

Cocoon – The Next Generation

What's new in Cocoon 2.2

Carsten Ziegeler

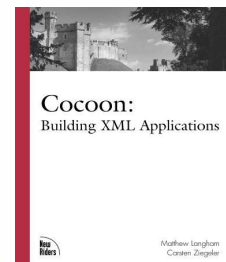
cziegeler@s-und-n.de

Competence Center Open Source
S&N AG, Germany



About

- Member of the Apache Software Foundation
- Apache Cocoon Committer since 2000
- Chief Architect of the Competence Center Open Source, S&N AG
- Article and Book Author
- Technical Reviewer



Agenda

cocoon

- Current State of Cocoon
- Why 2.2?
- The Vision
- What will be new?
- The current State
- Discussion

JAX 2005

Present

What is Apache Cocoon? ☺

Apache Cocoon is a web development framework built around the concepts of separation of concerns and component-based web development.

Cocoon implements these concepts around the notion of 'component pipelines', each component on the pipeline specializing on a particular operation. This makes it possible to use a Lego(tm)-like approach in building web solutions, hooking together components into pipelines without any required programming.

Cocoon is "web glue for your web application development needs". It is a glue that keeps concerns separate and allows parallel evolution of all aspects of a web application, improving development pace and reducing the chance of conflicts.

What is Cocoon?

- Cocoon is a powerful web application framework
 - It is **not** just an XML web publishing platform
- Serious web applications can still be fun
- Write code only when you need to
- The magical trio: pipelines, flow and forms
- Cocoon is aimed for larger projects/teams!

Apache Cocoon

- Top-Level Apache Open Source project
 - <http://cocoon.apache.org>
- Started in 1999
 - Original goal: XML Publishing Framework
- Today
 - A thriving healthy community
 - One of the most important Apache projects
 - Incorporates technologies from various project
 - Used/Supported by several major companies

Introducing Cocoon

- XML (Publishing) Platform
 - Framework integrated into a Servlet
 - Usable in other environments like command line
- Aim: Separation of Concerns (SoC)
 - Site Administrator, Content Deployer, Layout Deployer
 - Made for larger projects and teams
- Focus on *Composing* rather than *Programming*
 - “We figure out the hard parts, you get to do the fun stuff.”

Extensible Architecture

- Component Orientated
 - Based on Apache Avalon Framework
- Already integrates other projects
 - Xalan, Xerces
 - FOP, Batik, POI
- Add new/own components (if required)
 - Seamless Integration
- No lock-in – Use what you need!

