Developing with Apache Lenya 1.4

Andreas Hartmann

ApacheCon Europe 06

Questions To Be Answered

- Is Lenya the right product for me?
- Should I update from 1.2 to 1.4?
- What does the 1.4 architecture look like?
- How do I get started with my app?
- How do I add custom functionality?
- How do I support custom types of documents?

Agenda

- Overview
- New features in Lenya 1.4
- Architecture
- Core API
- Modules
- Publication Templating
- Usecase framework
- Resource types

Apache Lenya

- Content Management Framework
- based on Apache Cocoon (XML-based web publishing + application framework)
- Incubator 2003, TLP 2004
- Focus:
  - Cross-media publishing
  - WYSIWYG editor integration
  - Basis for implementing CMS functionality and integrating external apps

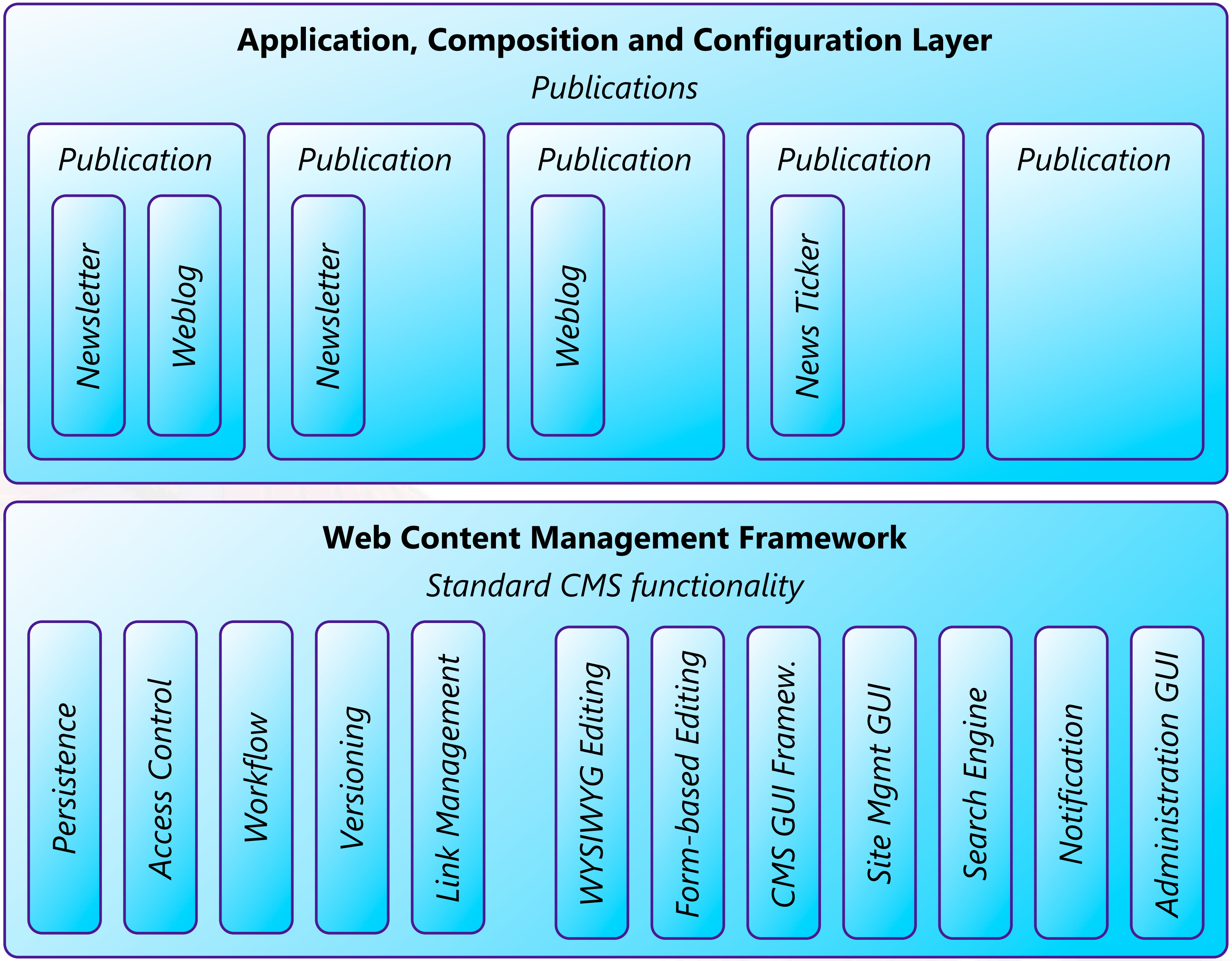
Some New Features

- Site overview
- SVG module
- WebDAV support
- Neutron / Phoenix
- Open Document resource type
- JCR integration

Architecture

- Lenya 1.2 Architecture
- Layering
- Modularisation
- Dependency management

Lenya 1.2



Application, Composition and Configuration Layer

Publications

Publication
- *Newsletter*
- *Weblog*

Publication
- *Newsletter*

Publication
- *Weblog*

Publication
- *News Ticker*

Publication

Web Content Management Framework

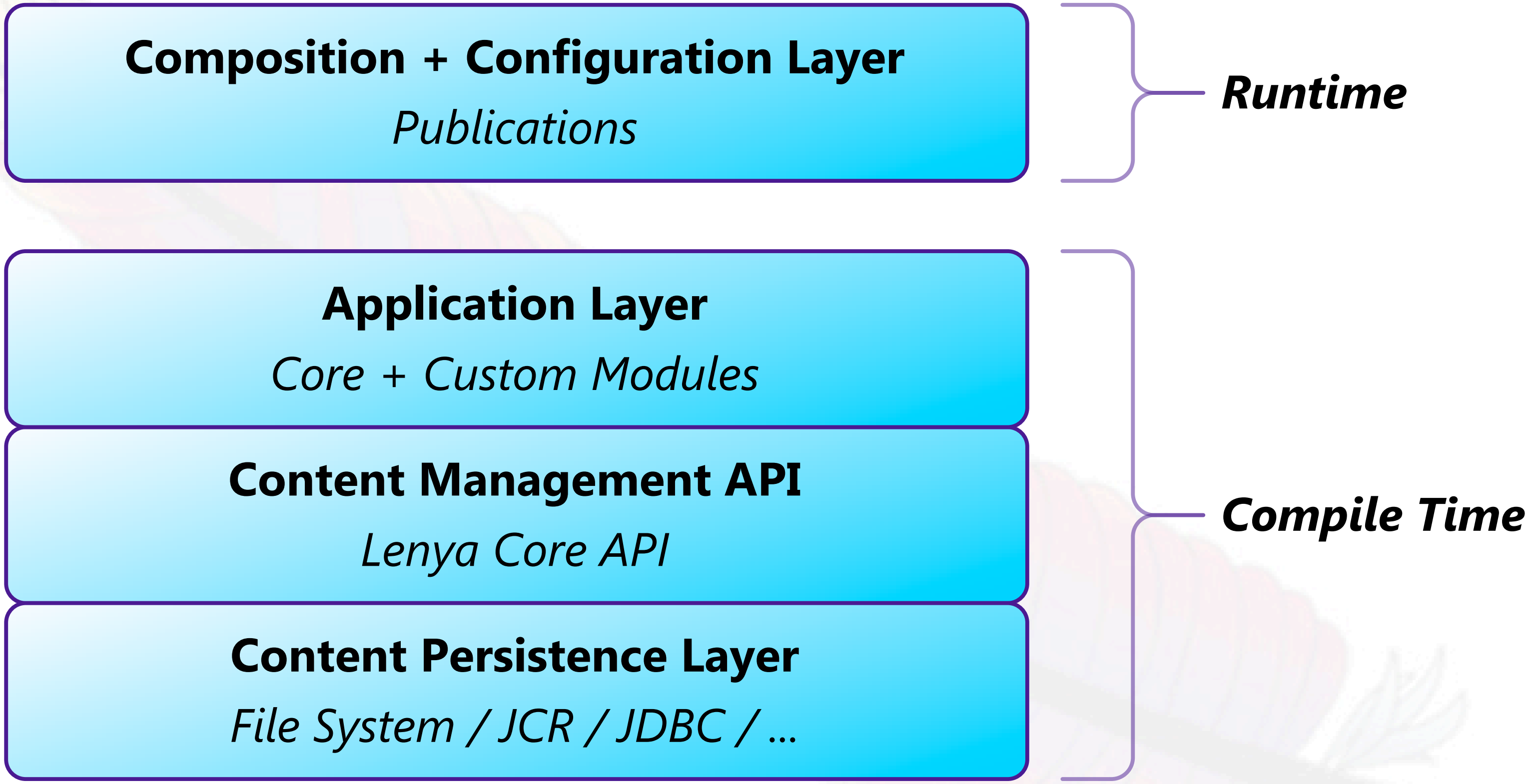
Standard CMS functionality

- *Persistence*
- *Access Control*
- *Workflow*
- *Versioning*
- *Link Management*
- *WYSIWYG Editing*
- *Form-based Editing*
- *CMS GUI Framew.*
- *Site Mgmt GUI*
- *Search Engine*
- *Notification*
- *Administration GUI*

