<Cocoon Day="2003-Nov-18" />

# Introducing Apache Cocoon

**Matthew Langham**

Leiter
Competence Center Open Source
S&N AG

# The Project

- Origins
  - Started by Stefano Mazzocchi
    - Redesign of Apache.org
  - Frustrated by the limitations of HTML
  - Wanted to use emerging technologies (XML/XSL)
- Today
  - Now one of the most important Apache projects
  - Incorporates technologies from various projects
  - Large community
  - Large commercial adoption
- Open Source

# Open Source

Timeline

- 1979 Sendmail – Eric Allmann
- 1983 GNU – RIchard Stallmann
- 1986 PERL
- 1991 Linux
- 1994 Red Hat founded
- 1995 Apache Web Server
- 1998 Netscape Mozilla
- 1998 „Open Source“

- “Open Source” coined in 1998
  - Netscape releases source code
- “Free software” has been around since at least the 1980’s
- Today “Open Source” is used liberally
  - Different licences
  - Different availability of source and binaries
- Licences
  - Main aspect when using commercially
  - At least 30 different licences
  - Cocoon is released under the Apache Software Licence

# What is Cocoon?

- In A Nutshell
  - Extensible Java framework for XML based publishing
  - Uses XSLT for multi-channel publishing
  - Allows the integration of various data-sources using provided components
  - Allows processing, aggregation of data
  - Additional components provide portal, authentication, forms, flow..
  - Drop in new components to extend functionality
- Usage scenarios
  - Web Sites
  - Publishing of Data (Web Printing)
  - Portals
  - XML Processing architectures
  - Just about anything really….

# Why is Cocoon different?

- Completely based on XML
- Separation of Concerns (SoC)
  - Site Administrator
  - Content Deployer
  - Layout Deployer
- Provides for many (all?) aspects of today's Web applications
  - Publishing, Aggregation Portals, Form handling
- Extensibility
  - Protects your investment
- Healthy Community

# Why is Cocoon different?

- Document generation on the server
  - Logical names used „http://myserver/cocoon/helloworld“
  - Powerful Process Description
  - Caching for performance
- Flexible Data Integration / Aggregation
  - XML files, XML over HTTP
  - Databases, LDAP, SAP
  - …
- Flexible Publishing to different formats using XSLT
  - HTML, WML, XML
  - PDF, SVG, PS, Office Documents
  - …

# Installation

- What do you need
  - A computer **J**
  - Java JDK
  - An Internet connection
    - Or some way of getting the distribution
- Download the distribution from an Apache mirror
  - Includes Jetty as a servlet engine
  - `$> ./build.sh`
  - `$> ./cocoon.sh servlet`
  - Enjoy

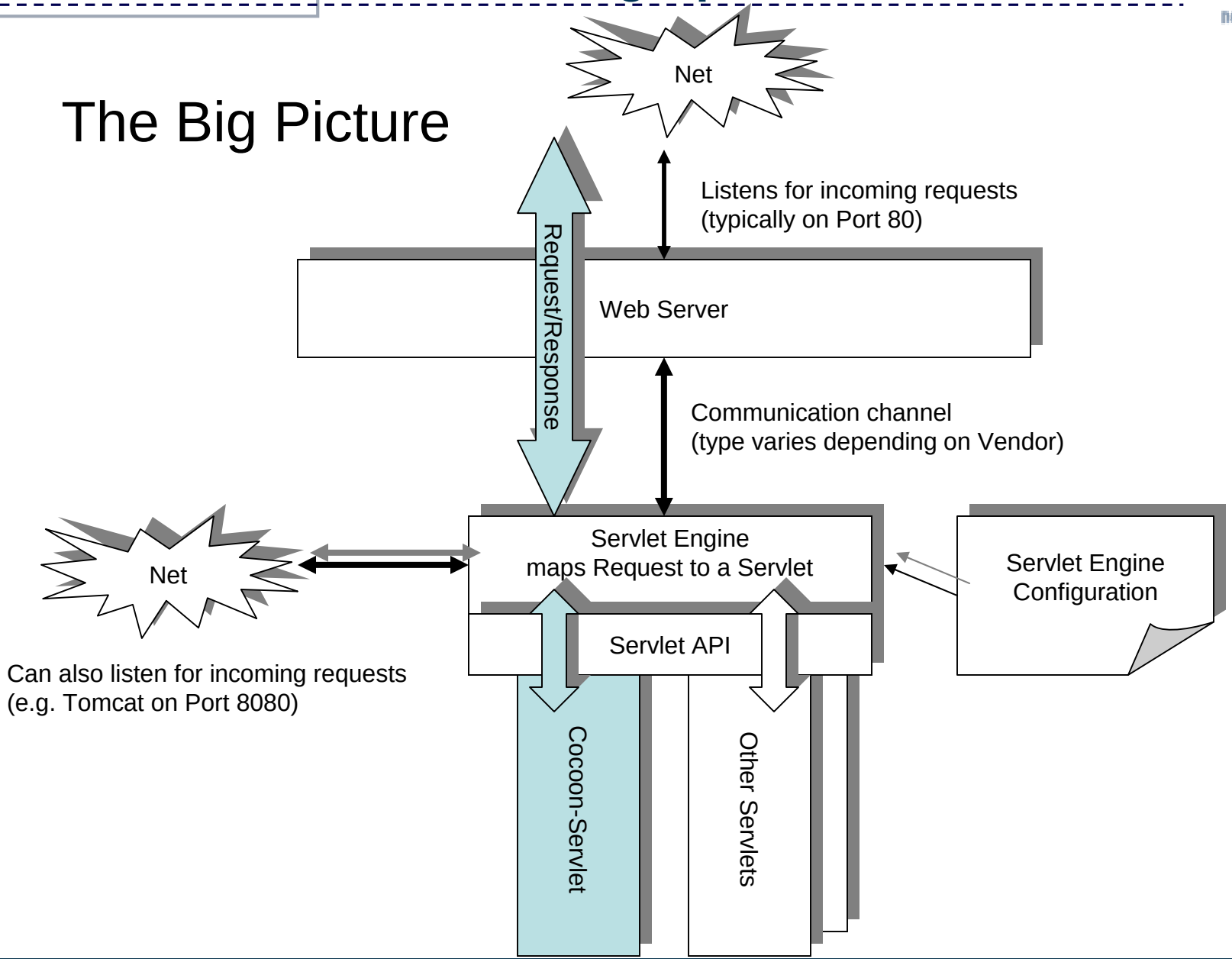
# Introducing Apache Cocoon

Finished!

Well....ok.....not quite....

