

---

# **All about URIs**

## **or: Find your sources**

### **(protocols from file:// to jcr:// and beyond)**

**Torsten Schlabach**  
**(tschlabach@apache.org)**

**The Cocoon GetTogether, edition 2005**  
**October 7th 2005 - Amsterdam - The Netherlands**

# Agenda

---

- I. Background
  - Sorting out terminology: URI, URL, URN?
  - What's the difference between nominating and locating something?
  - Relevant (and very little known) RFCs (This has all been standardized! Let's start using it!)
- II. State of the art in Cocoon
  - Understanding Avalon sources (the underlying mechanism in Cocoon)
  - What protocols / pseudo-protocols are available in Cocoon today?
  - The jcr:// protocol (JSR-170), how and what to use it for
- III. Potential future directions
  - When to implement a new pseudo protocol (and when to avoid it)
  - Food for thought: Implement a URNSource in Cocoon?

# URx quiz (1)

---

Ask yourself:

- Can you explain the difference between a URI and a URL?
- Ever heard of a URN?

# URx quiz (2)

---

Potential answers:

- URI is newer term for URL!
- I keep seeing the term URI in JavaDocs. So probably URL is deprecated and I got used to talkion about URIs instead. Can't be wrong
- URN? Cool! A new acronym! Let's get excited about it!!!!

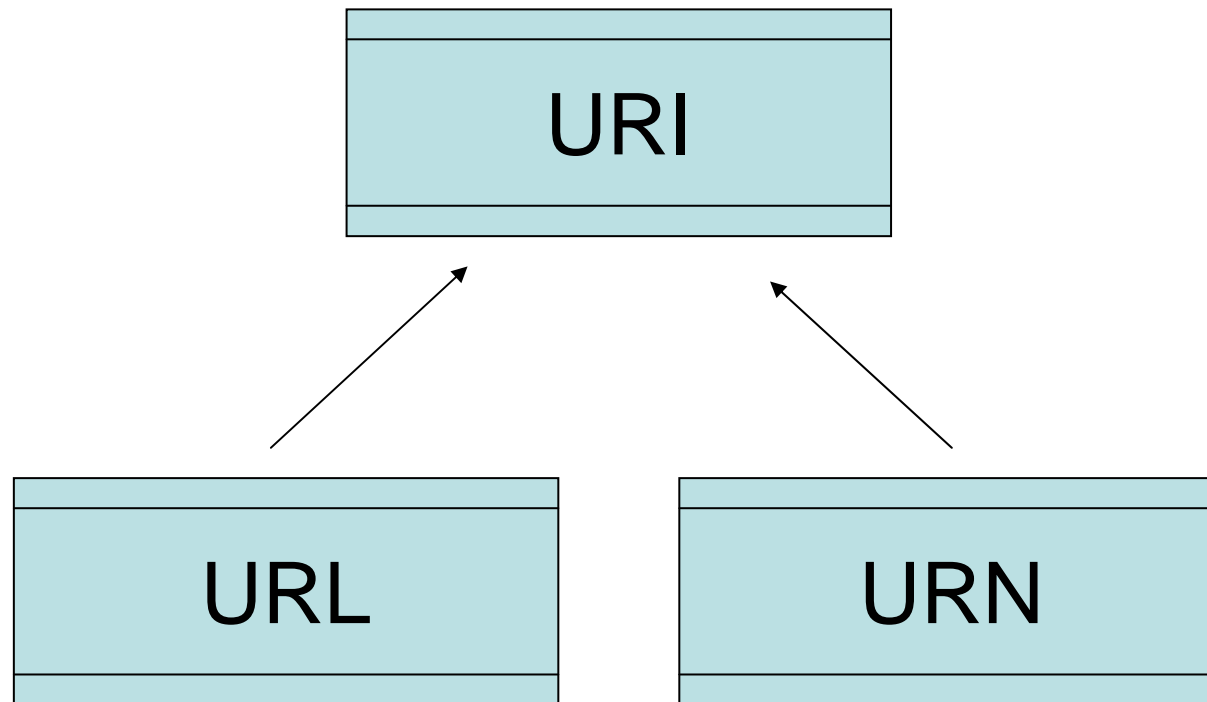
# URx quiz (3)

---

- URL: Uniform Resource Locator
  - locates a resource (says where it is, no matter what it is)
- URI: Uniform Resource Identifier
  - identifies a Resource by whatever means
  - superclass of both URL and URN
- URN: Uniform Resource Nomintator
  - nominates (calls by name = says what it is) a resource (no matter where it is)

# In other words ...

---



# Nominating versus Locating

---

## Locating:

- The guy on the 3rd seat conted from right seen from the audience, please come to the stage.
- Give me the 9th book from the 2nd row on the 4th shelf, please

## Nominating:

- Could Peter Miller please come to the stage?
- I need the „What is Lojban“ book, please!