Lenya 1.2 Architecture

- Core
  - Monolithic
  - Structured by file types, not by responsibility
  - Various CMS functionality included
  - No separation of “core” and “additional” functionality
- Publications
  - self-contained
  - sharing code required custom mechanisms

Layers in Lenya 1.4



Composition + Configuration Layer

Publications

Application Layer

Core + Custom Modules

Content Management API

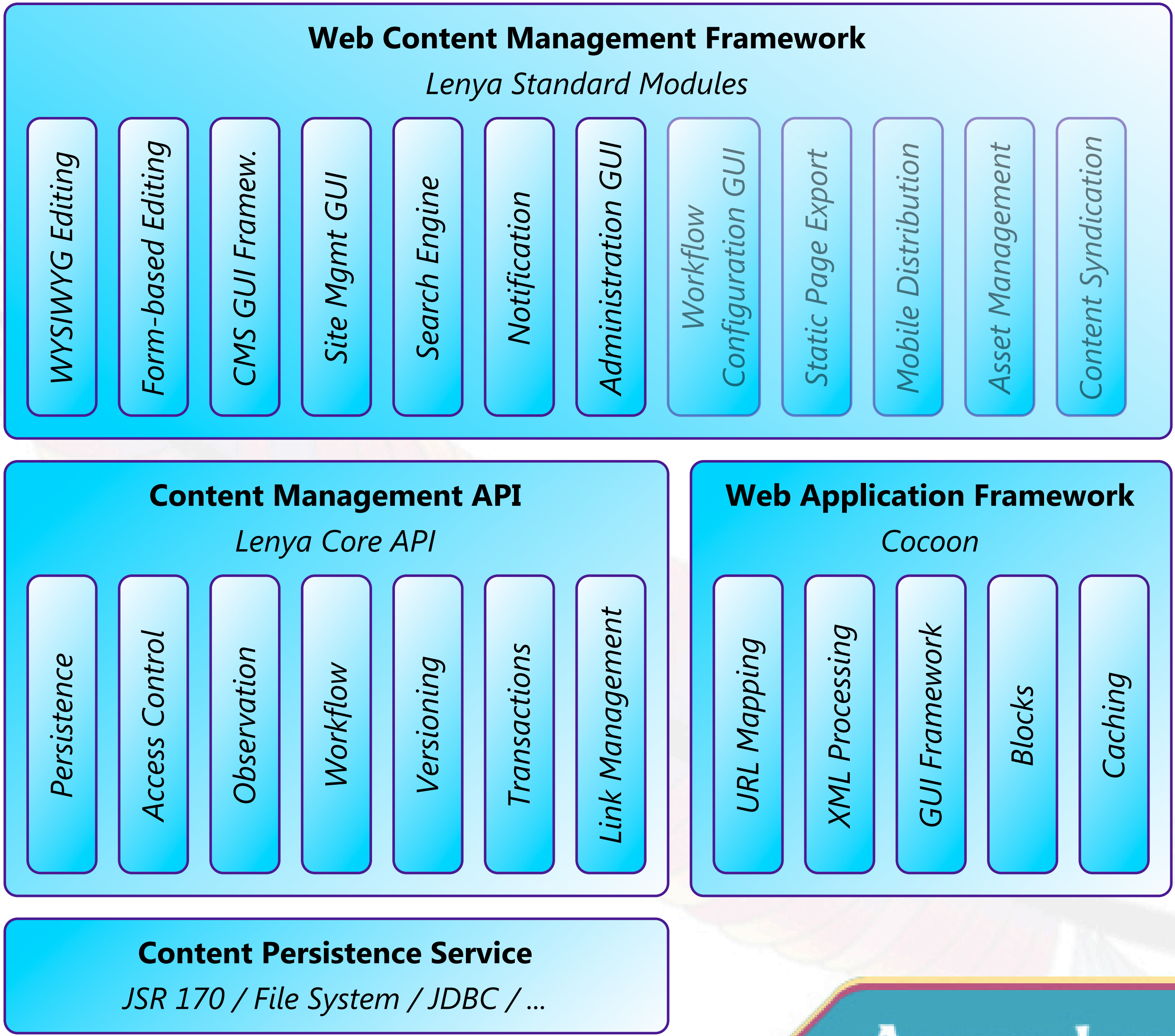
Lenya Core API

Content Persistence Layer

File System / JCR / JDBC / …

Runtime

Compile Time



Web Content Management Framework

Lenya Standard Modules

- *WYSIWYG Editing*
- *Form-based Editing*
- *CMS GUI Framew.*
- *Site Mgmt GUI*
- *Search Engine*
- *Notification*
- *Administration GUI*
- *Workflow Configuration GUI*
- *Static Page Export*
- *Mobile Distribution*
- *Asset Management*
- *Content Syndication*

Content Management API

Lenya Core API

- *Persistence*
- *Access Control*
- *Observation*
- *Workflow*
- *Versioning*
- *Transactions*
- *Link Management*

Web Application Framework

Cocoon

- *URL Mapping*
- *XML Processing*
- *GUI Framework*
- *Blocks*
- *Caching*

Content Persistence Service

JSR 170 / File System / JDBC / ...