# Introducing Apache Cocoon

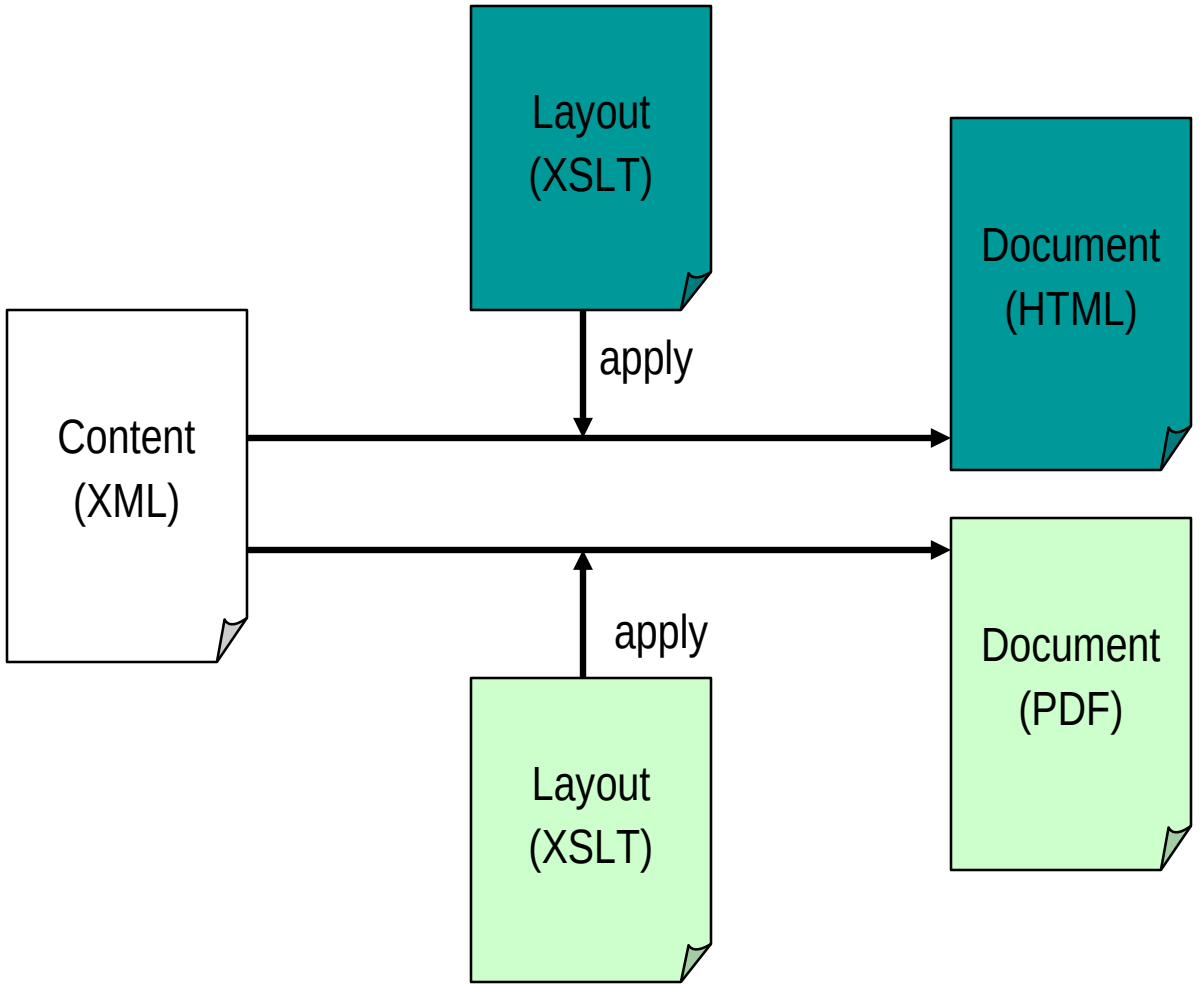
## The Big Picture



# Documentation Generation

- Dynamic Document Generation
  - Based on Request-Response-Cycle (when run as a Servlet)
    - Other Environments possible
  - By defining a processing pipeline per document
    - Getting Content
    - Adding Layout
    - Sending Response
  - Extensible by Logic

# Separating Content and Layout



# Putting Cocoon to work

- Hello World example
  - Content: a simple static XML file
  - Layout: an XSLT Stylesheet generating HTML

# Putting Cocoon to work

- Content: Hello World XML File

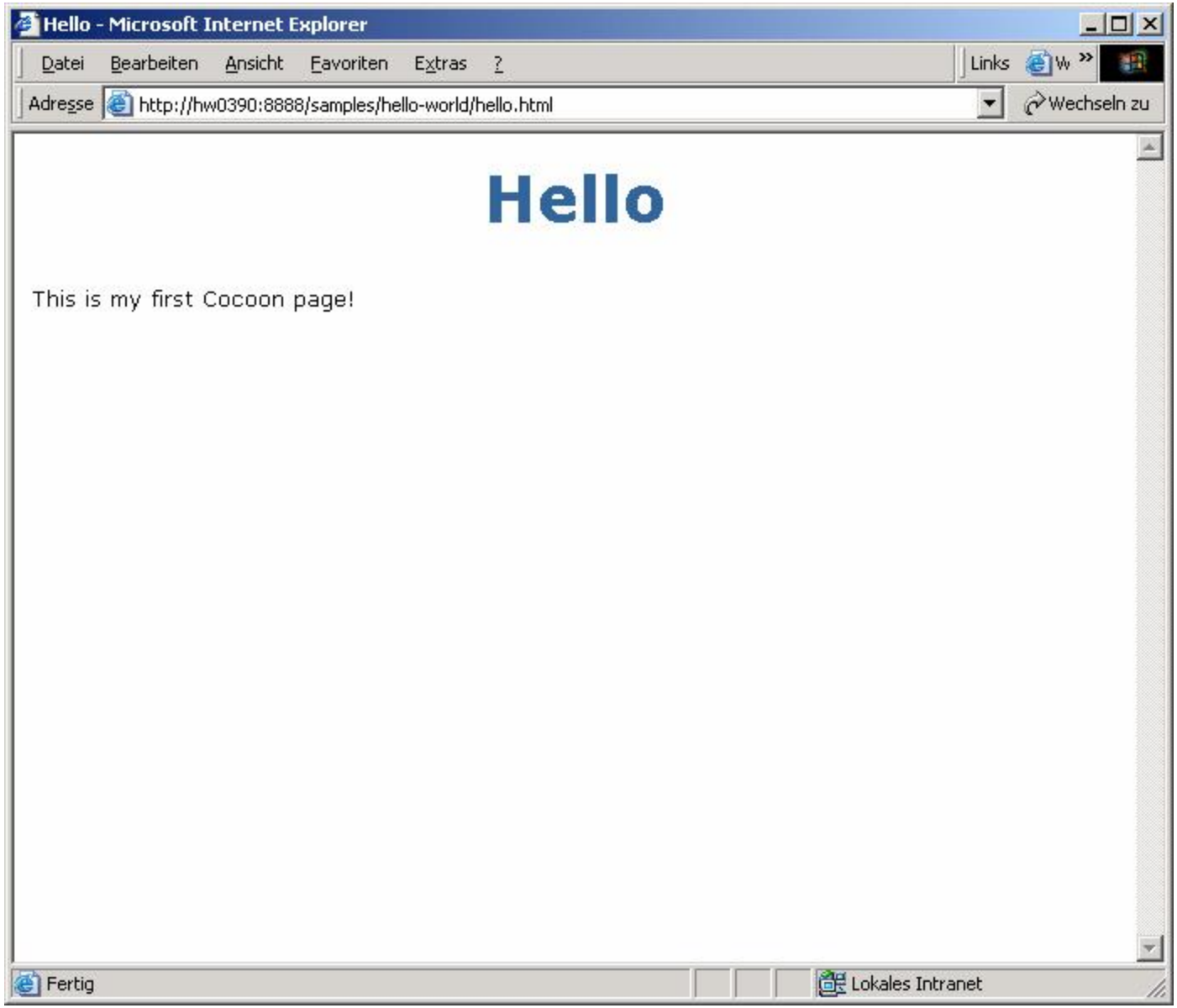
```
<?xml version="1.0"?>
<document>
  <title>Hello</title>
  <text>This is my first Cocoon page!</text>
</document>
```