# Anatomy of a URN

---

<URN> ::= "urn:" <NID> ":" <NSS>

URN: Makes the URI a URN, not a URL

<NID>: Namespace identifier, rules the interpretation of the <NSS> part

<NSS>: Namespace specific string

Examples:

- URN:ISBN:0-743-45730-7
- urn:publicid:-:OASIS:DTD+DocBook+XML+V4.1.2:EN  
(-//OASIS//DTD DocBook XML V4.1.2//EN)

Is jdbc:mysql:... a URN?

# RFCs and IANA

---

- RFC 2141: URN Syntax  
Uniform Resource Names (URNs) are intended to serve as persistent, location-independent, resource identifiers.
- RFC 2611: URN Namespace Definition Mechanisms  
Uniform Resource Names (URNs) are resource identifiers with the specific requirements for enabling location independent identification of a resource, as well as longevity of reference.

Vertical examples:

- RFC 3151: A URN Namespace for Public Identifiers  
How to locate DTDs by public identifier
- RFC 2288: Using Existing Bibliographic Identifiers as Uniform Resource Names  
How to locate books or journals by defining an isbn and/or issn namespace, i.e. urn:isbn:... or urn:issn:...

- More officially registered namespaces at:

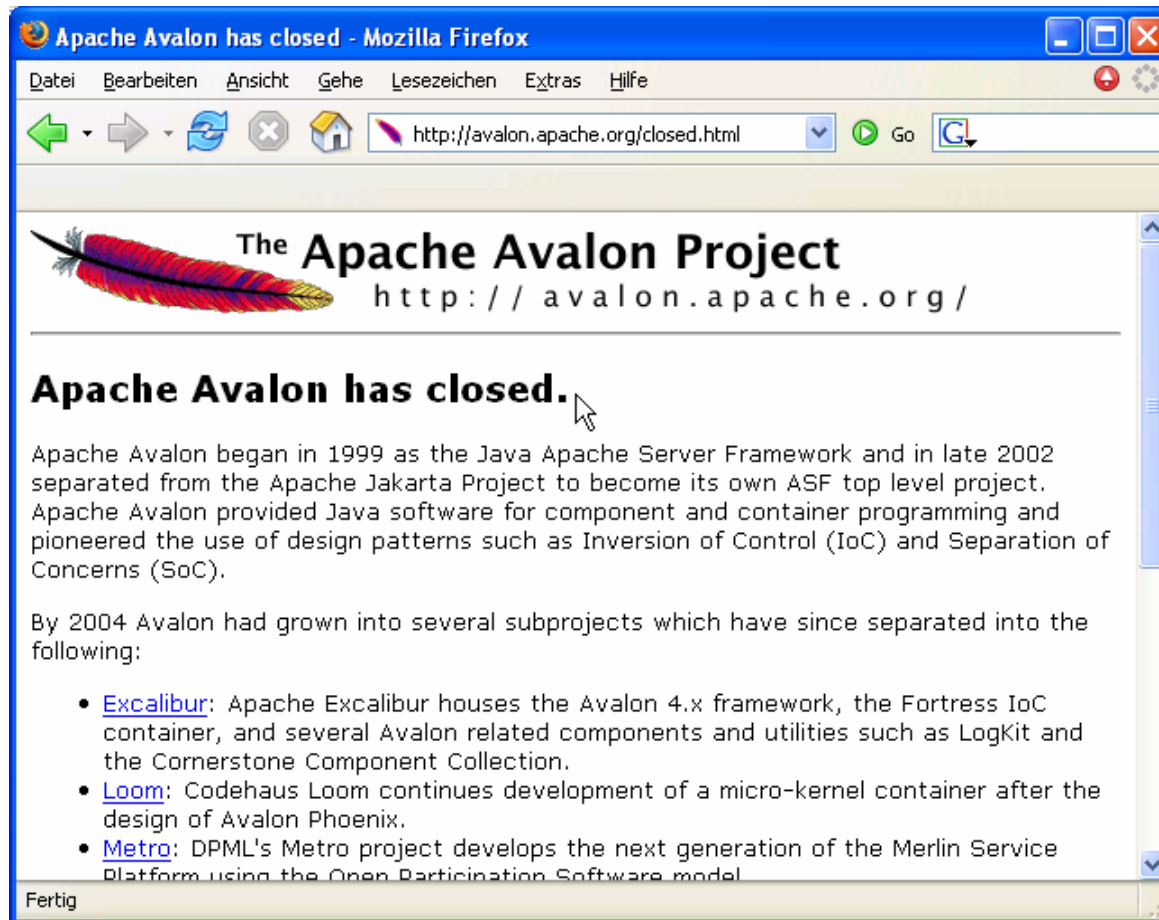
<http://www.iana.org/assignments/urn-namespaces>