Composition and Configuration Layer

Publications

- *Publication*
- *Publication*
- *Publication*
- *Publication*
- *Publication*

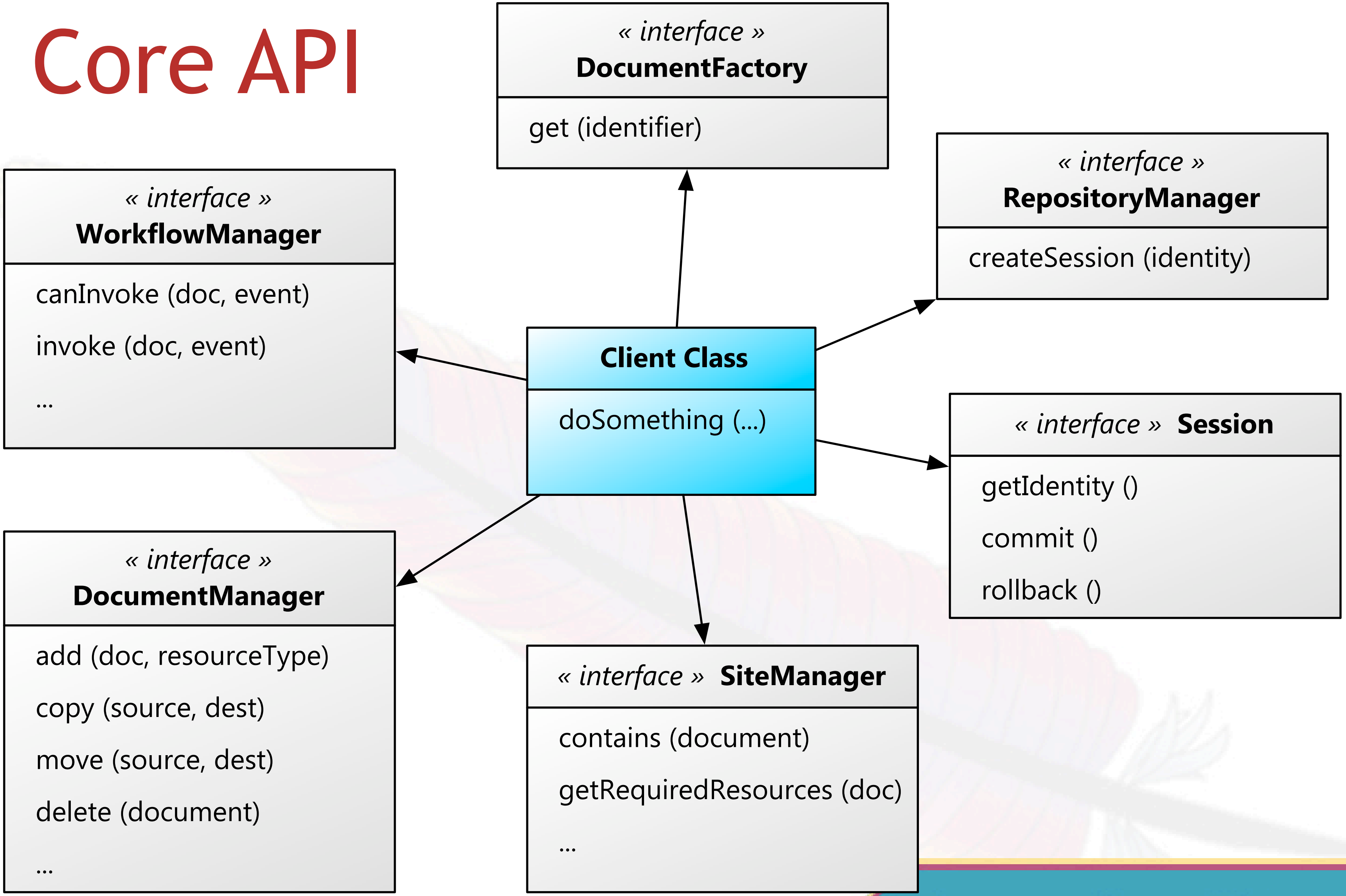
Application Layer

Optional and Custom Modules

- *Newsletter*
- *Forum*
- *Weblog*
- *Advanced Search*
- *Online Employment Application*
- *Contact Form*
- *News Ticker*
- *Image Gallery*
- *Online Shop*
- *Calendar*
- *RSS Feeds*
- *EAI Services*

Web Content Management Framework

Core API



« interface »
DocumentFactory
get (identifier)

« interface »
WorkflowManager
canInvoke (doc, event)
invoke (doc, event)
…

« interface »
RepositoryManager
createSession (identity)

Client Class
doSomething (...)

« interface » **Session**
getIdentity ()
commit ()
rollback ()

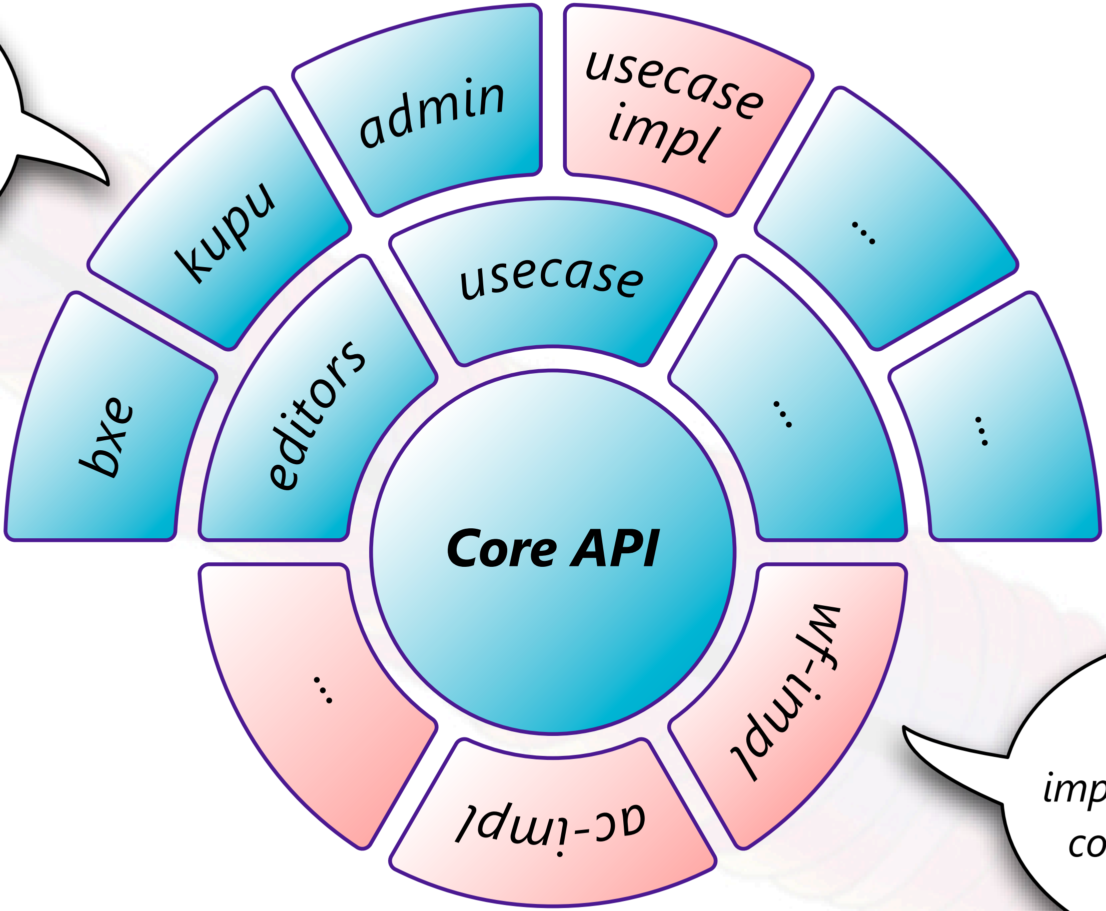
« interface »
DocumentManager
add (doc, resourceType)
copy (source, dest)
move (source, dest)
delete (document)
…

« interface » **SiteManager**
contains (document)
getRequiredResources (doc)
…

Modules: Purpose

- modularization of Lenya core
  - improve *Separation of Concerns*:
    isolate a specific aspect or functionality,
    encourage delegation
  - improve maintainability
  - improve long-term stability
  - reduce learning curve
- extend with 3rd party functionality
- reuse functionality