# Putting Cocoon to work

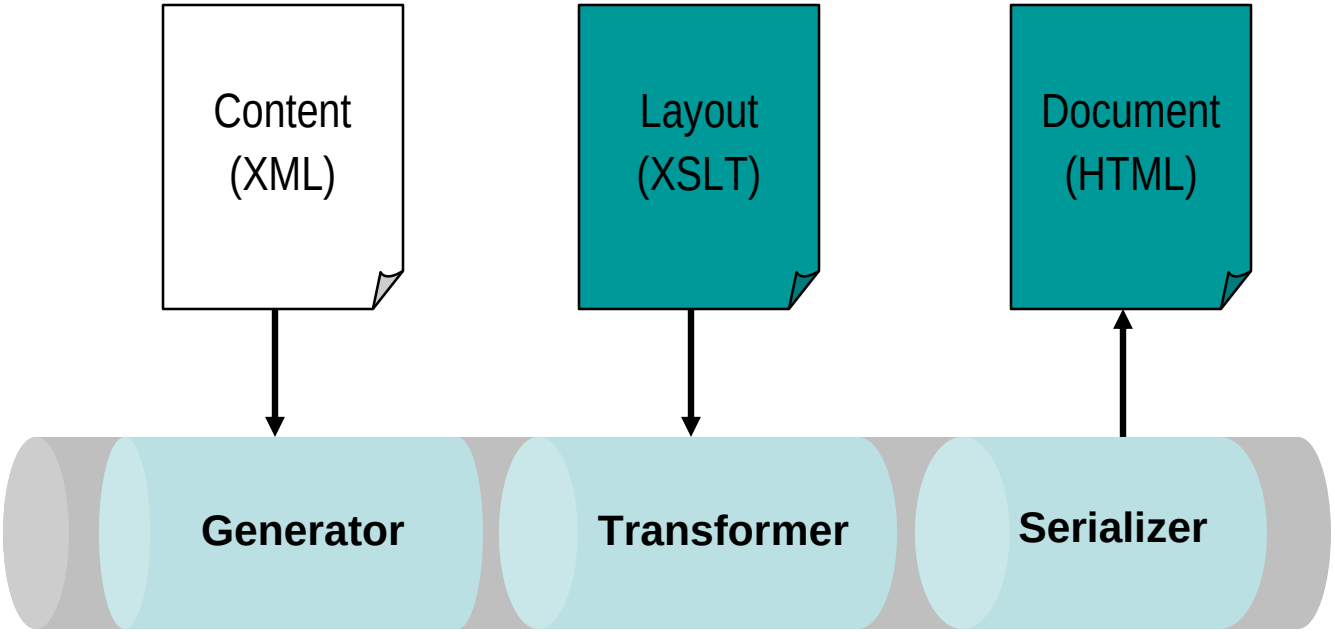
- Layout: The XSLT Stylesheet File

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="document">
  <html>
    <body>
      <h1><xsl:value-of select="title"/></h1>
      <p><xsl:value-of select="text"/></p>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

