Webapplications with Apache Cocoon

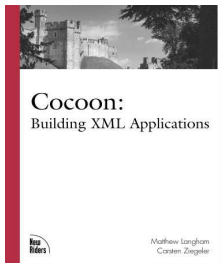
The Easy Way

Carsten Ziegeler
cziegeler@s-und-n.de

Competence Center Open Source
S&N AG, Germany

About

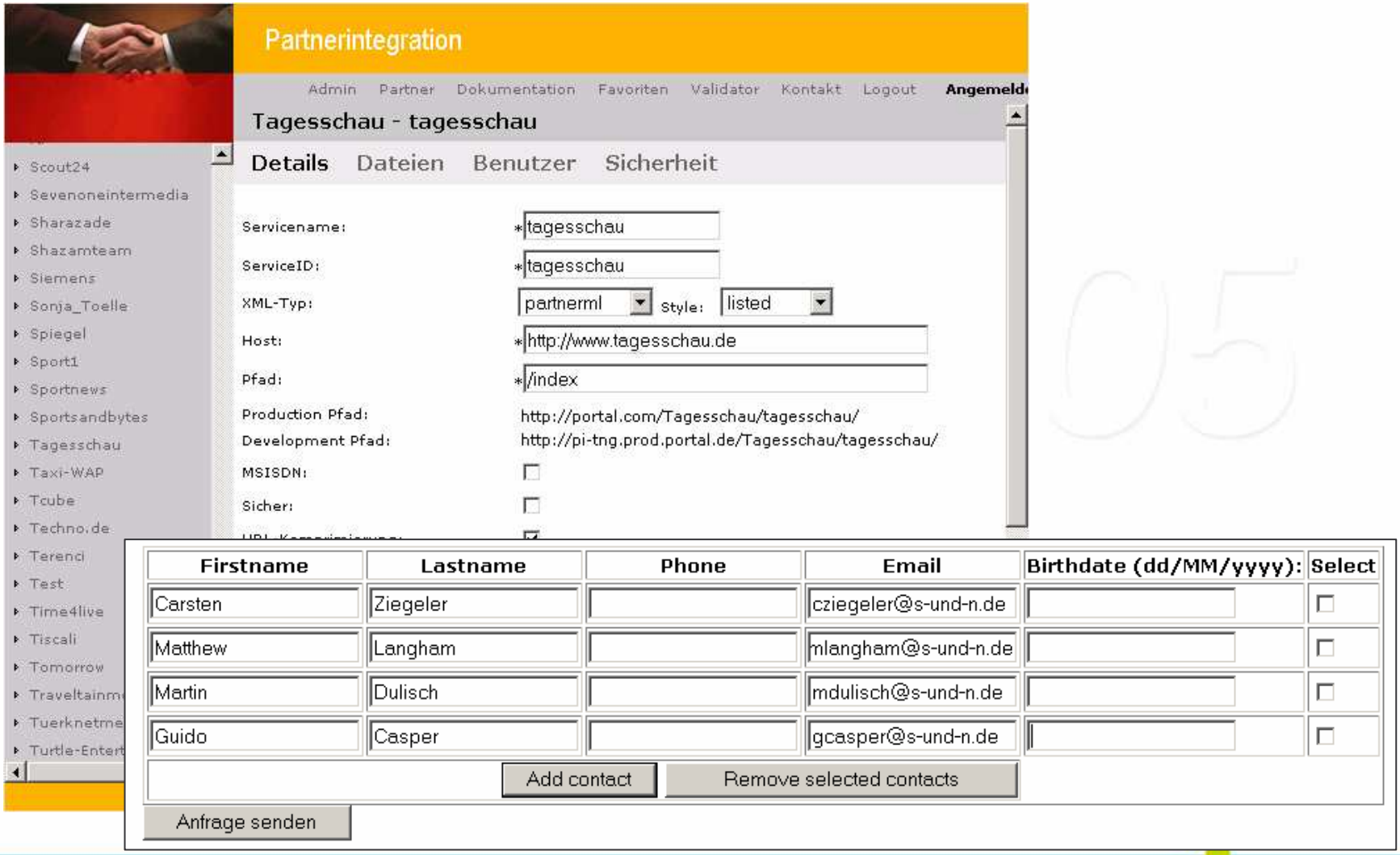
- Member of the Apache Software Foundation
- Apache Cocoon Committer since 2000
- Chief Architect of the Competence Center Open Source, S&N AG
- Article and Book Author
- Technical Reviewer