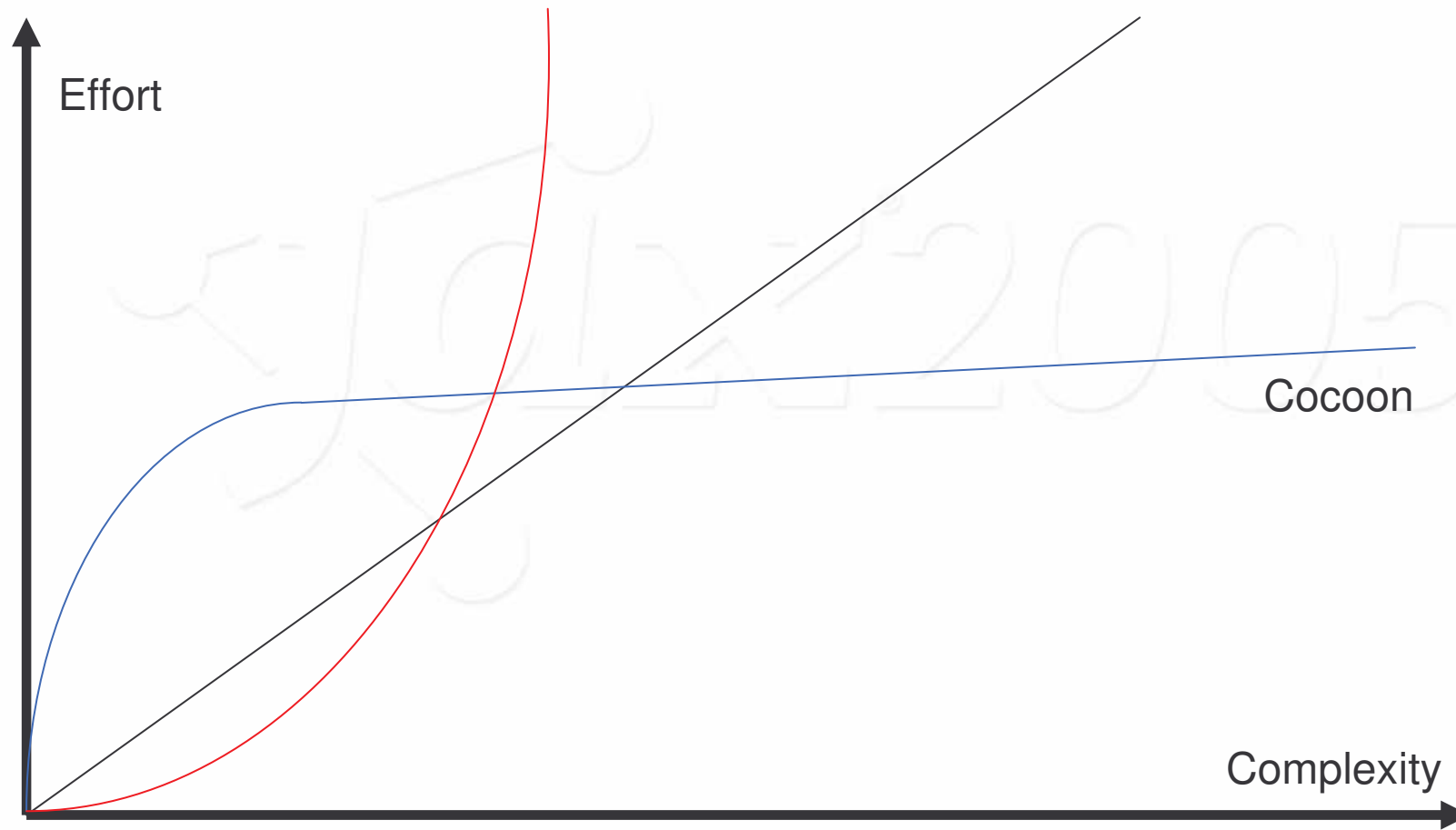
Scenarios

- Dynamic Document Generation
 - Based on XML/XSLT
 - But not limited to
- Used for various application scenarios
 - Web Sites
 - Web Publishing
 - XML Portals
 - XML Processing Systems
 - ...