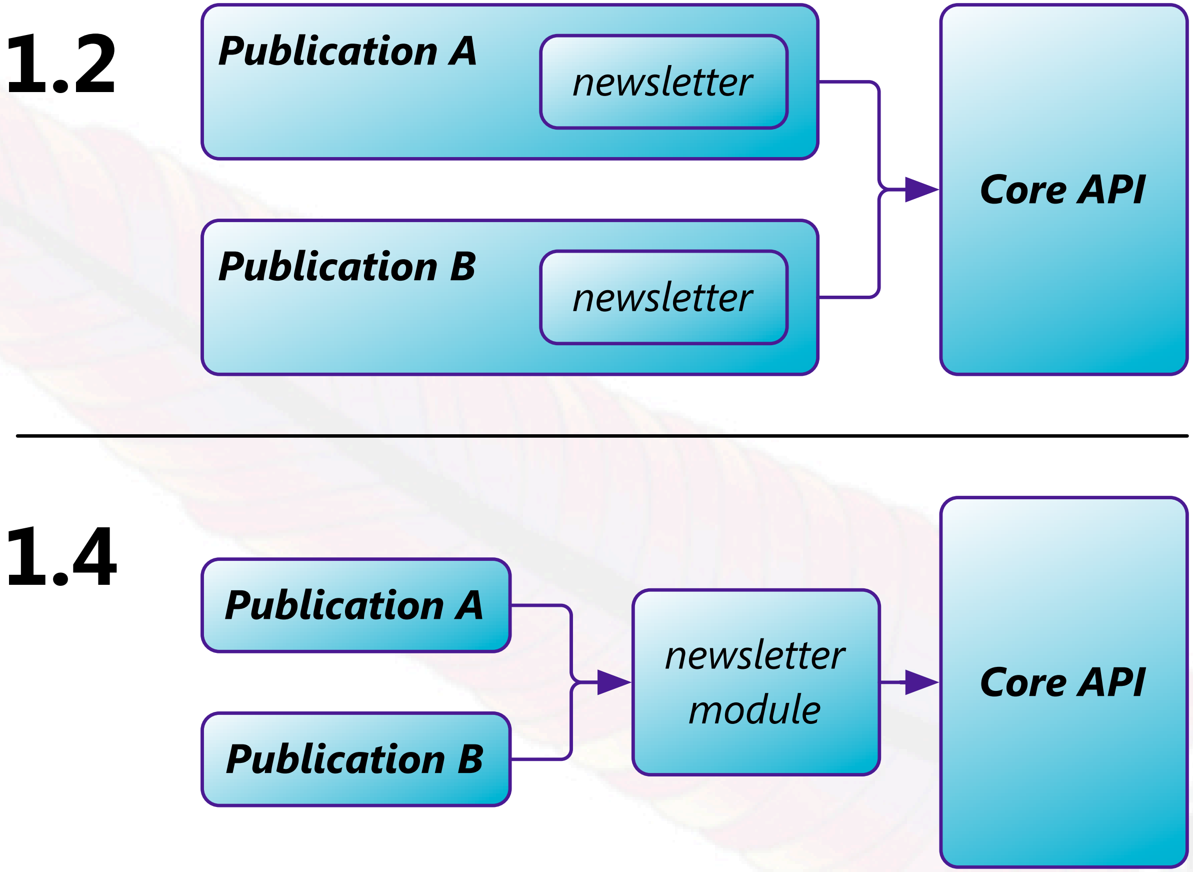
Modules - Core Modularisation



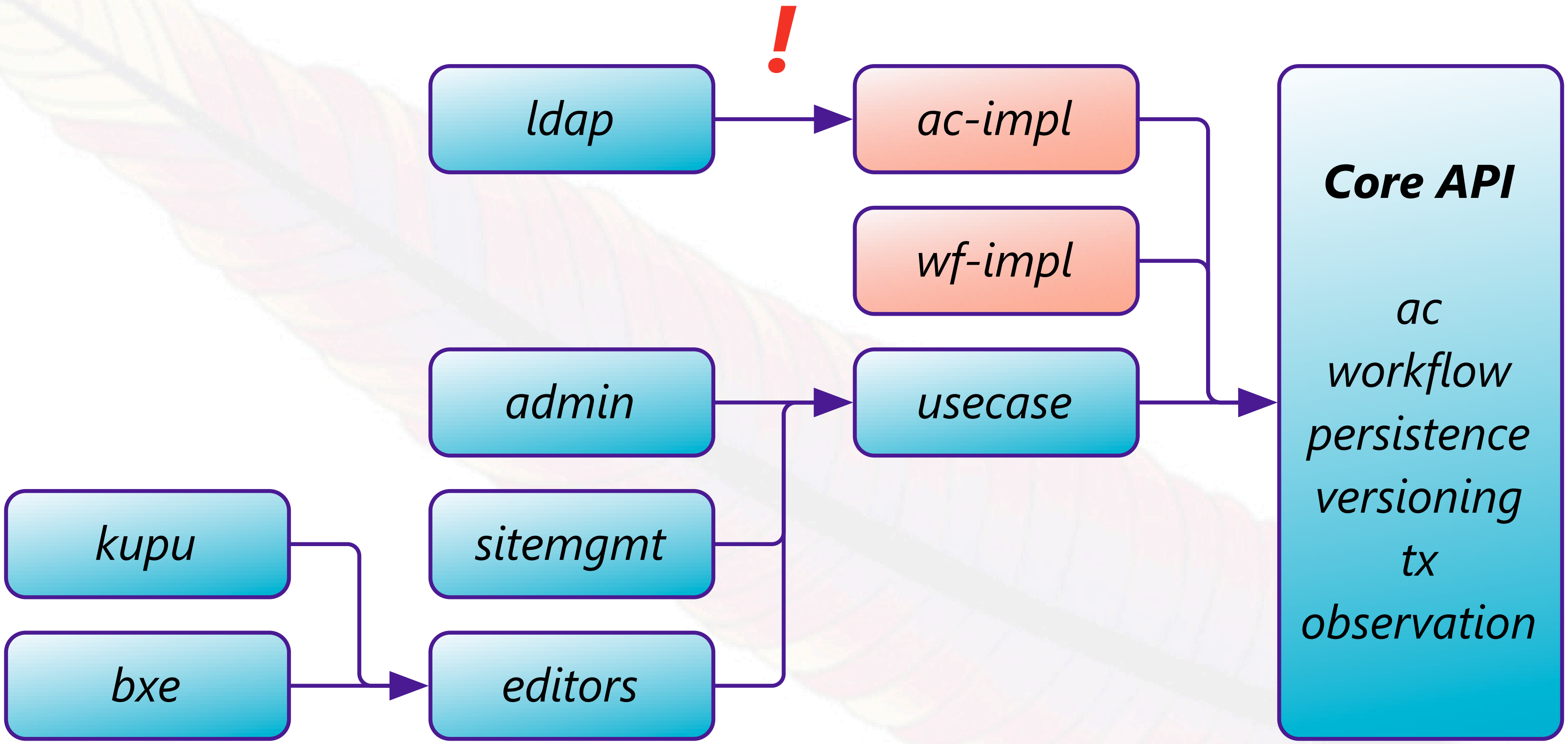
functionality beyond the Core API

default implementations of core API services

Modules - Reusing Functionality



Module Dependencies



!

ldap

ac-impl

wf-impl

admin

usecase

kupu

sitemgmt

bxe

editors

Core API

ac
workflow
persistence
versioning
tx
observation

Module Descriptor

```
<module xmlns="http://apache.org/lenya/module/1.0">
  <id>org.apache.lenya.modules.bxe</id>
  <published>false</published>
  <depends module="org.apache.lenya.modules.usecase"/>
  <package>org.apache.lenya.modules</package>
  <version>0.1-dev</version>
  <name>BXE Editor</name>
  <lenya-version>@lenya.version@</lenya-version>
  <description>
    Integration of the Bitflux WYSIWYG Editor.
    For more information, visit http://bxe.oscom.org
  </description>
  <installation>...</installation>
</module>
```

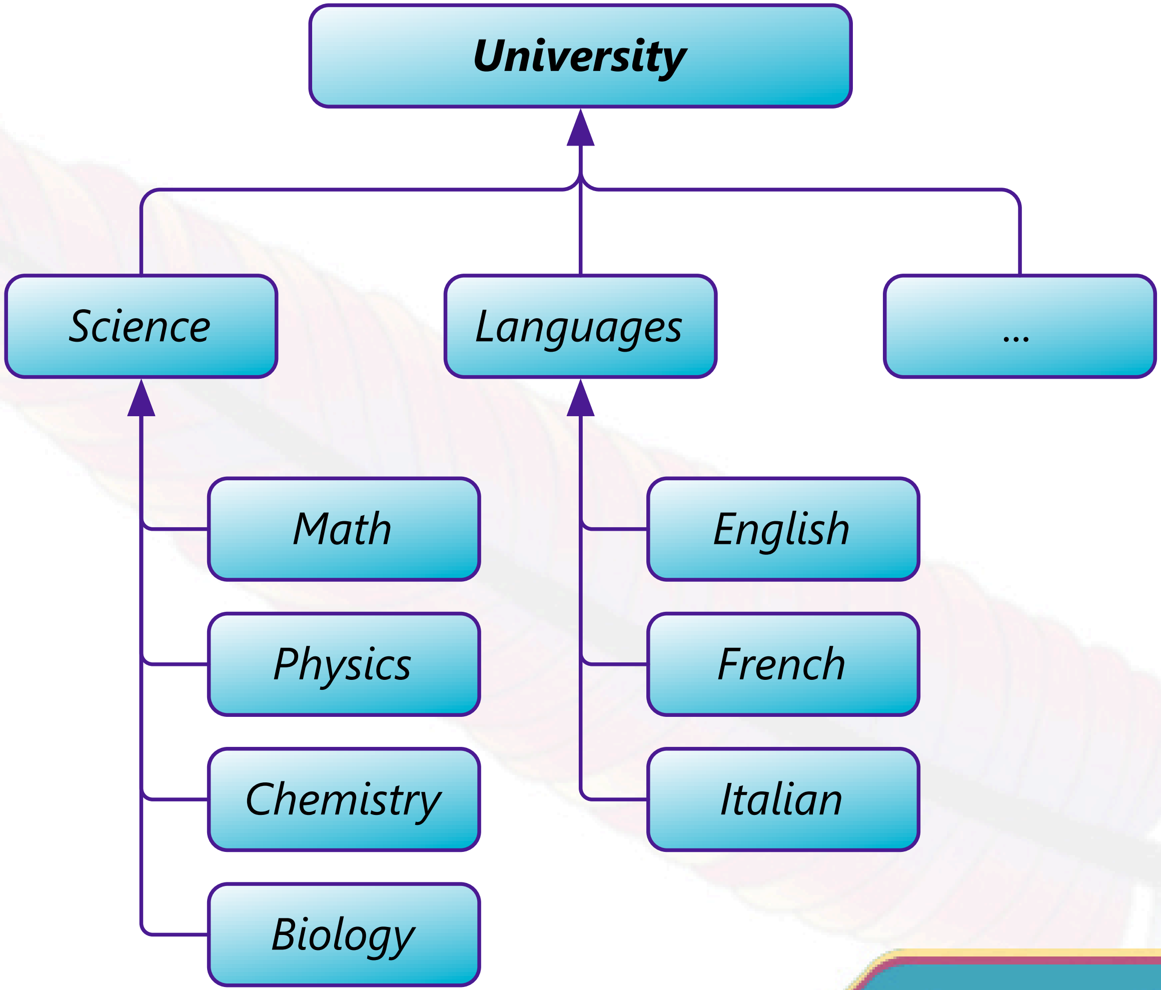
Module Layout

- yourmodule
  - config/
  - java/
    - src/
    - test/
  - usecases/
  - xslt/
  - sitemap.xmap