Agenda

- Cocoon in a nutshell
- Cocoon Flow
- Cocoon Forms
- Business Layer
- Q/A

Real World Applications



Introduction to Cocoon

Motivation

- 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 at 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
  - Makes extensive use of XML and XSLT
  - Usable in other environments like command line
- Aim: Separation of Concerns (SoC)
  - Site Administrator, Content Deployer, Layout Deployer
  - Made for larger projects and teams

Introducing Cocoon

- Focus on *Composing* rather than *Programming*
  - *"We figure out the hard parts, you get to do the fun stuff."*
- Core + Blocks
  - Core = pipelines, sitemap, flow, forms, i18n, jxtemplate
  - Blocks: portal, cron, fop, axis, poi, batik, databases, authentication, ojb, SAP web3, WebDAV …
  - Built-time application assembly configuration
  - (Hot-deployment and -reconfiguration is planned)

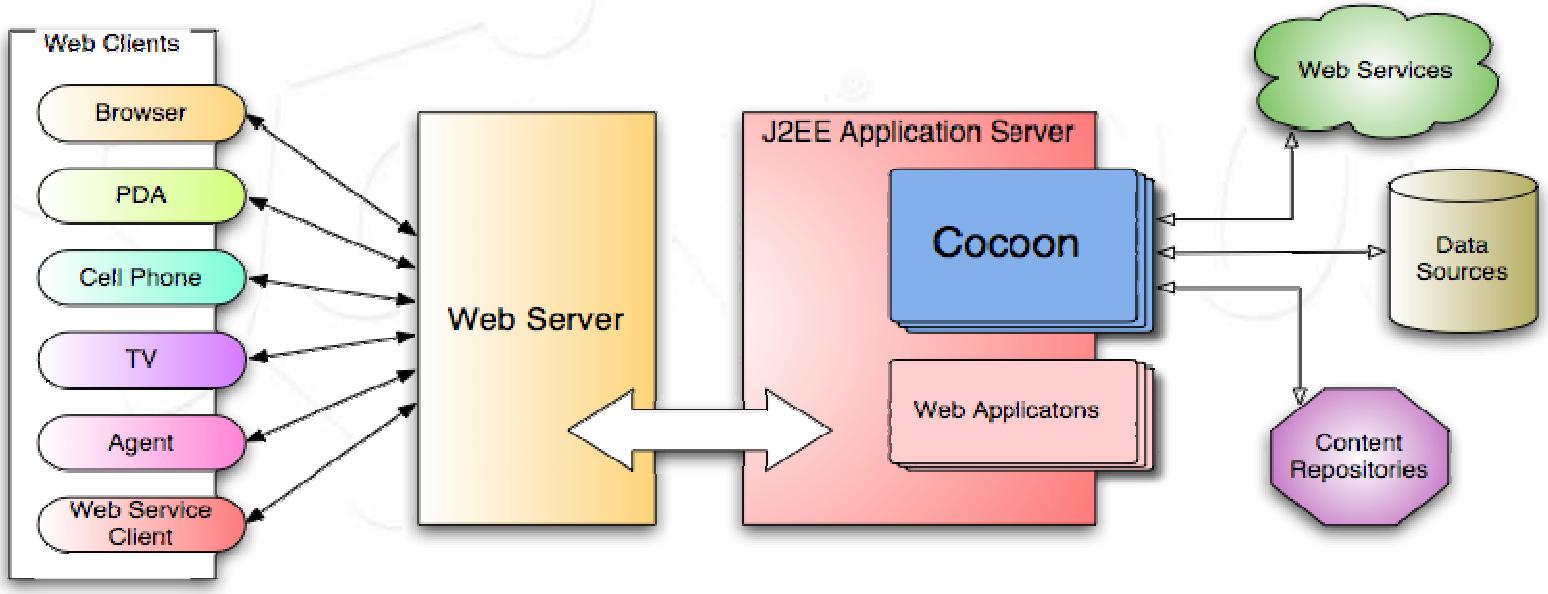
Extensible Architecture

- 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
  - …

Apache Cocoon



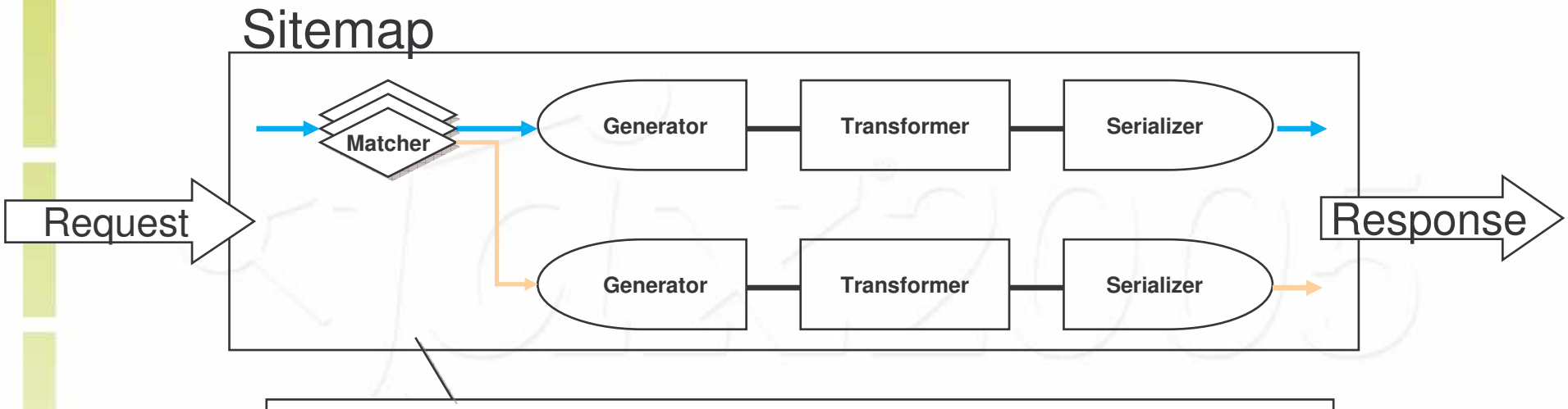
Major Advantages

- 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
  - …

Document Generation

- Dynamic Document Generation
  - Based on Request-Response-Cycle
  - Described in the **sitemap**
  - Separating Content and Layout by defining a processing pipeline per document
    - Getting Content
    - Adding Layout
    - Sending Response
  - Extensible by Logic and Flow

Request Handling (Sitemap)