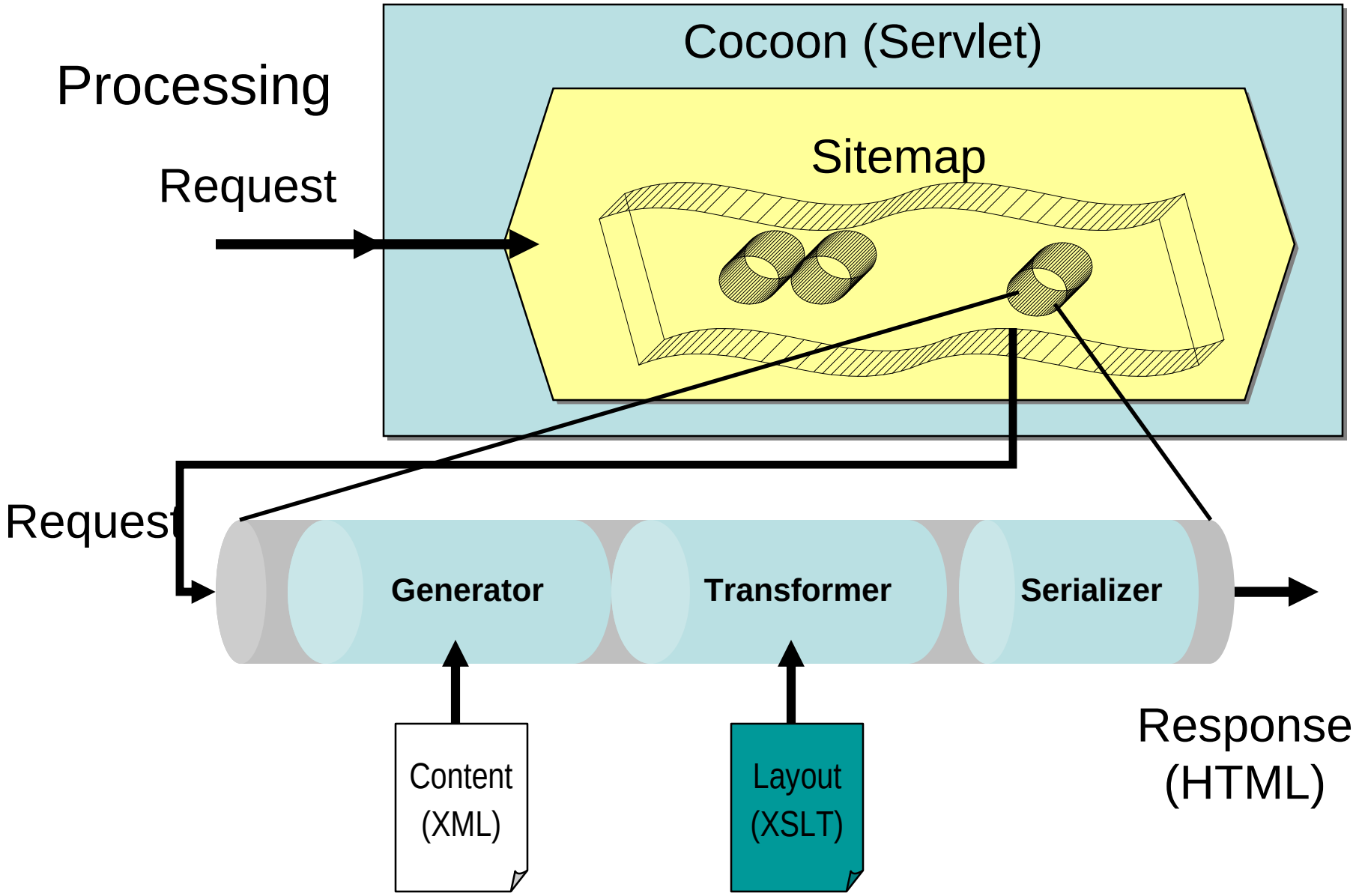
# Introducing Apache Cocoon



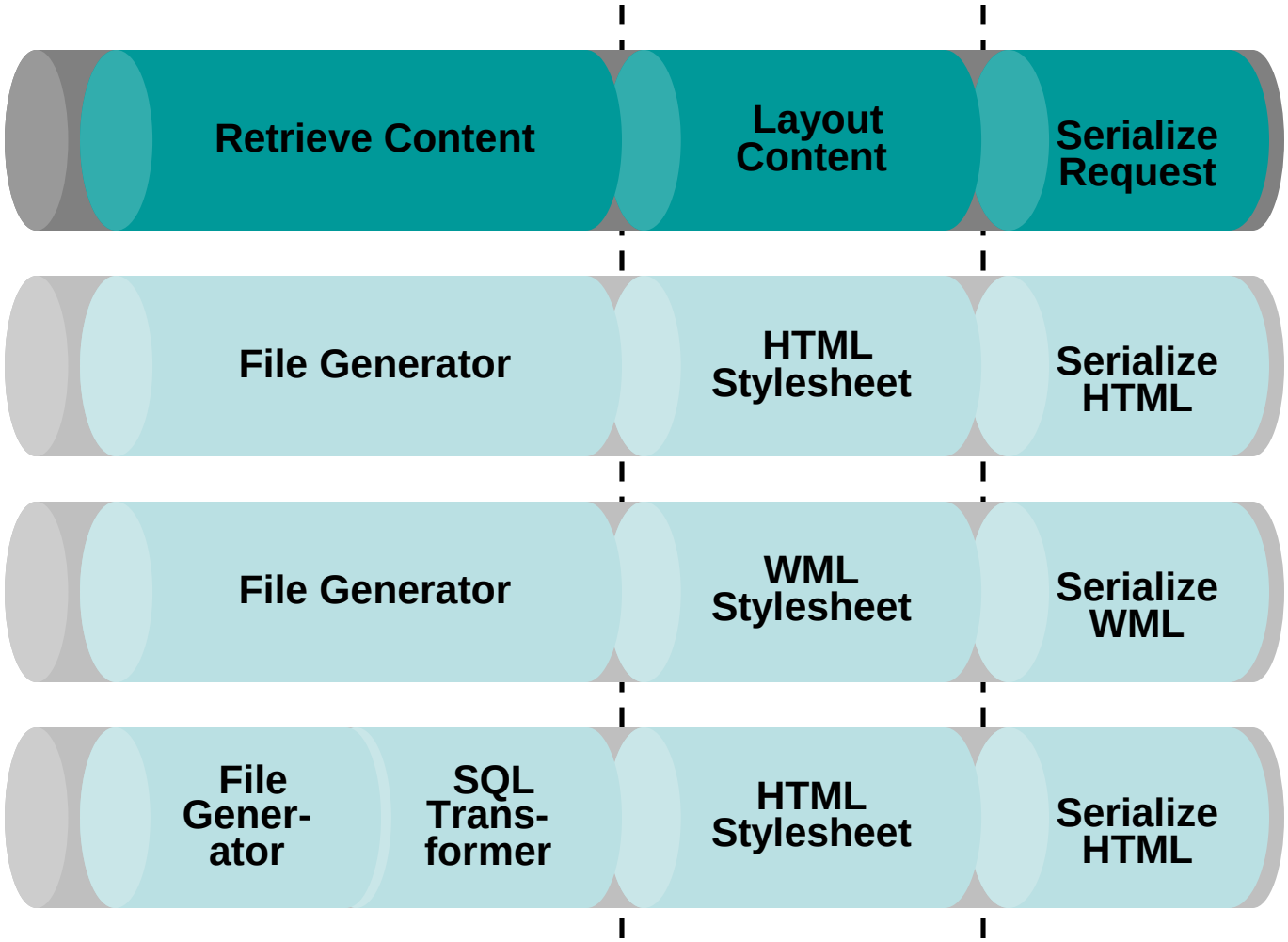
# Building a Pipeline



# Introducing Apache Cocoon



# Separating Content and Layout



# Cocoon Basics

- Sitemap
  - Centralized Configuration File in XML
    - Sub-sitemaps possible
  - Contains set of ready-to-use components
    - Use them to build functions
    - Write and „drop-in“ your own
  - Contains collection of Pipelines (Functions)

```
<map:sitemap>
  <map:components>
    <map:generators/>
    <map:transformers/>
    <map:serializers/>
    <map:readers/>
    <map:selectors/>
    <map:matchers/>
    <map:actions/>
    <map:pipes/>
  </map:components>
  <...>
  <map:pipelines>
    <map:pipeline/>
    <map:pipeline/>
    <...>
  </map:pipelines>
</map:sitemap>
```

# Defining a Pipeline

- Sitemap is an XML document
  - Describes the processing steps for a request
  - Defines the pipeline
    - One Generator
    - 0-n Transformer
    - One Serializer

```
<map:generate  type="file" src="helloworld.xml"/>
<map:transform type="xslt" src="helloworld2html.xsl"/>
<map:serialize type="html" />
```