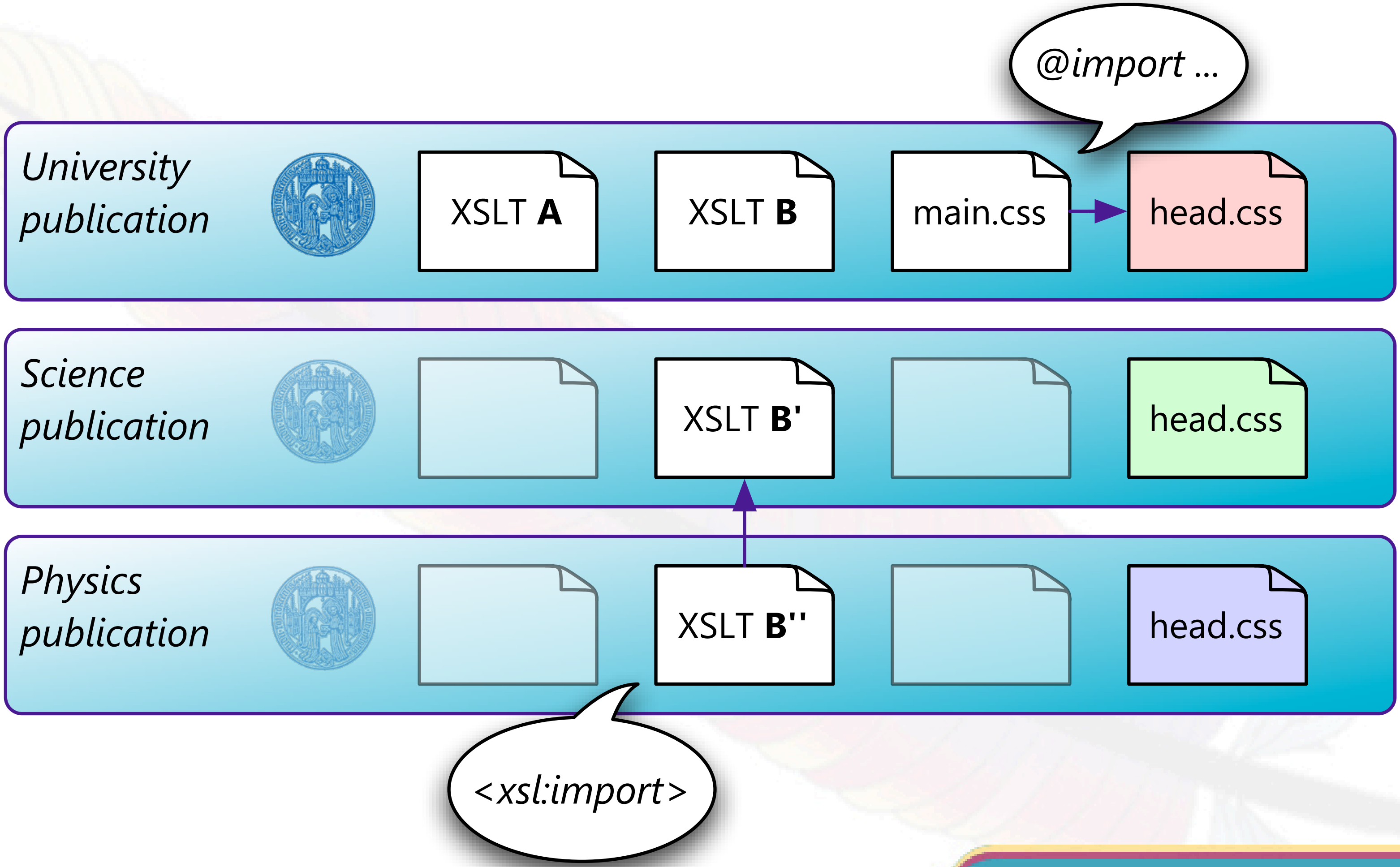
Modules: Summary

- Provide reusable functionality
  - Resource types
  - GUI functionality (usecases)
  - Integration code for external apps
  - Reusable pipelines
  - Static resources (images, ...)
- Declare service interfaces
- Provide implementations for interfaces