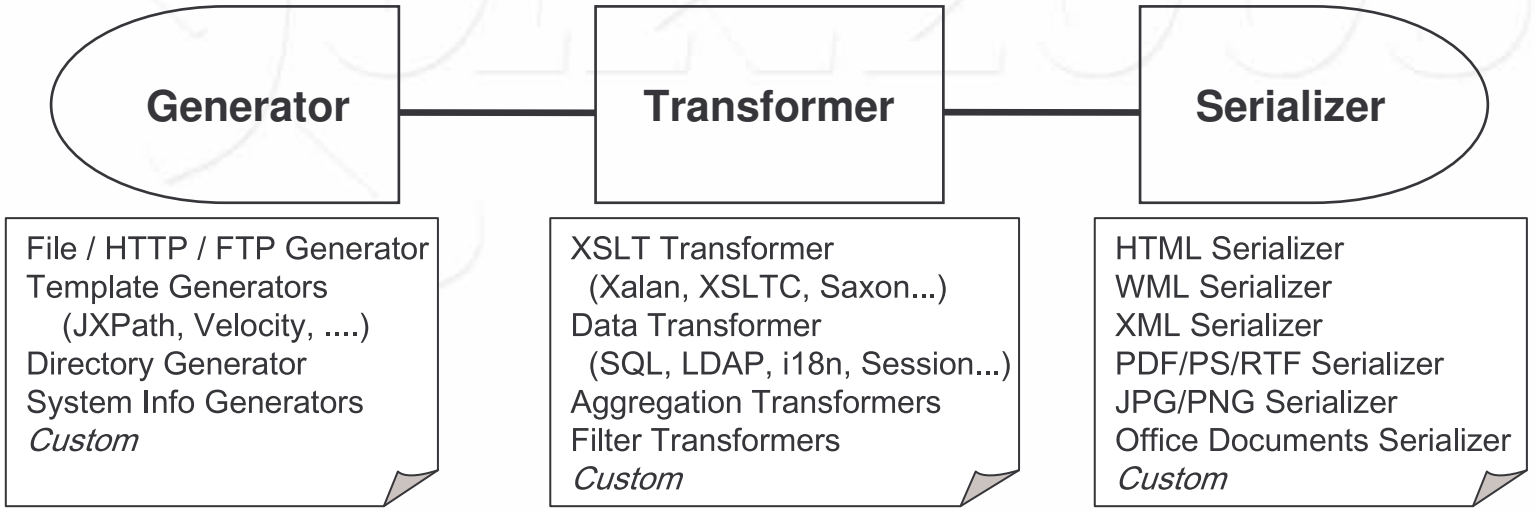

```
…
<map:match pattern="helloworld" type="wildcard">
   <map:generate  type="file" src="helloworld.xml"/>
   <map:transform type="xslt" src="helloworld2html.xsl"/>
   <map:serialize type="html"/>
</map:match>
…
```

The Cocoon Sitemap…

- describes the URI namespace of a web app (XML)
- uses Matchers to select a pipeline (depending on request/environment information)
- Pipeline consists of:
  - Generator streams XML into the pipeline
  - One or more Transformers manipulating the XML
  - Serializer convert XML into the output format
- Other component types: Readers, Actions, Views, Selectors and Error Handlers

Standard Pipeline Components

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



Non-exhaustive list

Sitemap Components

- Matchers
  - wildcard URI, regexp URI, request & session parameter
- Selectors
  - browser (UA), host, sitemap, request & session parameter
- Actions
  - authentication, database, mail, session, form
- Input Modules
  - *(accessing various input parameters in the sitemap)*

Non-exhaustive list

But there is more!