# Defining a Pipeline

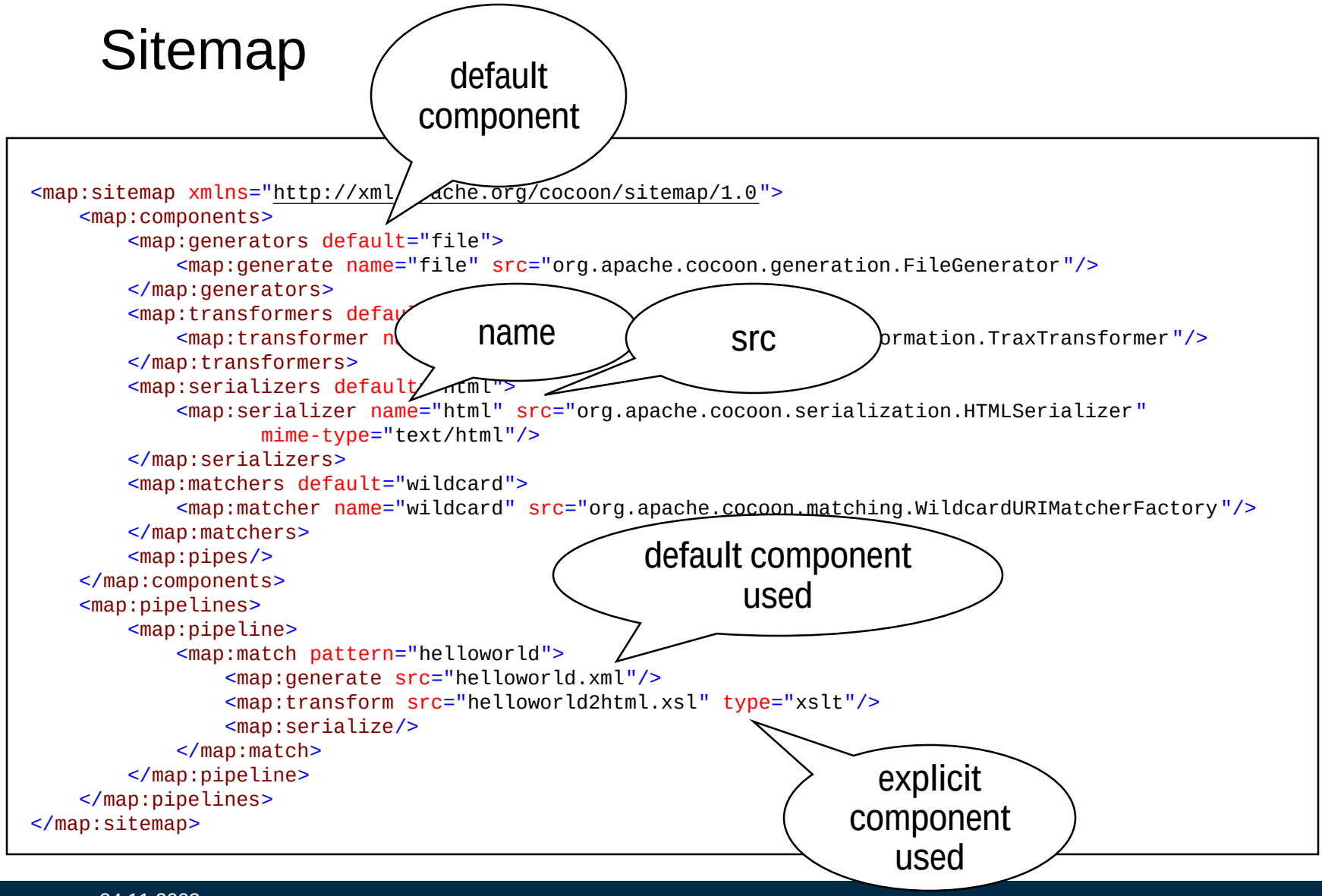
- Sitemap is an XML document
  - Describes the processing steps for a request
  - Matches a pipeline to a request URI

`http://localhost:8888/helloworld`

```
<map:match pattern="helloworld" type="wildcard">
     <map:generate  type="file" src="helloworld.xml"/>
     <map:transform type="xslt" src="helloworld2html.xsl"/>
     <map:serialize type="html"/>
</map:match>
```

# Introducing Apache Cocoon

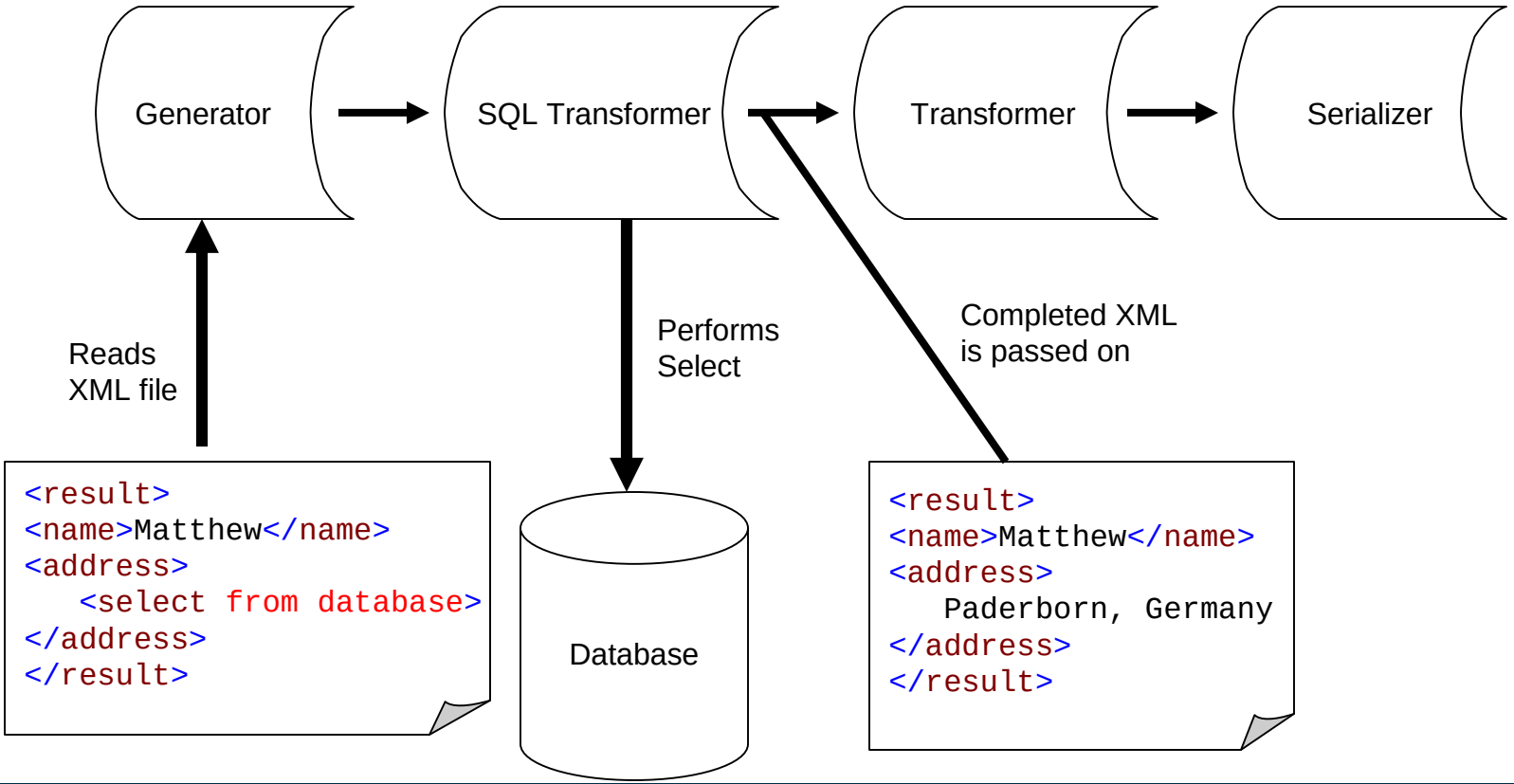
## Sitemap




```
<map:sitemap xmlns="http://xml.apache.org/cocoon/sitemap/1.0">
    <map:components>
        <map:generators default="file">
            <map:generate name="file" src="org.apache.cocoon.generation.FileGenerator"/>
        </map:generators>
        <map:transformers defau
            <map:transformer n                    ormation.TraxTransformer"/>
        </map:transformers>
        <map:serializers default       tml">
            <map:serializer name="html" src="org.apache.cocoon.serialization.HTMLSerializer"
                  mime-type="text/html"/>
        </map:serializers>
        <map:matchers default="wildcard">
            <map:matcher name="wildcard" src="org.apache.cocoon.matching.WildcardURIMatcherFactory"/>
        </map:matchers>
        <map:pipes/>
    </map:components>
    <map:pipelines>
        <map:pipeline>
            <map:match pattern="helloworld">
                <map:generate src="helloworld.xml"/>
                <map:transform src="helloworld2html.xsl" type="xslt"/>
                <map:serialize/>
            </map:match>
        </map:pipeline>
    </map:pipelines>
</map:sitemap>
```