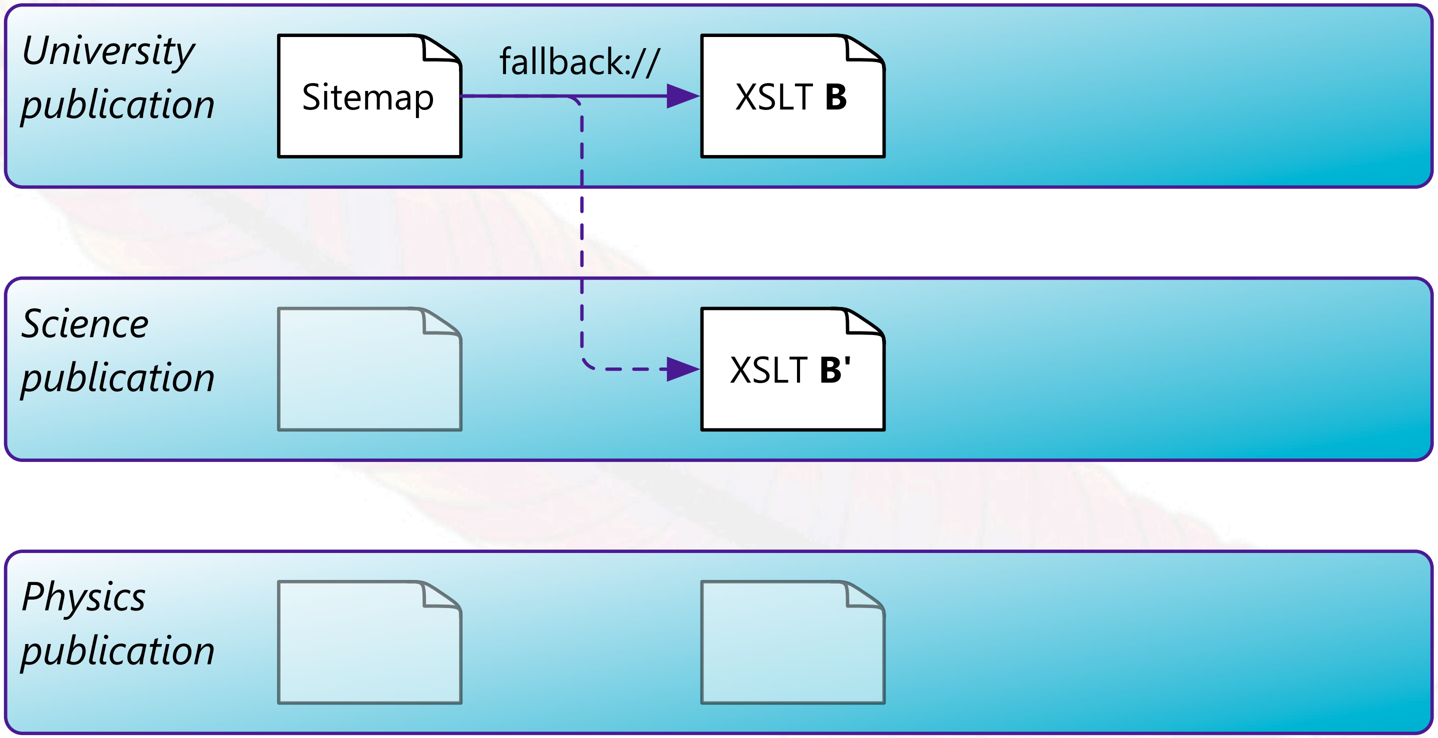
Publication Templating



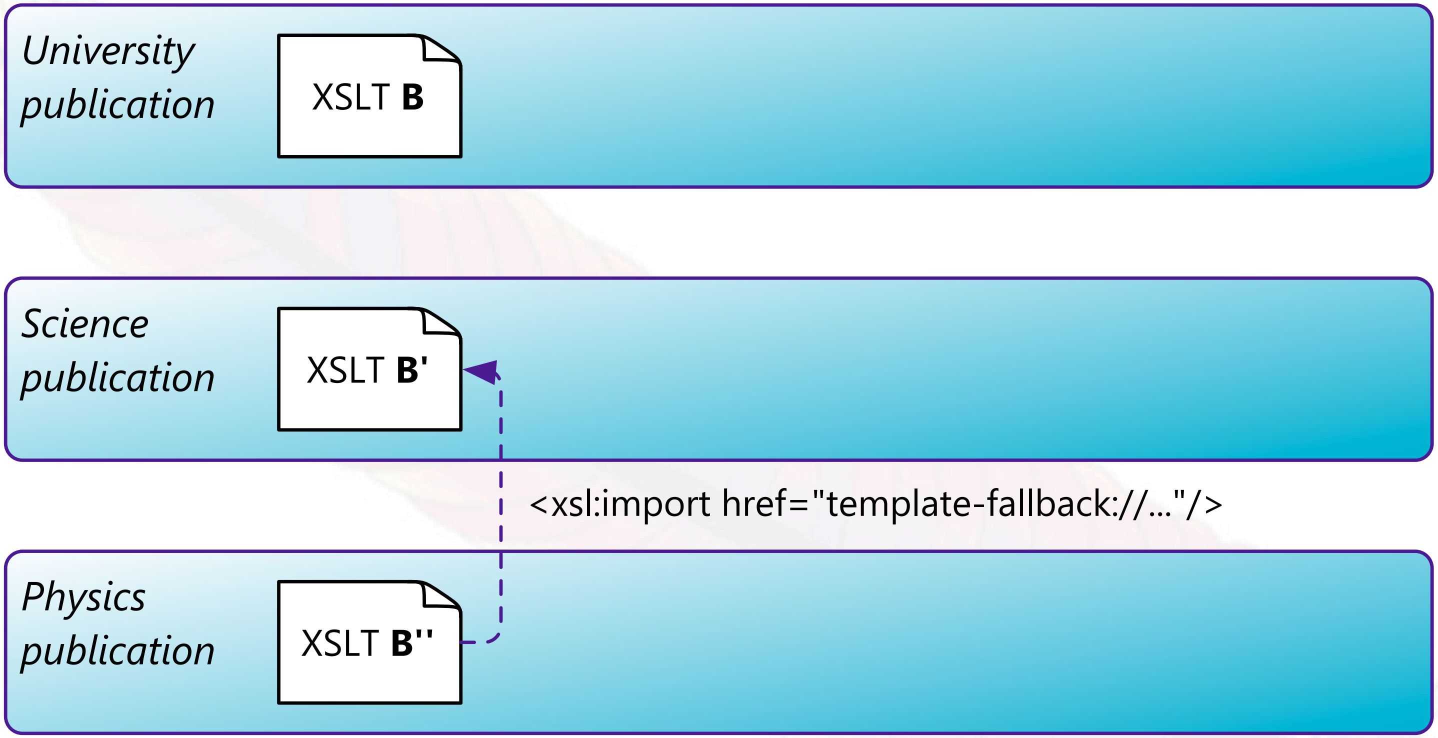
Publication Templating



Fallback


University publication
Sitemap
fallback://
XSLT **B**
Science publication
XSLT **B'**
Physics publication

Template Fallback



University publication

XSLT **B**

Science publication

XSLT **B'**

<xsl:import href="template-fallback://..."/>

Physics publication

XSLT **B''**

Pub. Templating: Summary

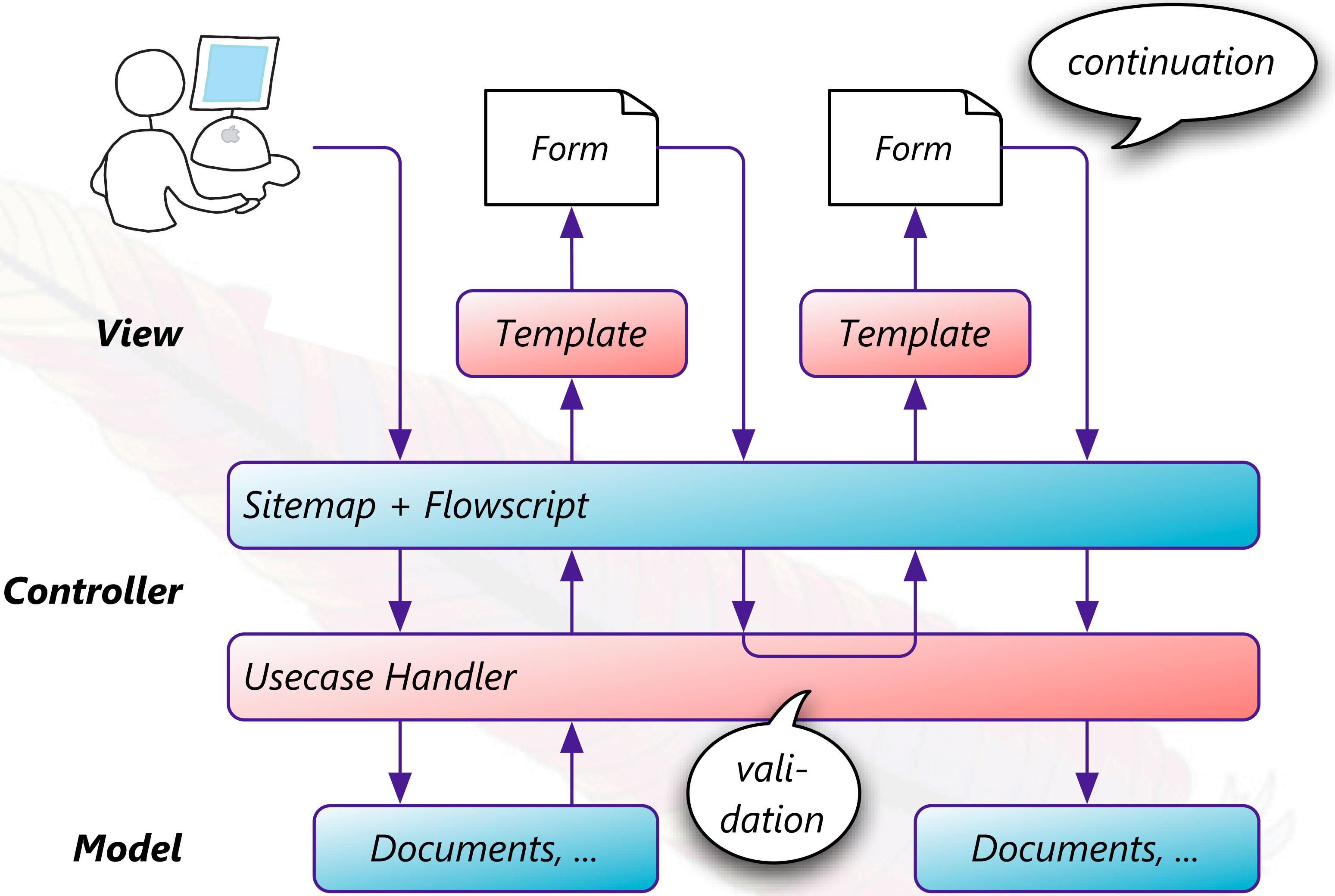
- Purpose:
  - Publication “hierarchies”
  - Inherit resources from templates
  - Override resources
  - Add functionality
- How-To:
  - Use fallback:// to include resources
  - Just put the overriding resources in the equivalent location

Usecase Framework

- Overview
- Declaring the usecase
- Implementing the usecase
- Adding a menu item
- Access control

