided into several File Groups that correspond to DSpace storage Bundles in DSpace. The original content is found within a “CONTENT” file group, license details under, a “LICENSE” file group, extracted text under a “TEXT” filegroup, thumbnails under “THUMBNAIL” and so on.

```
<mets:fileSec>
   <mets:fileGrp USE="CONTENT">
      <mets:file SIZE="7190373" GROUP_ID="group_file_8898"
         CHECKSUM="0fdb412dae2606682696a68f131d10ab" KIND="JPEG Image"
         MIMETYPE="image/jpeg" CHECKSUMTYPE="MD5" ID="file_8898">
         <mets:FLocat LOCTYPE="URL"
            xlink:href="/labs/bitstream/handle/123456789/7618/earth-map-huge.jpg?sequence=1"
            xlink:title="earth-map-huge.jpg" xlink:type="locator"
            xlink:label="Huge Map" />
      </mets:file>
   </mets:fileGrp>
```

DSpace Object METS representations are accessed via XSLT “document” calls during the theming transformation phase. Resulting in additional XML trees representing Metadata content required for generating presentation (found in themes/dri2xhtml/structural.xsl)

```
<xsl:template match="dri:reference" mode="summaryView">
    <xsl:variable name="externalMetadataURL">
        <xsl:text>cocoon:/</xsl:text>
        <xsl:value-of select="@url"/>
        <!-- No options selected, render the full METS document -->
    </xsl:variable>
    <xsl:apply-templates
           select="document($externalMetadataURL)"
           mode="summaryView"/>
    <xsl:apply-templates />
</xsl:template>
```

DSpace sitemap

Main sitemap

The main sitemap (`dspace-xmlui-webapp/src/main/webapp/sitemap.xmap`) provides resolution of the following resources:

- The primary DRI Aspect Pipeline utilized to merge default and custom Aspects:
  - DRI AspectTransform responsible for merging DRI Aspects

```
<map:match pattern="DRI/**">
        <map:mount check-reload="no" src="aspects/aspects.xmap" uri-prefix="DRI/"/>
</map:match>
```

  - Theme Transformer responsible for converting from DRI to HTML serialization

```
<!-- handle common theme resources, such as dri2xhtml -->
<map:match pattern="themes/*">
        <map:read src="themes/{1}"/>
</map:match>

<!-- handle theme specific resources static or dynamic -->
<map:match pattern="themes/*/**">
        <map:mount check-reload="no" src="themes/{1}/sitemap.xmap" uri-prefix=""/>
</map:match>
```

- The BitstreamReader and its Pipeline responsible for providing download of bitstreams.

```
<map:pipelines>
        <!-- Bitstream pipeline -->
        <map:pipeline type="noncaching">
                <map:parameter name="outputBufferSize" value="8192"/>
                <map:parameter name="expires" value="now"/>
                <map:match pattern="bitstream/handle/*/*/**">
                        <map:read type="BitstreamReader">
                                <map:parameter name="handle" value="{1}/{2}"/>
                                <map:parameter name="name" value="{3}"/>
                        </map:read>
                </map:match>
```

- OpenURL Resolution
- RSS/Atom Syndication Feed Pipeline responsible for rendering

```
<map:pipelines>
        <!-- Bitstream pipeline -->
        <map:pipeline type="noncaching">
                <map:parameter name="outputBufferSize" value="8192"/>
                <map:parameter name="expires" value="now"/>
                <map:match pattern="bitstream/handle/*/*/**">
                        <map:read type="BitstreamReader">
                                <map:parameter name="handle" value="{1}/{2}"/>
                                <map:parameter name="name" value="{3}"/>
                        </map:read>
                </map:match>
```