# Complex Pipeline

- Transformer example with commands



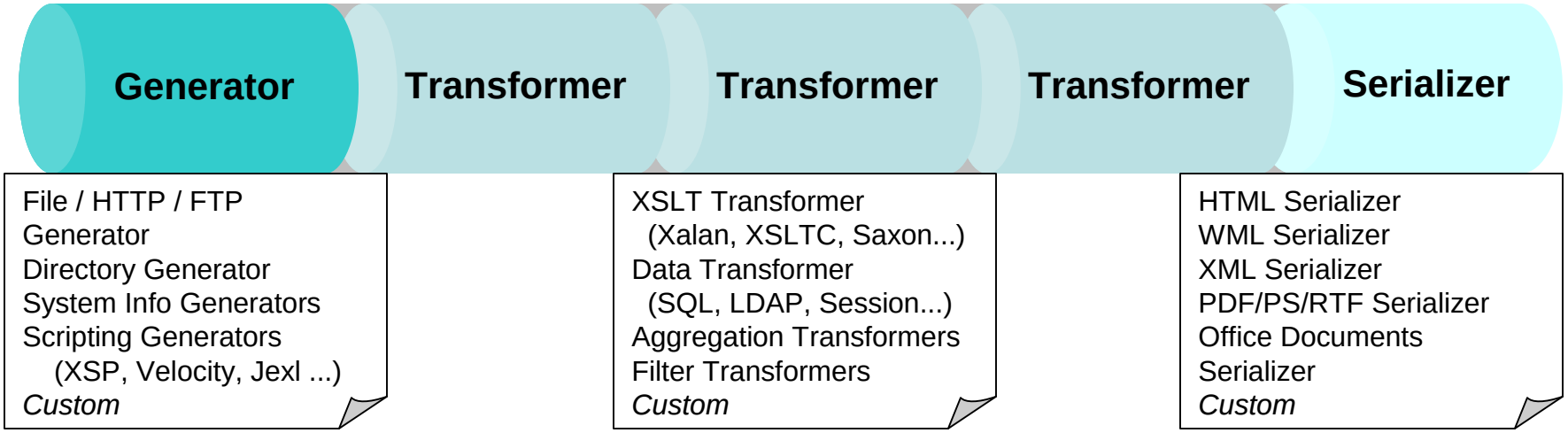
# Matching a request

- Matcher
  - Matches the incoming request to the correct pipeline
  - Different types of matchers available
  - Most common: wildcard

```
<map:match pattern="news/*" type="wildcard">
    <map:generate src="newsfeeds/{1}.xml" type="file"/>
    <map:transform src="news2html.xsl" type="xslt"/>
    <map:serialize type="html"/>
</map:match>
<map:match pattern="products/*" type="wildcard">
    <map:generate src="products/infos/product_{1}.xml" type="file"/>
    <map:transform src="products2html.xsl" type="xslt"/>
    <map:serialize type="html"/>
</map:match>
```

# Cocoon Basics

- Using Cocoon
  - Program the Sitemap
  - Define the matches and pipelines
  - Separating Content from Layout