Motivation for 2.2

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

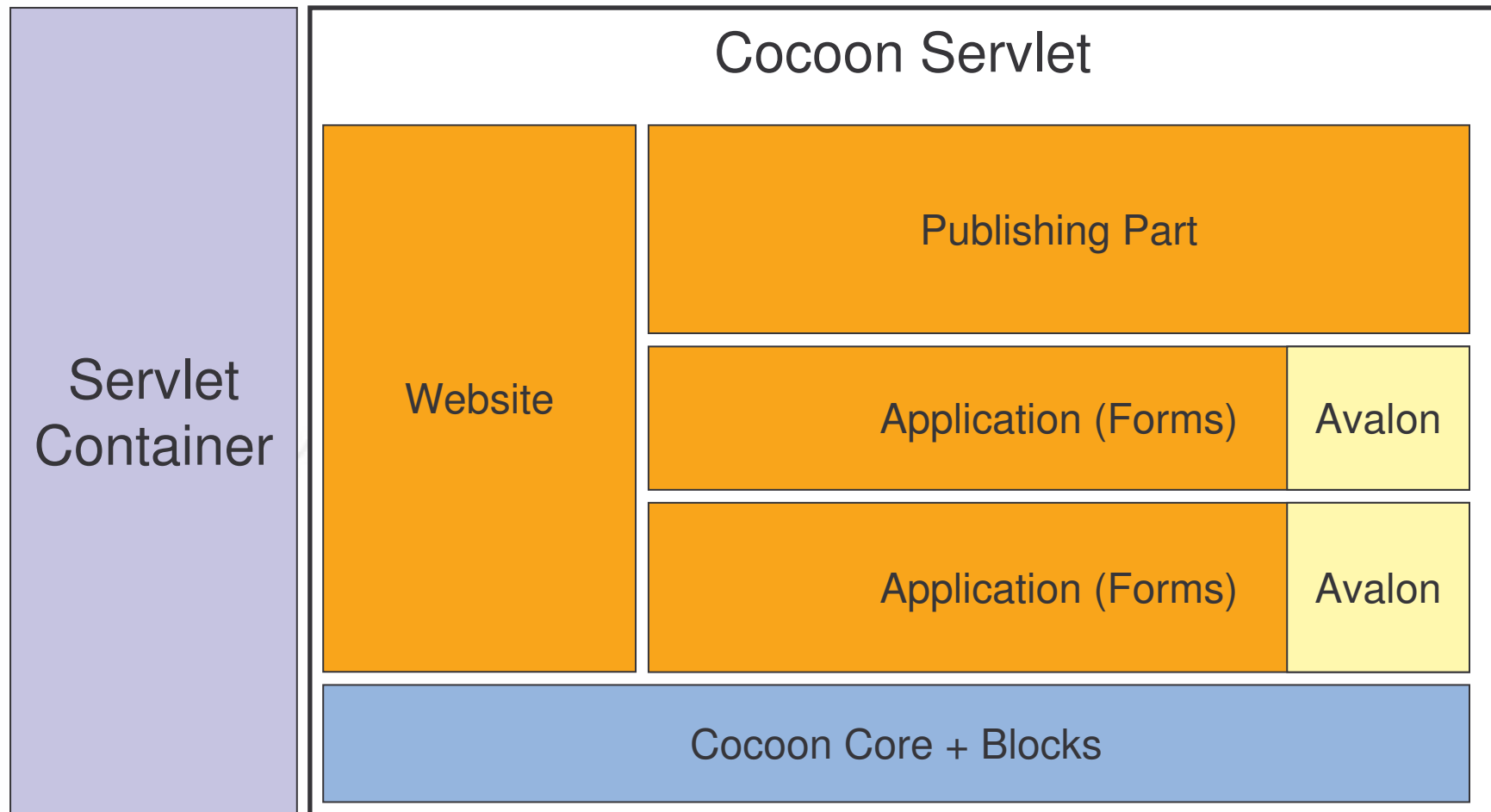
Learning Curves



Ingredients

- Cocoon Core
 - Component Container (Avalon based)
 - Basic Functionality (Sitemap, Flow)
- Additional Functionality through blocks
 - Currently over 50 blocks
 - Ranging from simple to complex
 - For Example: FOP, Forms, Cron, Portal etc.

A Typical Cocoon Installation



Functionality Configuration (2.1)

- Block Selection before building Cocoon!
 - Check Dependencies
- Other configuration possibilities for building
 - Samples, Documentation
 - Upload enabling etc.

Changes in Configuration require a rebuild!

Cocoon Configuration (2.1)

- **All** components are configured in one central place: cocoon.xconf (> 120 kb)
- Additional (global) configuration
 - web.xml
 - Logging configuration (logkit.xconf)
- Different ways to add own stuff!

Changes in Configuration require a rebuild!

But changes will be overwritten during rebuild!



Cocoon 2.2

„Nothing is carved in stone!“

Goals for 2.2

- Flatten the learning curve
- Consolidation
- Support rapid prototyping/development
- Make building and configuring easier
- Better documentation
 - Make writing documentation easier
- Enabler for Real Blocks™
- BUT: Be as compatible as possible!

The Vision: Real Blocks™

- **Blocks** are reusable functional modules at the cocoon web application level.
- **Blocks** were introduced in Cocoon to allow users to:
 - easily deploy their content on Cocoon without requiring operation downtime
 - package functionality /services in modules that can be reused as-is
 - easily extend existing modules
 - create complex web applications by high-level composing of these functional modules
 - depend on module behaviors, allowing for polymorphic module composition

The Vision: Real Blocks™

- Hot-Deployment
- Resolving of dependencies
 - Maven-style repository downloads
 - Versioning
 - Classloading
- Reconfiguration
- Make blocks independent (own versioning)

The Gordian Knot

- Plans for 2.2 are **very** old (nearly 3 years)
- Last year: „quiet period“ for development
 - Rest on our laurels
 - Hope that Avalon improves/helps
- Other component frameworks evolved
 - Especially Hivemind and Spring
- Rapid application frameworks
 - Ruby on Rails
- Noone dared to tackle the issues
 - Way too complex!