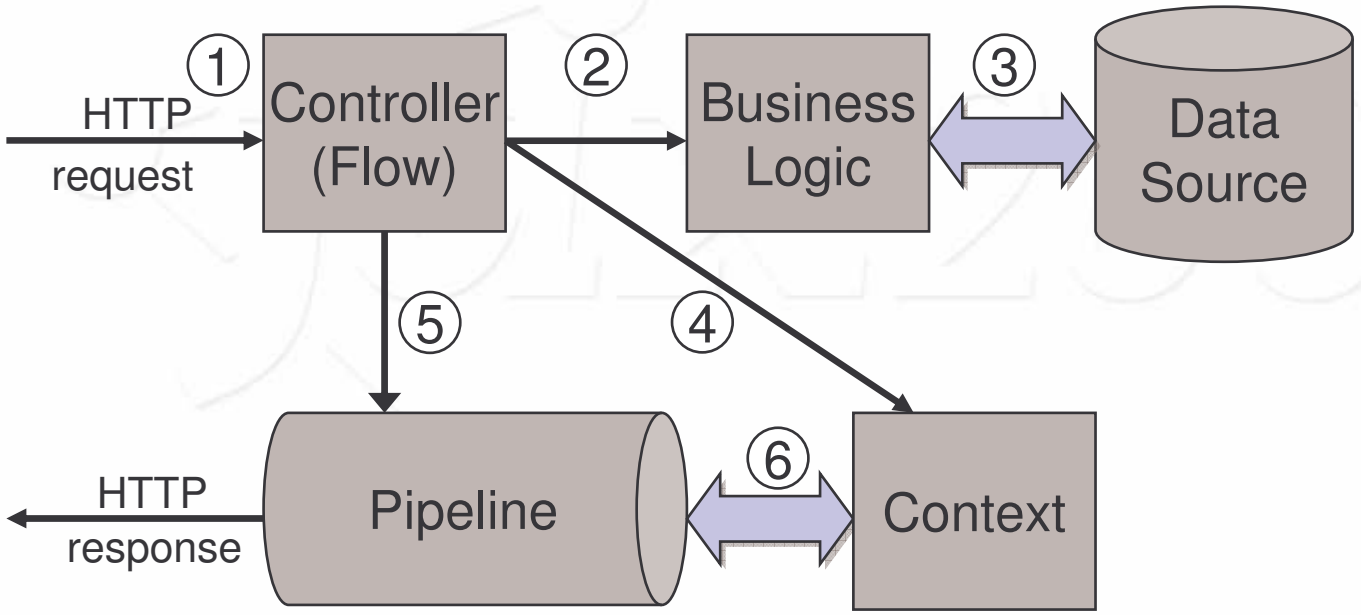
- Many ready-to-use components
- Managing the application
  - Sub sitemaps
  - Variables
- More features in the sitemap
  - Content Aggregation
  - Redirects
  - Resources and Views

Web Publishing with Cocoon

- Dynamic Document Generation
  - Separation of Content and Layout
- Powerful Processing Description (Sitemap)
- Flexible Pipeline Concept
  - Many available components
- Conditional Processing
  - Define matches and selects
- Caching

Web Applications
Cocoon Flow

Cocoon “MVC”


HTTP
request
1
Controller
(Flow)
2
Business
Logic
3
Data
Source
5
4
HTTP
response
Pipeline
6
Context

Cocoon Flow - A Control Layer

- Glues
  - business logic
  - page flow
  - presentation together
- Uses JavaScript scripts
  - server side

Flow - Example

- Newsletter Registration
  - User enters email
    - Page with form
  - If the email is not valid, user has to re-enter
    - Redisplay page with form
  - If the email is valid, show message
    - Another Page