# URI handling in Cocoon

---

- URIs appear in
  - src= attributes in the sitemap
  - href= attributes in SAX events (documents)
  
- URIs are handled by:
  - The Java (JRE)
  - Avalon Sources

# Apache Avalon has closed.



# ... but don't get that wrong!

---

- The condemned live longer!
  - The Excalibur project carries on ... Well, somehow (see: <http://excalibur.apache.org/>)
  - Cocoon is based on Excalibur libraries and Avalon concepts and this is not going to change in the short run
  - Does anyone in the audience know any other significant project that uses Excalibur and Avalon?

# Avalon Source Factories

---

org.apache.excalibur.source

## Source

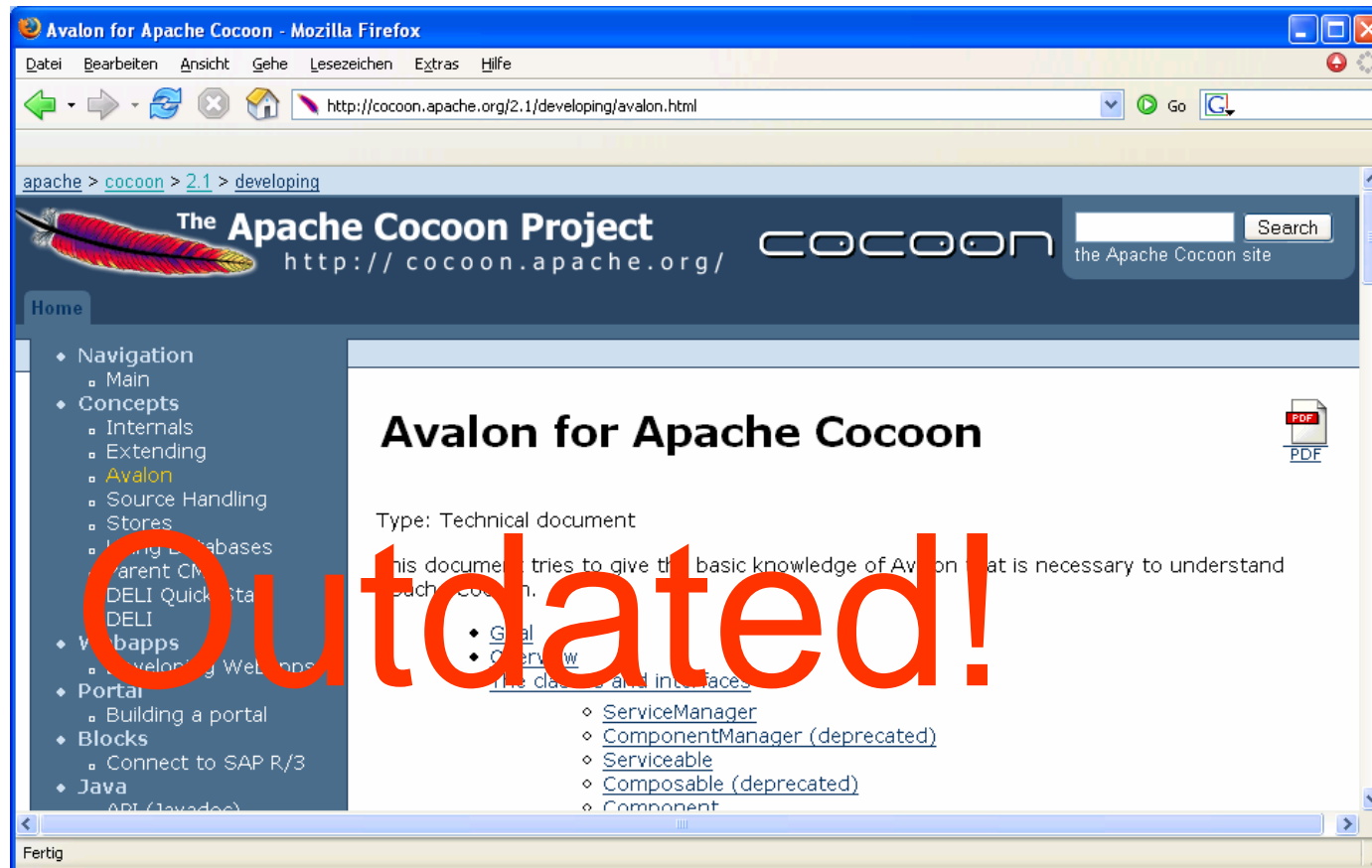
- Provides a Java process with access to a distinct, readable (sometimes writeable), concrete resource
- Has *read* (and optionally *write*) as well as *close* methods