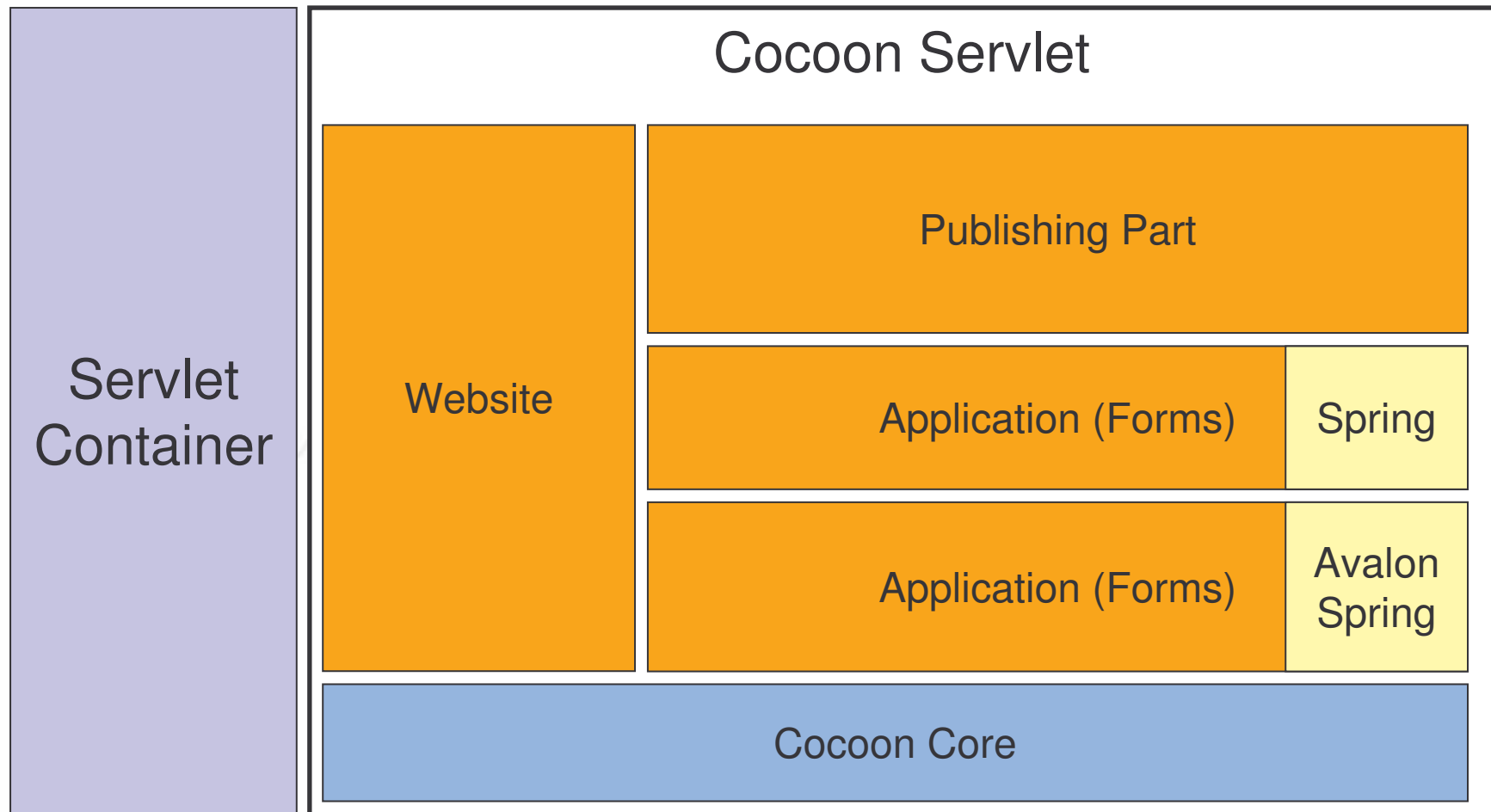
The Gordian Knot has been cut!

- The Legend Of Avalon...has ended
 - Avalon is closed but maintained at Excalibur ☺
- Avalon was a core-dependency for Cocoon
 - Cocoon not runnable without it!
 - No evolution possible in Cocoon without Avalon
- Cocoon wanted to
 - have an independent core but
 - reuse efforts of other projects
 - without reinventing the wheel

The New Core

- Own Component Container
 - Provides Avalon Compatibility
 - Adds own functionality
- **Slowly** Turning away from Avalon Interfaces
 - Only for Cocoon itself!
- Separation between core and application!
 - Core uses own container
 - Applications are encouraged to use what they need! (Avalon, Spring etc.)