Usecases: Overview

- User interaction, usually form-based
  - Edit and manage content
  - Interact with other applications (newsletter)
  - Visitor functionality (contact form)
  - ...
- Triggered using a request parameter
- Full screen or document-based



continuation

Form

Form

View

Template

Template

Sitemap + Flowscript

Controller

Usecase Handler

vali-dation

Model

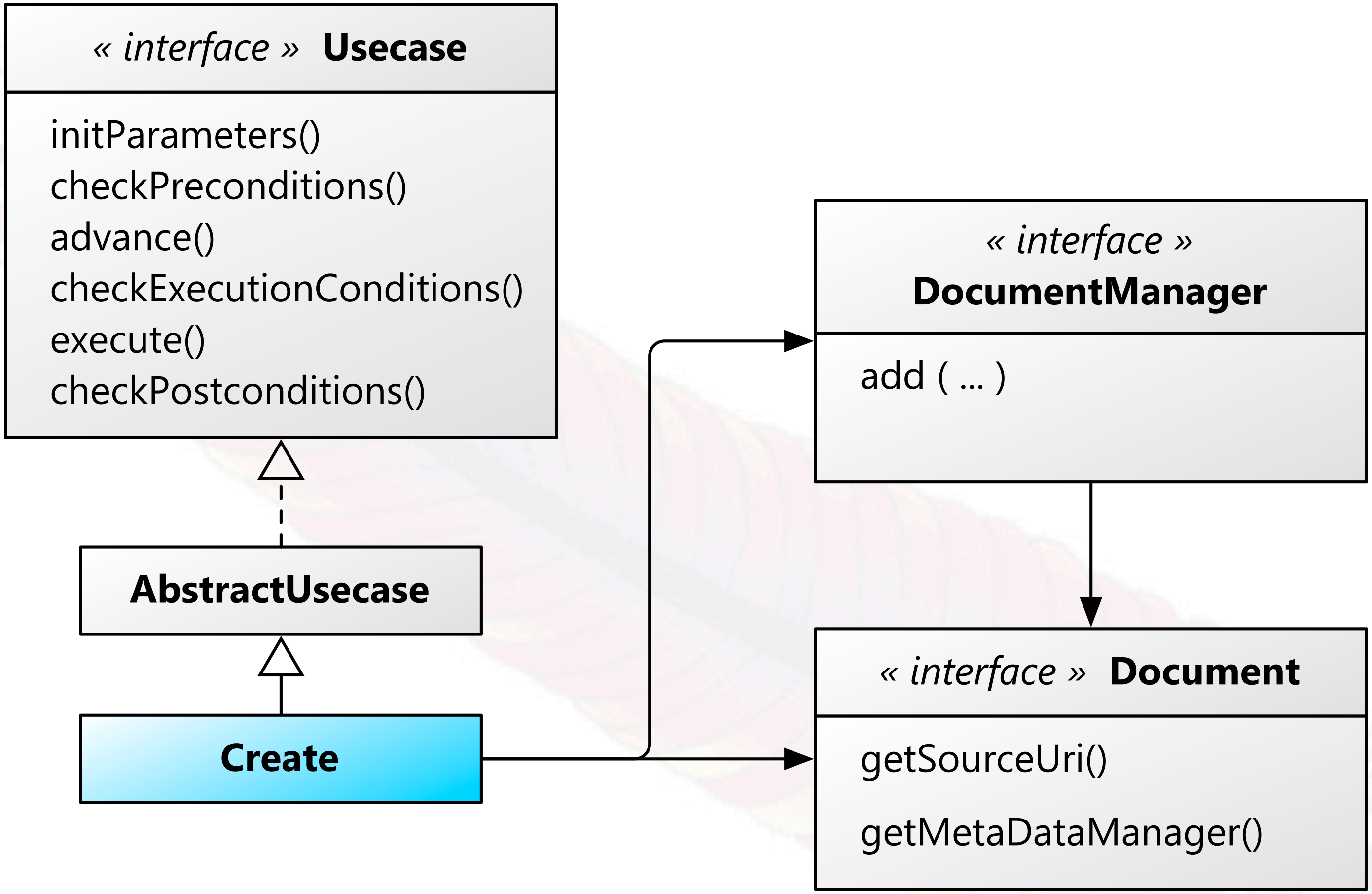
Documents, ...

Documents, ...

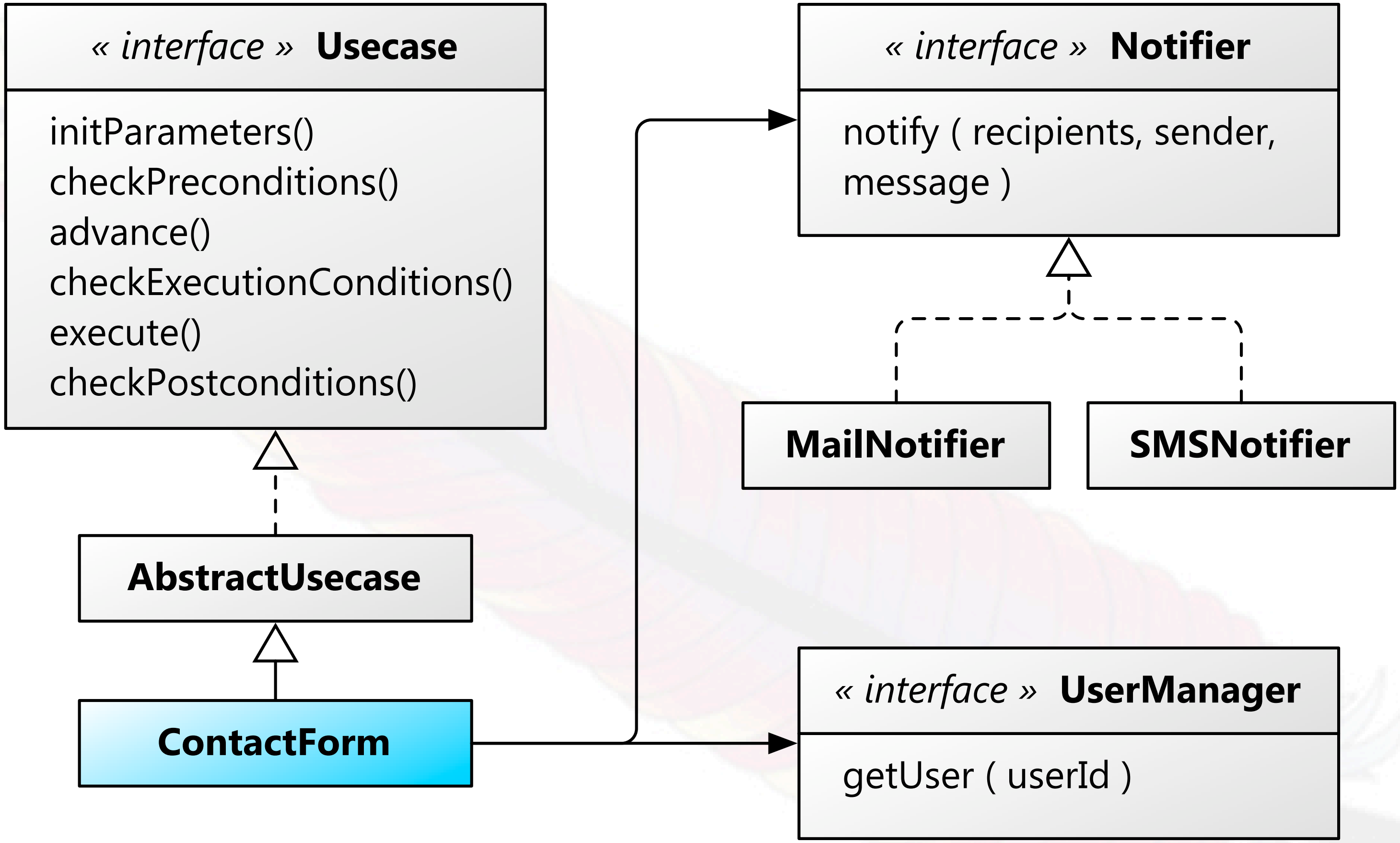
Declaring a Usecase

```
<usecases>
  ...
  <component-instance name="site.create"
    class="o.a.lenya.cms.site.usecases.CreateDocument">

    <view template="modules/sitemanagement/ \
       usecases/site/create.jx"/>

    <parameter name=”...” value=”...”/>
    <parameter name=”...” value=”...”/>

    <transaction policy="optimistic"/>

  </component-instance>
  ...
</usecases>
```

Example: Create Document



Example: Contact Form



Usecase Menu Item

```
<menu>

  ...

  <block>
    <item uc:usecase="site.create" href="?doctype=xhtml">
      <i18n:text>New XHTML Document</i18n:text>
    </item>
  </block>

  ...

</menu>
```