org.apache.excalibur.source

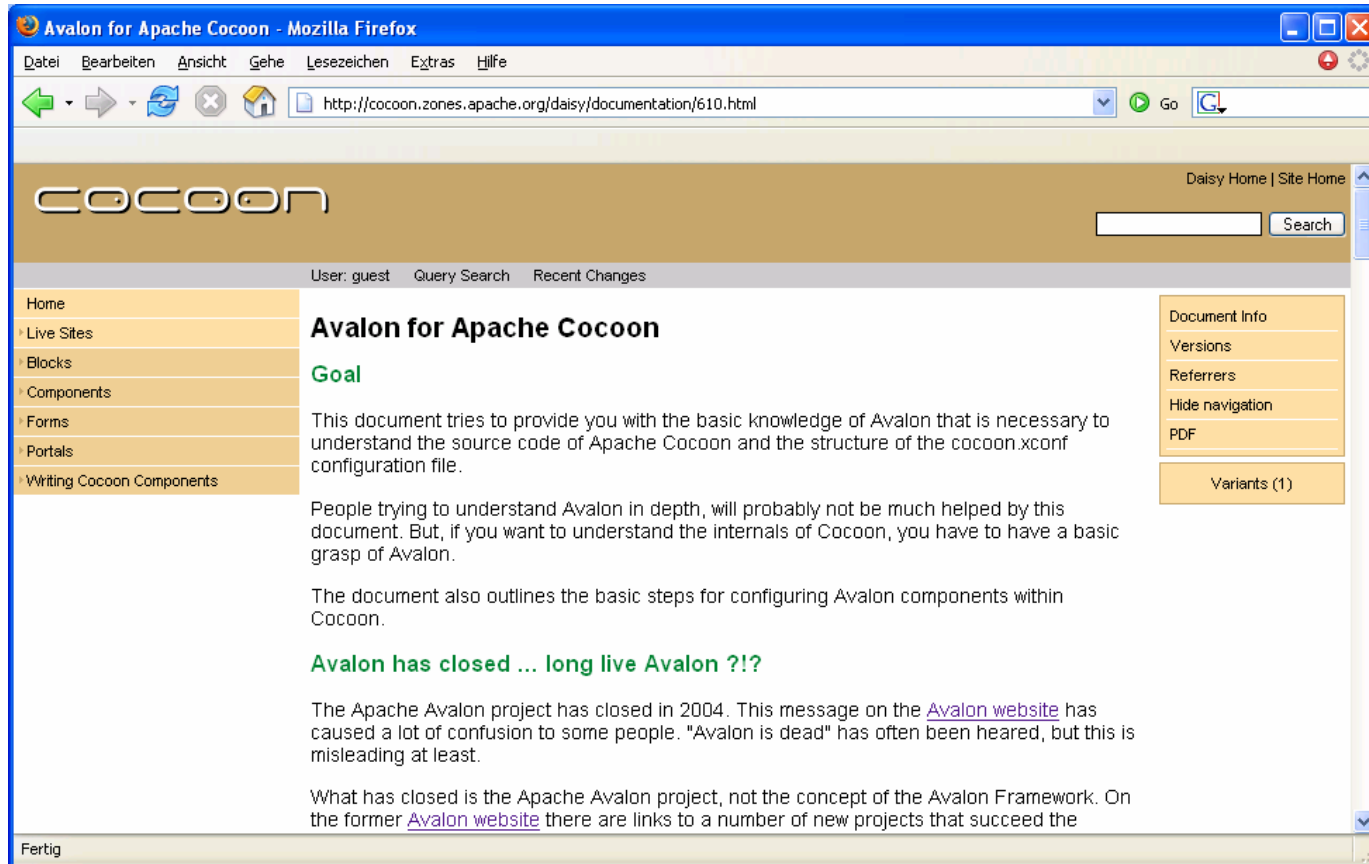
## SourceFactory

- Needs a URI as a parameter
- Creates an instance of a source accessing a distinct URI
- Might delegate the source creation to a different source factory (context -> file)

# More on Avalon related to Cocoon



# Even more on Avalon related to Cocoon



<http://cocoon.zones.apache.org/daisy/documentation/610.html>

The **Cocoon GetTogether**, edition 2005: *October 7th 2005 - Amsterdam - The Netherlands*

<http://www.cocoongt.org/>

# On the spot information on Avalon related to Cocoon

---

cocoon.xconf:

```
1.      <!--+
2.          | Source Factories
3.          |
4.          | Each source factory adds a special uri schemes to the system.
5.          +-->
6.      <source-factories>
7.          <component-instance class="org.apache.excalibur.source.impl.ResourceSourceFactory" name="resource"/>
8.          <component-instance class="org.apache.cocoon.components.source.impl.ContextSourceFactory" name="context"/>
9.          <component-instance class="org.apache.cocoon.components.source.impl.SitemapSourceFactory" name="cocoon"/>
10.         <!-- the "*" protocol handles all uri schemes that are not explicitly specified. This includes all
11.             JDK standard protocols, with special handling of the "file:" protocol which is modifiable
12.             (can be written to) and traversable (directory structures can be crawled). -->
13.         <component-instance class="org.apache.excalibur.source.impl.URLSourceFactory" name="*/>

14.     <!-- pseudo protocol for Jakarta Slide repositories -->
15.     <component-instance class="org.apache.cocoon.components.source.impl.SlideSourceFactory" name="slide"/>

16.     <!-- xmldb pseudo protocol -->