A Typical Cocoon Installation



Configuration

- Central Configuration (cocoon.xconf) is split
 - Includes per block configurations
 - Dependencies for configurations are checked
 - Removing/adding a block by removing/adding the appropriate configuration
 - Jars are not covered
 - No tool **yet**
 - Own configuration can be separated into own files
- The same applies to logkit configuration

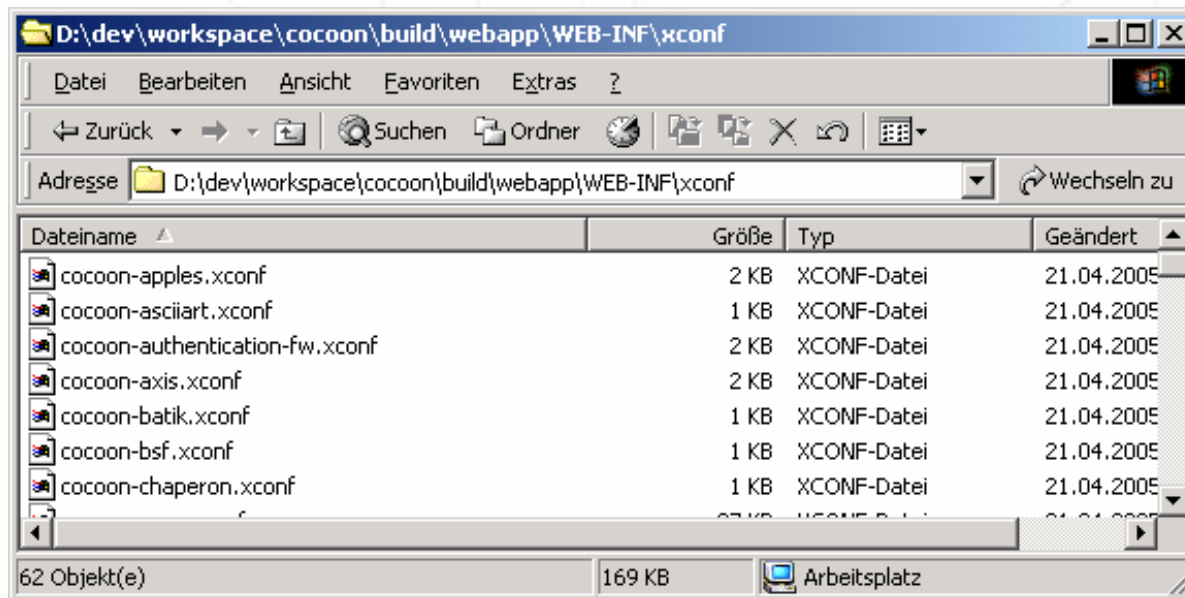
Configuration - Sample

```
<cocoon version="2.2">
```

```
  <include src="context://WEB-INF/cocoon-core.xconf"/>
```

```
  <include dir="context://WEB-INF/xconf" pattern="*.xconf"/>
```

```
</cocoon>
```

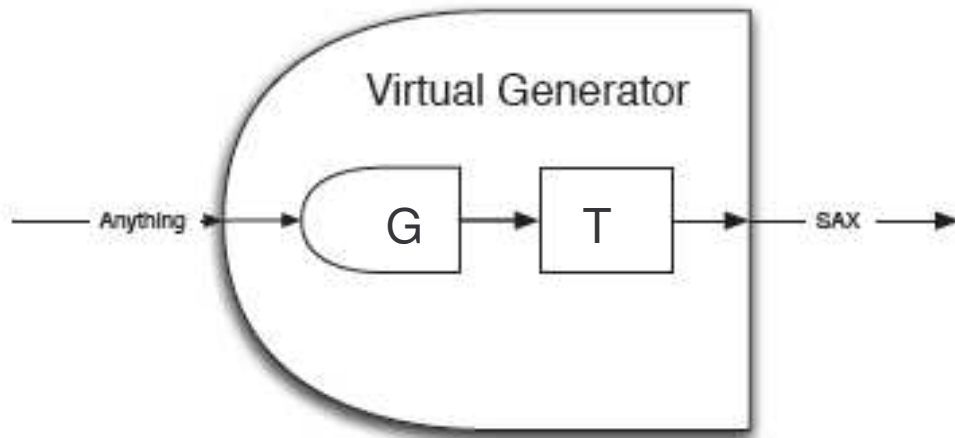


Configuration - Properties

- Dynamic properties can be used
 - In all component configurations (cocoon.xconf etc)
 - In all logkit/log4j configurations
- Properties are searched in:
 - System Properties (defined on startup)
 - Local Property File (defined by system property)
 - Default in webapp: cocoon-settings.properties
 - <dburl>\${myapplication.dburl}</dburl>
 - <user>\${myapplication.dbuser}</user>
 - <password>\${myapplication.dbpasswd}</password>