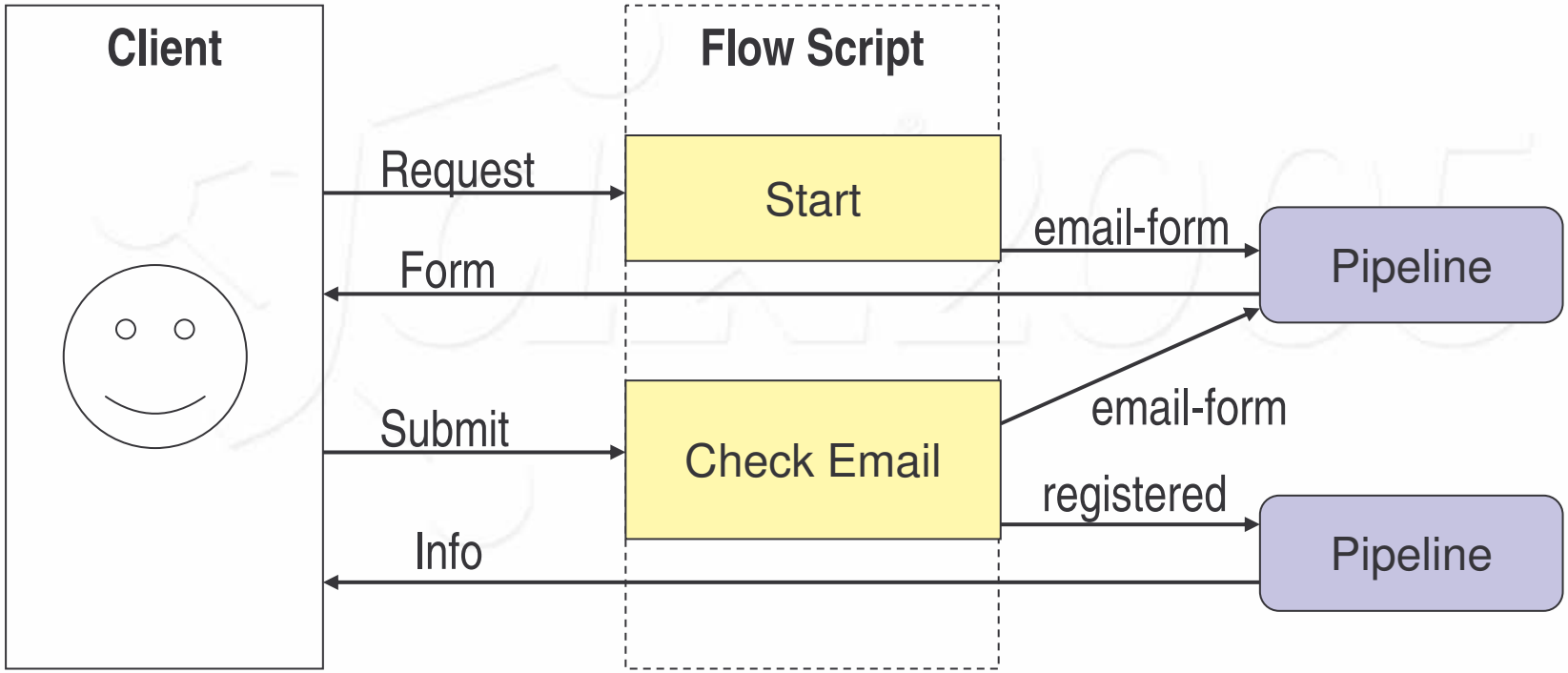
Flow - Example

```
function newsletter() {

  var email;

  while(email == null) {
    cocoon.sendPageAndWait("email-form");

    email = cocoon.request.get("email"));

    if (!checkEmail(email))
        email = null;
  }
  cocoon.sendPage("registered", {who: email});

}
```

Flow Script

- Regular JavaScript
- Access to “cocoon” object (and others)
  - Access to environment
    - E.g. cocoon.request.getParameter("name");
  - Continuation functions
- Intercepted by response request cycle
  - Continued using the continuation identifier

Flow - Example



Flow - Example

```
function newsletter() {

  var email;

  while(email == null) {
    cocoon.sendPageAndWait("email-form");

    email = cocoon.request.get("email"));

    if (!checkEmail(email))
        email = null;
  }
  cocoon.sendPage("registered", {who: email});

}
```

Flow – Continuations

- A Continuation contains
  - Stack of function calls
  - Value of local variables
  - Unique identifier
    - Passed to the view
- URL Encoding with continuation identifier
- Creating a continuation does not halt a thread

Flow Script – Calling the View

- cocoon.sendPage()
  - Invokes the output page (view)
    - The view pipeline
    - A context for the view

```
cocoon.sendPage("registered",
        {"who": name, "mail": email});
```

  - Is like a forwarding to the pipeline
  - Control then comes back to the script

    → Should normally terminate

Flow Script – Calling the View

- cocoon.sendPageAndWait()
  - Similar to cocoon.sendPage, but
  - Script is suspended after the view is generated
    - Continuation is created
    - Response is sent to the client
  - Next request continues the script
    - Using continuation id

How does it all work?