17.     <component-instance class="org.apache.cocoon.components.source.impl.XMLDBSourceFactory" name="xmldb">
18.         <!-- Xindice driver -->
19.         <driver class="org.apache.xindice.client.xmldb.DatabaseImpl" type="xindice"/>
20.         <!-- Add here other XML:DB compliant databases drivers -->
21.     </component-instance>
```

# Existing protocols in Cocoon

---

- resource://
- context://
- cocoon://
- any protocol implemented in the JRE, especially file:// and http://

# Other protocols in Cocoon

---

- Stable (or „overly“ stable):
  - slide://
  - xmldb://
  - xindice://
- Evolving:
  - jcr:// (see Jackrabbit, JSR-170)

# Reminder:

## Source versus Generator

---

### Source

- Provides a Java process with physical access to a byte stream somewhere on this planet
- There are read-only and writeable (modifyable) sources

### Generator

- Generates SAX Events from a non-XML byte stream
- Makes non-XML input accessible to XML transformations and serialisations
- Generators are never writable

# RFyC:

## Do we need an URN source?

---

- A sub-framework, plugging into Cocoon as a SourceFactory for the „urn“ protocol
- Providing means to plug in NSS (Namespace Specific String) resolvers
- Isolate the resolving of nomintator to a locator
- There is a problem with

```
<source-factories>
```

```
  <component-instance
```

```
    class="org.apache.excalibur.source.impl.UrnIsbnSourceFactory"
```

```
    name="URN:ISBN"/>
```

# What problems do URNs solve?

---

- Cleaner sitemaps:
  - Focus on content versus implementation
  - Hide content storage implementation details (*think of migrating content from a filesystem into a JCR for example*)
  - Longevity of references (*links between documents as well as deep links into your content*)
- Save not only lots of work but also prevent lots of broken links and 404s in the future