Virtual Sitemap Components

- Pipeline Components that emulate pipeline fragments
 - Possible today with resources but neither elegant nor block-friendly



Central Controller: Sitemap

- A Sitemap mimics a Cocoon Application
 - Each application inside Cocoon has own sitemap
- A Sitemap can contain:
 - Own Configuration
 - Own Classloader
 - Application Container
 - Listeners

Per Sitemap Configuration

- Per Sitemap Components Possible
 - Components are only available to sitemap and their sub sitemaps
 - Configure components where they are used
 - Especially interesting with classloading...

```
<map:components>  
  <map:include src="conf/myown-config.xconf"/>  
  <map:include dir="conf" pattern="*.xconf"/>
```

Sitemap Classloading

- Per Sitemap classloading possible
 - Use different versions of libraries
 - Hot-Reloading when sitemap is reloaded
 - (Tracking will be available soon)
 - Faster development
 - Just drop your webapp including Java code as a sub sitemap into Cocoon

```
<map:components>  
  <map:classpath>  
    <class-dir src="local-classes"/>  
    <lib-dir src="local-libs"/>  
  </map:classpath>
```

Per Sitemap Container

- Use your own container for your application
 - Seamless integration
 - Cocoon ServiceManager looks up components
 - Using the „Avalon“ hierarchy
 - But queries the sitemap container first!
 - E.g. Flow: `getComponent("spring-test")`
 - Bidirectional support: Cocoon components are available in app container as well
 - E.g. as Spring beans
- Currently just a prototype!

Per Sitemap Container

- Have a look at the Cocoon spring-app block
 - Own Application Context per Sitemap
 - Create Hierarchy of contexts
 - Separate Configurations

```
<map:components>  
  <map:application-container class="o.a.c.spring.SpringComponentLocator">  
    ...Configuration...  
  </map:application-container>
```

Sitemap Listeners

- Configure per sitemap listeners
 - Invoked when a request enters a sitemap
 - Invoked when a request leaves the sitemap
- Events contain information about request
- Enabled initialization of request values
 - Can be done with actions or flow as well
- Enabled cleanup after request is processed!
- Currently just a prototype!

Consolidation in 2.2

- The Burden of Compatibility ☹
- Remove deprecated blocks
- Streamline parallel development
- Cocoon Forms
- Templates (JXTG)

Distribution/Release

- 2.2 will be released when it's finished ☺
 - Hopefully released in 4th quarter of 2005
 - Many features already implemented/prototyped
 - Nothing is carved in stone!
- Blocks will be made available separately?
- Binary Distribution?
- Different Flavours (Core, Demo, Full)?



The Current State



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

Summary

- Stable platform (3 years+)
- Large community
 - Including large corporations
 - "Awareness" in the public is growing fast
- Separation of Concerns
 - Team Development
- **Driven by user/developer needs**
 - Not a Research Project!

And Development Continues!

- Cocoon 2.2 matures
 - Blockathon at ApacheCon Europe (Stuttgart/July)
 - Check it out today (SVN: trunk)
 - Not recommended for projects yet!
- 2.1.7 Released, 2.1.x will be continued
- Release Early – Release Often ☺
- New features are constantly added
 - New Blocks

The Road Ahead

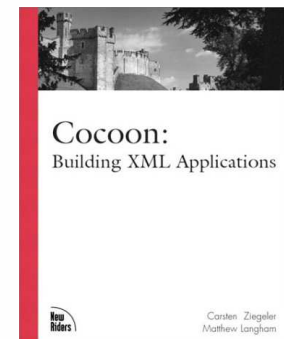
- Real Blocks™
- Switch from JavaScript to Java for Flow?
 - Don't panic – JS will be maintained for a long time
- Intercepted Flow?
- Change configuration at runtime?
- Monitoring (JMX)?

Stay Tuned ☺

Further Information

COCOOON

- Apache Cocoon Project
 - <http://cocoon.apache.org>
 - Downloads, Mailing-Lists, Links
- Cocoon Documentation Wiki
 - <http://wiki.apache.org/cocoon>
- Books
 - Currently 4,5 published
- Competence Center Open Source ☺
 - <http://www.s-und-n.de>



Thanks for your attention!

Discussion?

Questions?