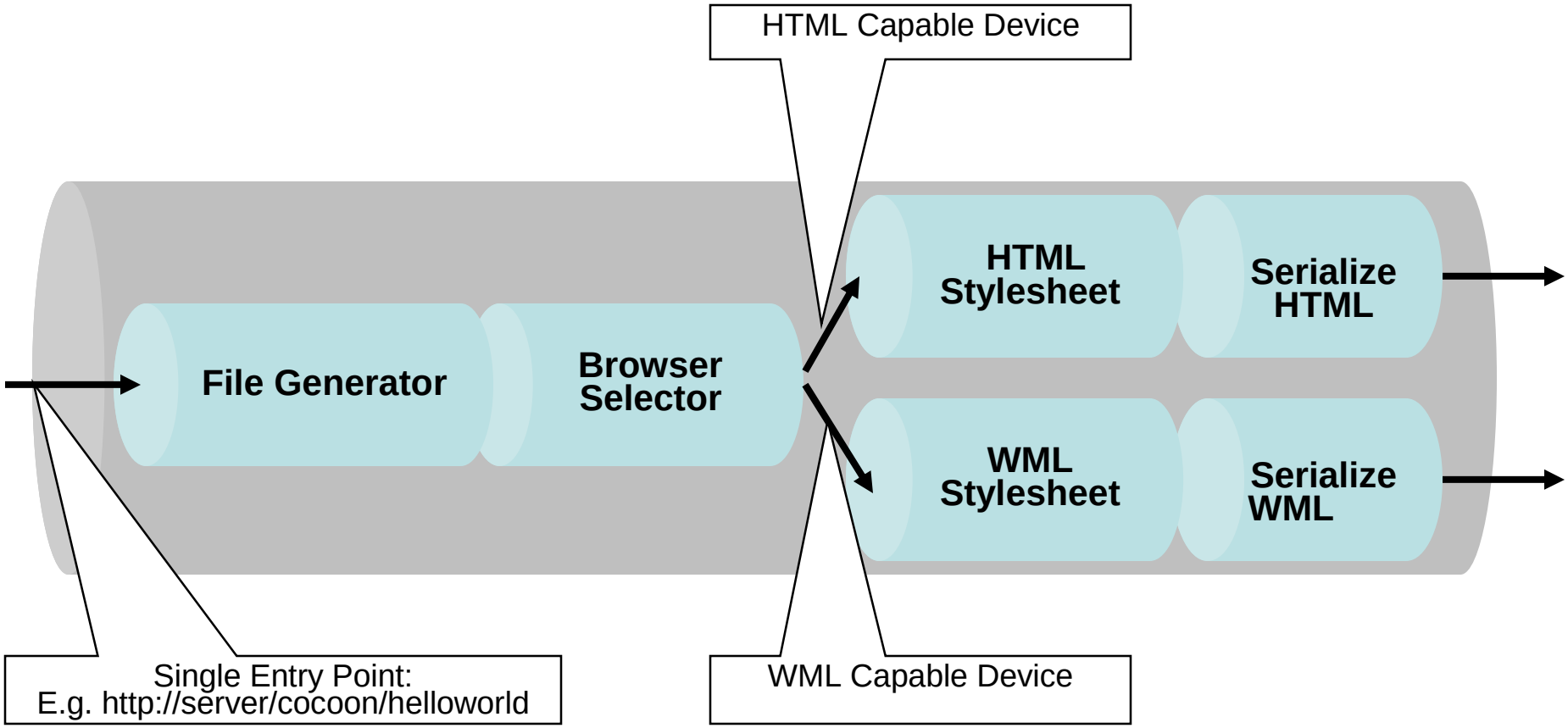


# Beyond The Basics

# Additional components

- Selectors
  - Are used to determine the "flow" through a pipeline
  - They are like "if-else-if" statements in programming languages
  - Example: Browser Selector
    - Can detect used browser
    - Select stylesheets dependent on the browser
  - Other examples
    - IP Selector
    - Season Selector

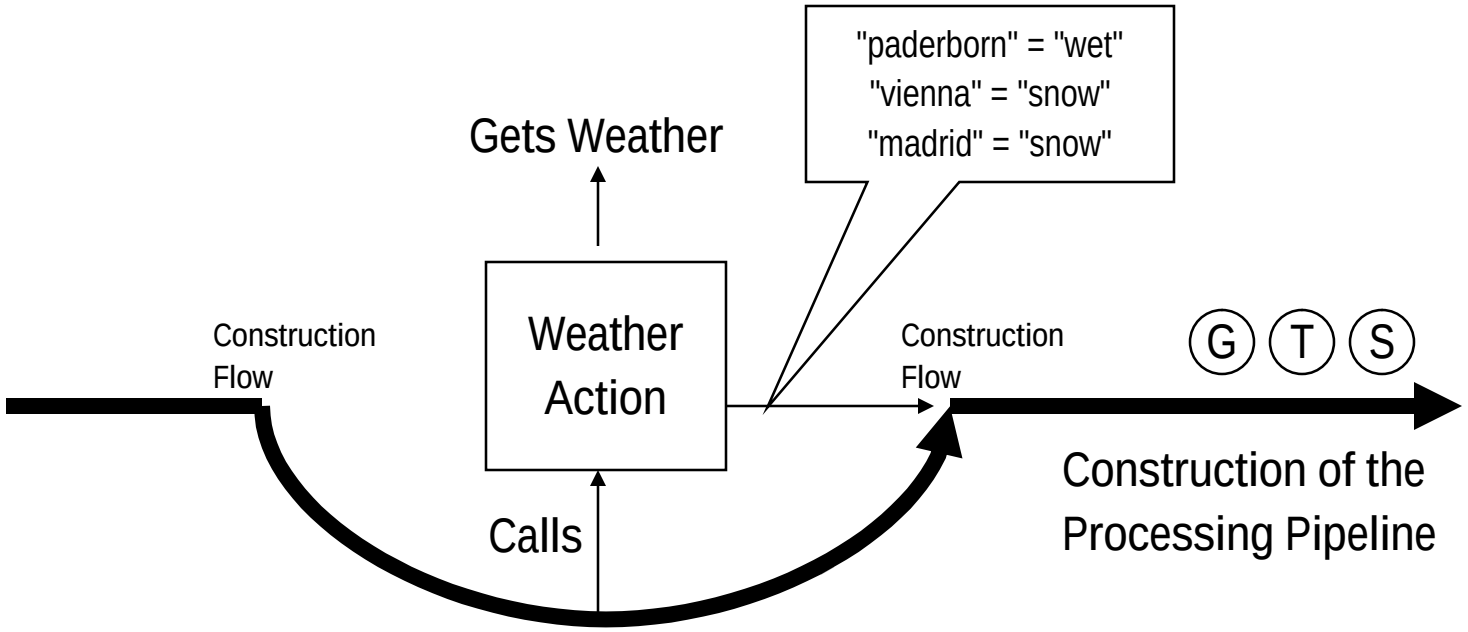
# Multi Channel Capabilities



# Additional components

- Actions
  - A component that can influence the construction of the processing pipeline!
  - Actions are instantiated and called before the pipeline processing starts!
  - Can return a Java Map of key-values
    - {userName} {Matthew}
    - {userAge} {37}
  - Other components can then act on these values
  - Examples
    - Access Request Information
    - Authentication
    - Database access (shopping carts etc.)

# Additional components




```
<map:match pattern="weather">
  <map:act type="weather">
    <map:generate src="{paderborn}.xml"/>
    ...
  </map:act>
</map:match>
```

# Additional components

- Readers
  - Component for reading non XML formats
    - Images, Films etc.
  - Use when the "document" is already in the needed format
  - Can be thought of as a combination of Generator+Serializer
  - Can be configured with the MIME-Type

And there is much more!