Usecase Access Control

- <pub>/config/ac/usecase-policies.xml

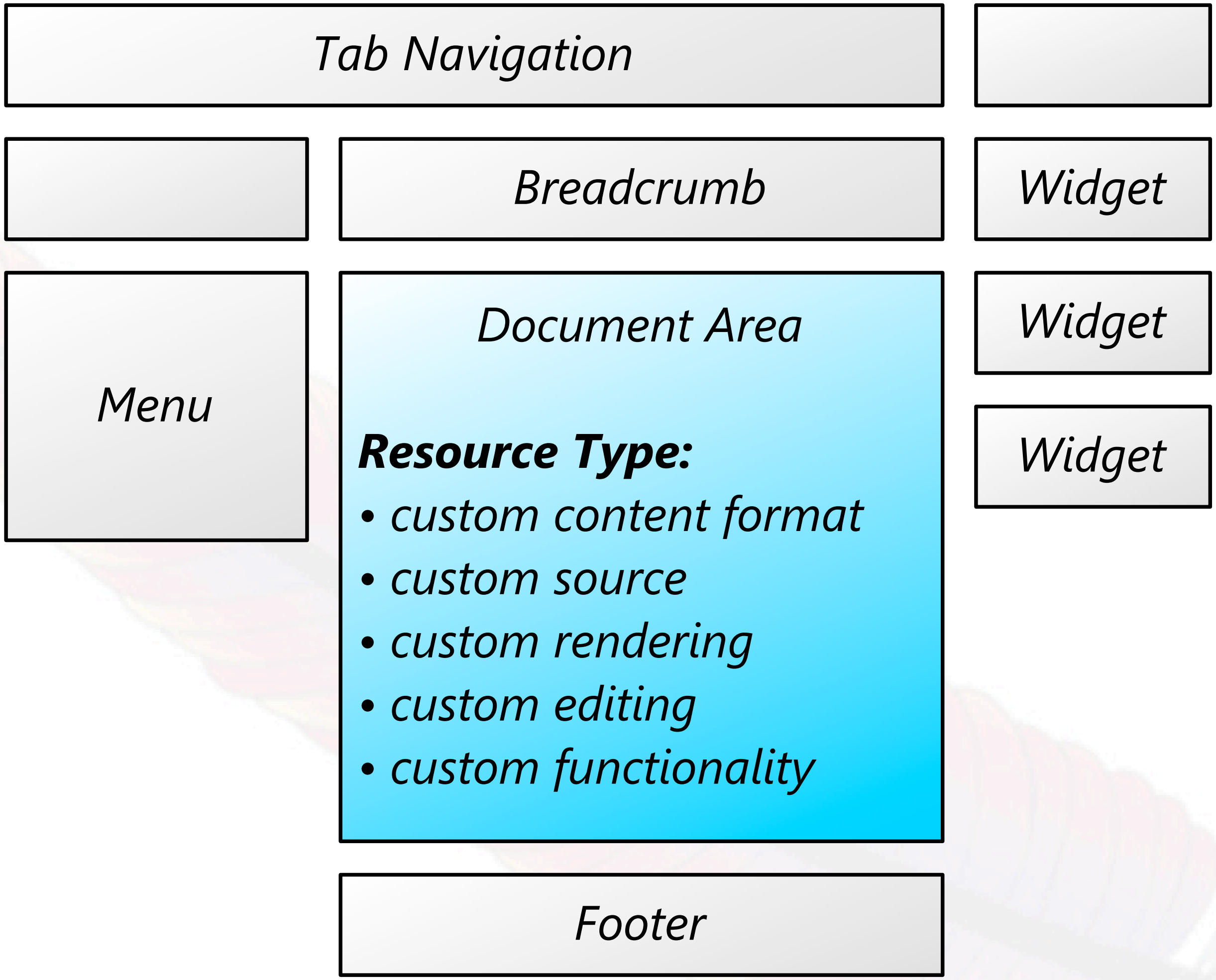
```
<usecases xmlns="...">
  ...
  <usecase id="site.create">
    <role id="admin"/>
    <role id="edit"/>
  </usecase>
  ...
</usecases>
```

Usecases: Summary

- Purpose:
  - User-controlled, GUI-based functionality
- How-To:
  - Declare (patch for cocoon.xconf)
  - Add the usecase handler class
  - Add the view JX template
  - Add policy to usecase-policies.xml

Resource Types

- Page Setup
- Examples
- Implement a Resource Type



Tab Navigation

Breadcrumb

Widget

Menu

Document Area

Resource Type:

- custom content format
- custom source
- custom rendering
- custom editing
- custom functionality

Widget

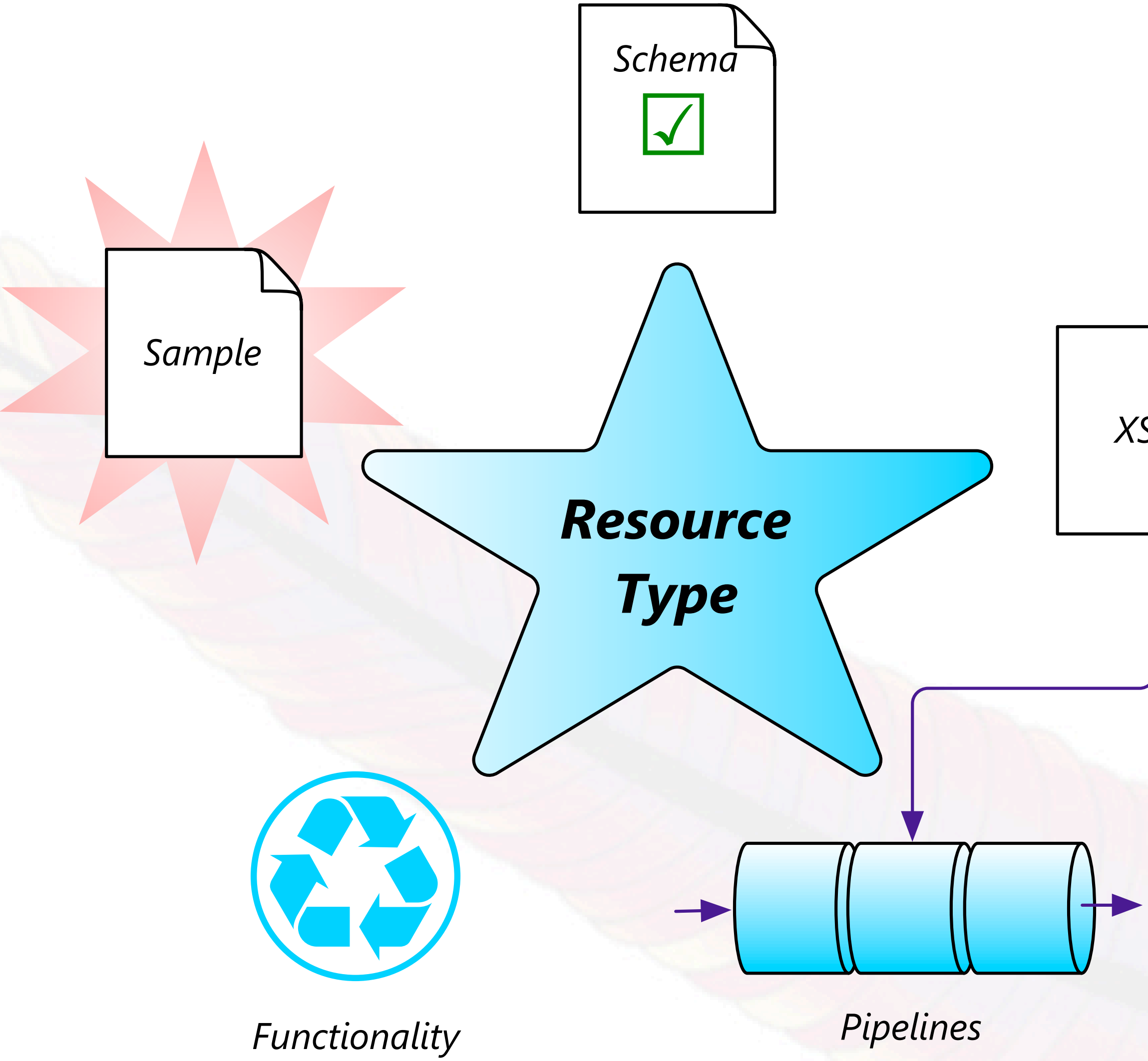
Widget

Footer

Resource Type Examples

- xhtml
- links
- opendocument
- usecasedocument

- *docbook*
- *word*
- *externalpage*


Schema
Sample
XSLT
Resource Type
Functionality
Pipelines

Resource Type Declaration

```
<component-instance name="links">
  <schema
    language="http://relaxng.org/ns/structure/0.9"
    src="fallback://lenya/modules/[...]/links.rng"/>

  <sample-name>[...]/links.xml</sample-name>

  <format name="xhtml"
          uri="cocoon://modules/links/xhtml.xml"/>
  <format name="luceneIndex"
          uri="cocoon://modules/xhtml/index.xml"/>
  <format name="webdavGET"
          uri="cocoon://modules/xhtml/davget.xml"/>

</component-instance>
```

Implement a Resource Type

- Add the resource type module
- Declare the resource type
- Add a validation schema
- Add a sample
- Add the pipelines
- Add the XSLT
- Configure the menu items

Resource Types: Summary

- Purpose:
  - Support specific content types
  - Support presentation formats for content types
- How-To:
  - Add a module
  - Add sample + schema
  - Add pipelines+XSLT for presentation
  - Add functionality

Thank you!