- HTML Sitemaps and Sitemaps.org Sitemaps for whole repository

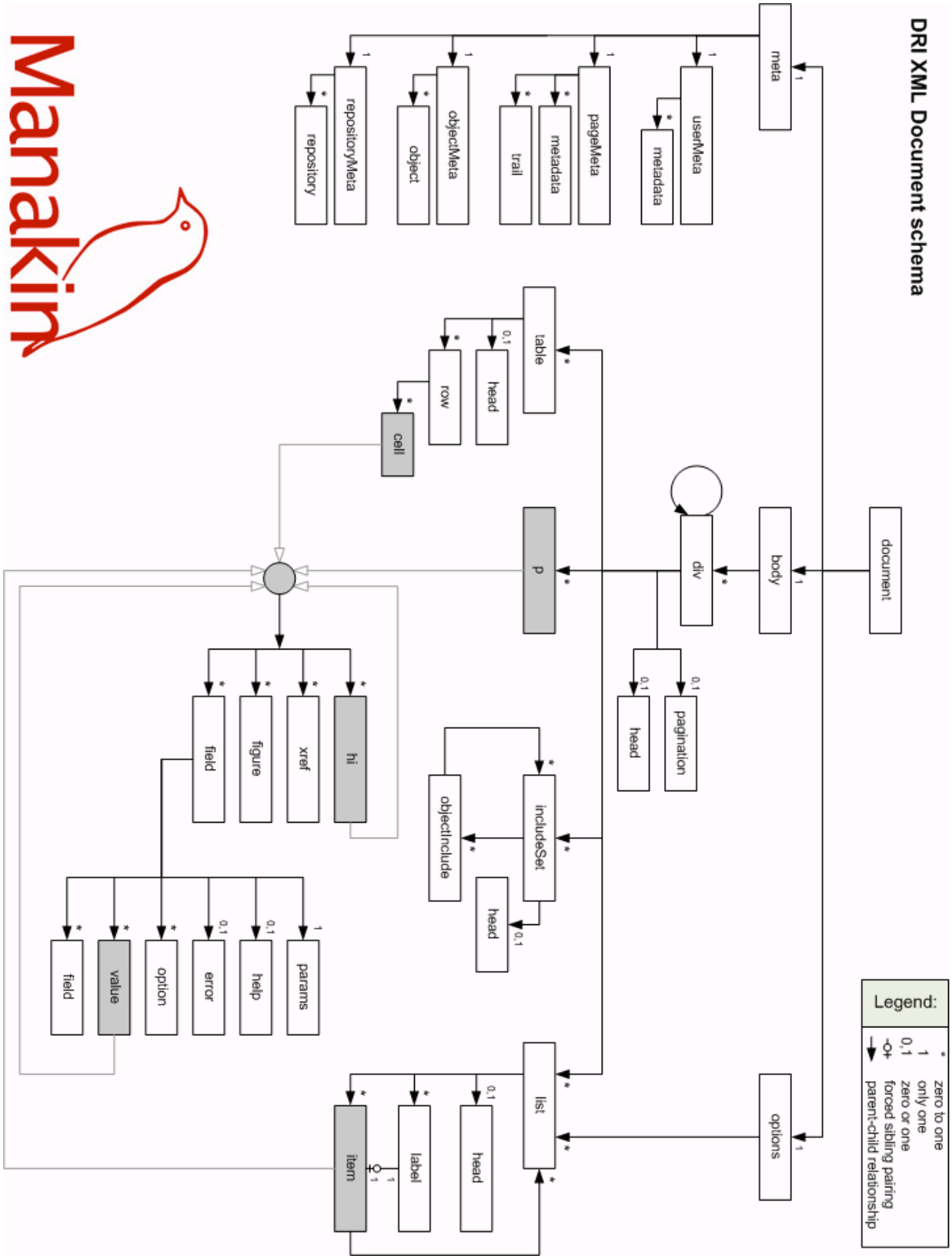
```
<map:match pattern="htmlmap">
        <map:read type="SitemapReader">
                <map:parameter name="type" value="html"/>
        </map:read>
</map:match>
<map:match pattern="sitemap">
        <map:read type="SitemapReader">
                <map:parameter name="type" value="sitemaps.org"/>
        </map:read>
</map:match>
```


- DSpaceObject METS Metadata Representation pipeline

```
        <map:match pattern="metadata/**">
                <map:match pattern="metadata/handle/*/*/**">
                        <map:generate type="DSpaceMETSGenerator">
                                <map:parameter name="handle" value="{1}/{2}"/>
                                <map:parameter name="extra" value="{3}"/>
                        </map:generate>
                        <map:serialize type="xml"/>
                </map:match>
                ...
```


4. Reference

DRI Reference Diagram [5] :



[5] http://di.tamu.edu/projects/xmlui/images/Drawing6c.png

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.