- Input Modules
- XSP
- Content Aggregation
- Redirects
- Resources
- Views
- …

# Moving Beyond "just" Publishing

# An Overview

- What else does Cocoon include?
  - Session Handling
  - Portal Framework
  - Authentication Framework
  - Forms Handling
  - Flow

# Session Framework

- Manage User sessions with Cookies or URL Rewriting
- Include Session Action in pipeline for session creation
- Use Transformer for data handling
  - Create application contexts (buckets `J`) in the session
  - Store/Retrieve data as XML
  - Access to Request context

# Portal Framework

- Coming later today **J**

# Authentication Framework

- Authentication of Users
  - Default: Using an XML file
  - But it's just a pipeline **J**
- Flexible Integration of (existing) User Databases
- Uses Session Framework
- Protection of Documents (Pipelines)
  - Documents can be grouped using Handlers
- Everything Configurable in the Sitemap
  - Flexibility through the use of Pipelines
  - Mapping: Function **è** Pipeline Call

# Forms Handling

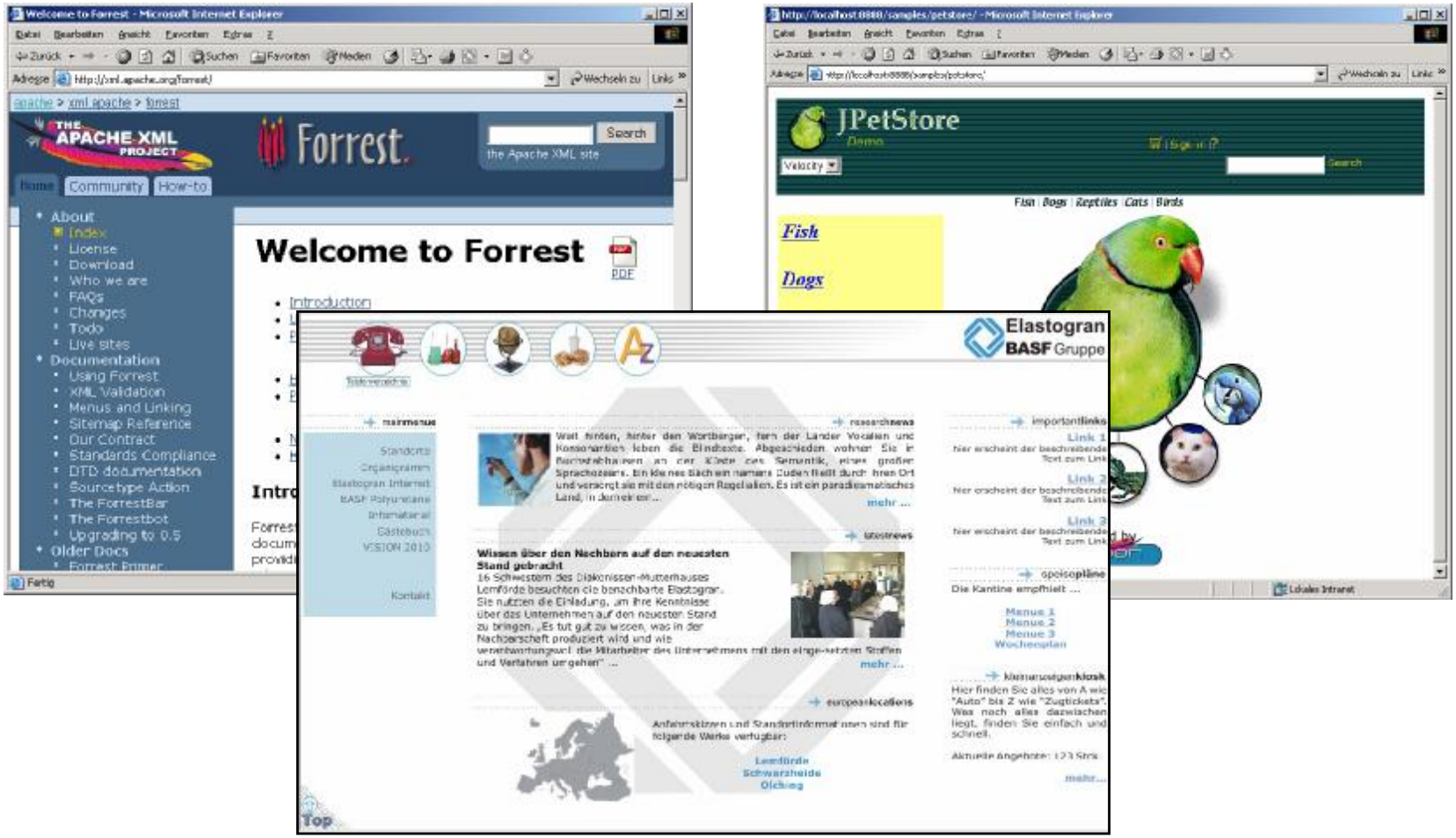
- Cocoon Forms (aka "Woody")
- Good for complex / multi forms
- Widget Oriented Approach
  - Custom validation rules
  - Dependencies
  - Strongly type data
- Separation of declarations and templates
  - Modular
  - Reuse
- Business Integration via Flow
- I18n Support

# Flow

- Control application flow
  - Examples: Registration to a newsletter, Checkout process in a shop
- Using regular JavaScript, Continuations and Pipelines
- How does it work
  - Request arrives in sitemap
  - Main Javascript function is called
  - For each "step" in the script a page (view) is returned to the user and the state saved on the server
  - The returned page can contain data from the script and *must* contain a continuation id
  - For each subsequent step, the continuation is sent to the sitemap and the script activated in the right place

# Introducing Apache Cocoon

## Simple Concept – Huge Impact



# What's next for Cocoon?

- Big move to "Blocks"
  - Making Cocoon easier to install and manage
  - Easier Deployment and Change Management
  - Functionality can be packed into separate blocks
  - Blocks can be local or installed from remote
  - Mix and Match only the blocks you really need
- Coming to a computer near you in 2004

# Summary

- Learning curve can be steep at the beginning
  - New technologies: XML, XSL, SAX
  - New architecture: Sitemap, Pipelines
  - Lots of "features"
    - What do I really need?
  - "Could be better" documentation
    - Books are available
  - Tools are just now becoming available

# Benefits

- No real alternative
  - That offers everything available in Cocoon
- XML driven architecture
  - Extensible with own components
- Flexible data integration and publishing
  - Often: No programming needed
- Large code-base
  - Many components provided
  - Most of the hard work is done already

# Benefits

- Stable platform (3 years+)
- Java + Servlet makes WORA possible (well nearly...)
- Large community
  - Including large corporations
  - "Awareness" in the public is growing fast
- Separation of Concerns
  - Team Development

# Further Information


- Apache Cocoon Project
  - http://cocoon.apache.org
  - Downloads of source (and binaries)
  - CVS Access
  - Links to additional sites, information etc.
  - In addition - Mailing-Lists
- Cocoon Documentation Wiki
  - wiki.cocoondev.org/Wiki.jsp
- Apache Forrest
  - Cocoon based documentation framework
- Books
  - Currently several published
- Competence Center Open Source
  - http://s-und-n.de