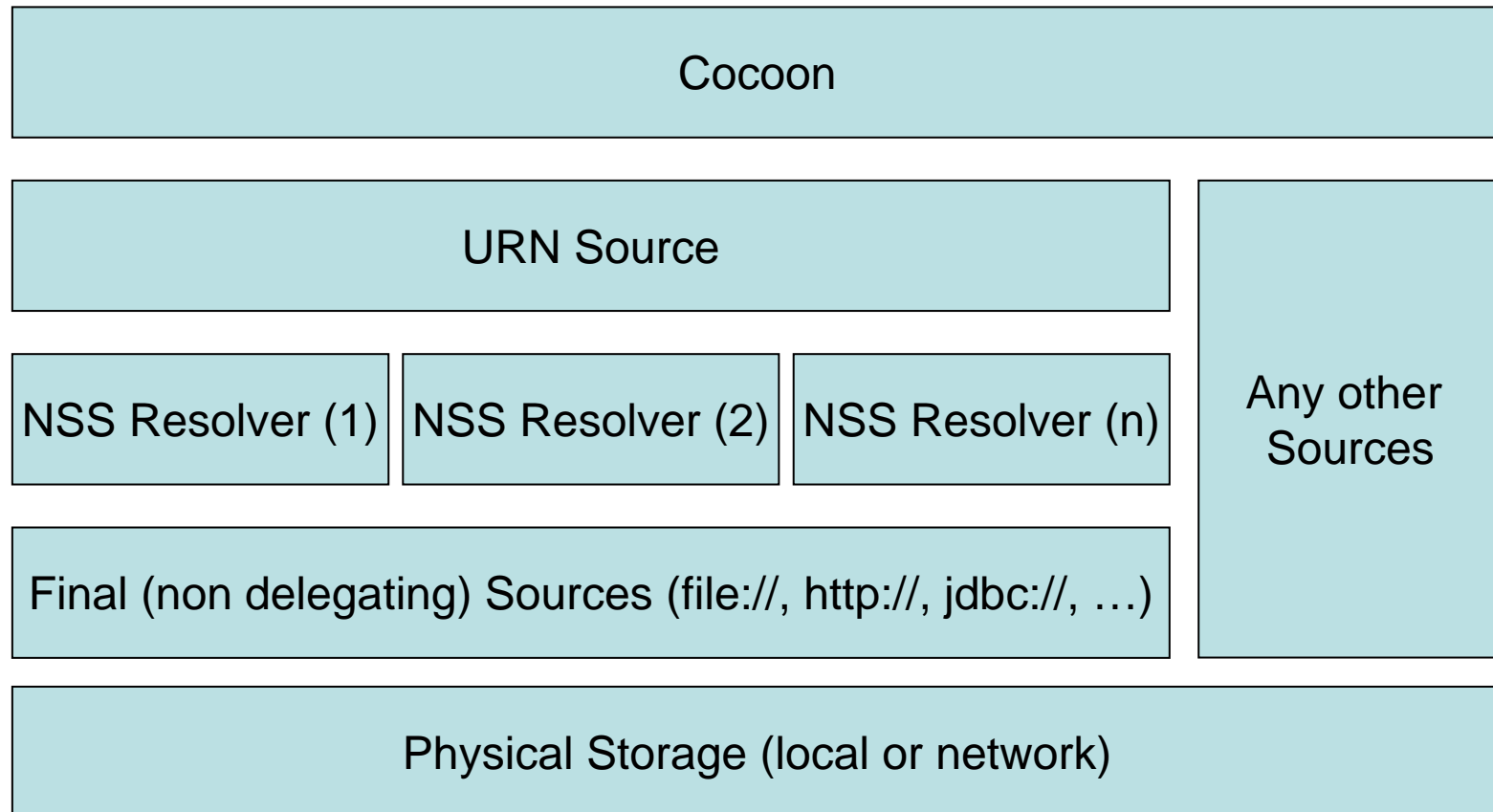
# What problems do URNs introduce?

---

- They need to be implemented
  - This takes a well trained Java programmer with some background in Cocoon internals, Avalon/Excalibur and probably some networking background
- They need to be well standardized before they get implemented
  - This might require working with standard bodies including but not limited to W3C, IETF, IEEE, ISO, the EU Commission, industry and company specific standards boards, etc.
- They add complexity to the Cocoon configuration

# Suggested anatomy of a URN source

---



# tbd: What does the source deliver?

---

<map:generate src=„URN:ISBN:0-743-45730-7“ />

- Content related metadata such as
  - Author
  - Publisher
  - Year
  - Edition
  - ...
- Commercial metadata such as
  - Price
  - Availability
- Other relating information such as
  - Reviews
- The content of the book?
  - as ASCII text
  - as HTML
  - as DocBook?
  - as FOP source?

# I don't know either

---