- First request arrives at the sitemap
- Main JavaScript function is called
- For each "step" in the script
  - the state is saved on the server
  - a page (view) is returned to the user
    - can contain data from the script
    - must contain continuation id
- Each subsequent request (with cont. id)
  - Arrives at the sitemap
  - Activates the script in the right place

The View Layer - Pipeline

- JXTemplateGenerator
- JXTemplateTransformer
- Other alternatives available
  - Velocity
  - In the scratchpad part of Cocoon
- Custom components
- Cocoon 2.2: Template Block

The JXTemplateGenerator

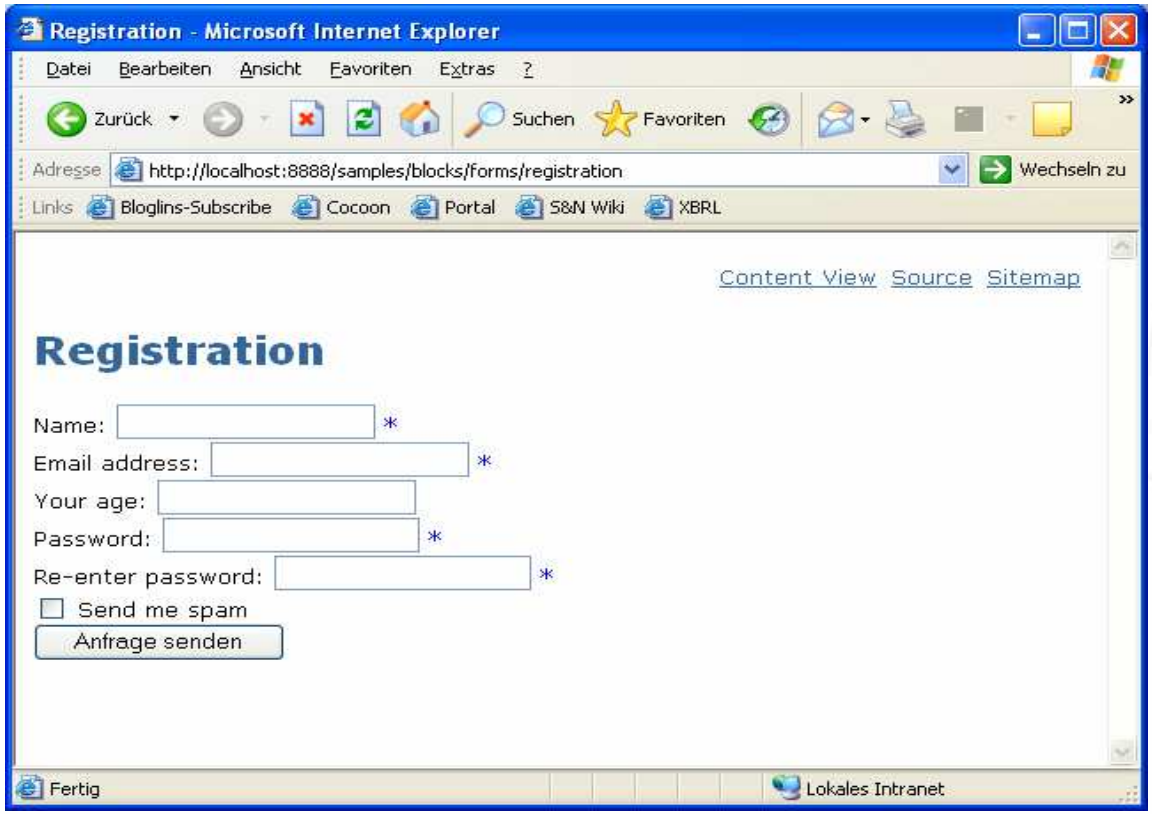
- Access to flow context
  - XPath Expression: #{user/address/city}
  - EL (JSTL) Expression: ${3 + 4}
- Display Logic
  - Loops
  - Conditions
  - Number Formatting

Flow In a Nutshell

- Cocoon Flow is a big improvement
  - Powerful concept for building applications
  - Page flow
- Different options for building the View
  - JXTemplateGenerator
- Look at the PetStore example in Cocoon
- Documentation is not yet that good
  - Check the Cocoon Wiki

Web Applications
Cocoon Forms

Simple Form Application



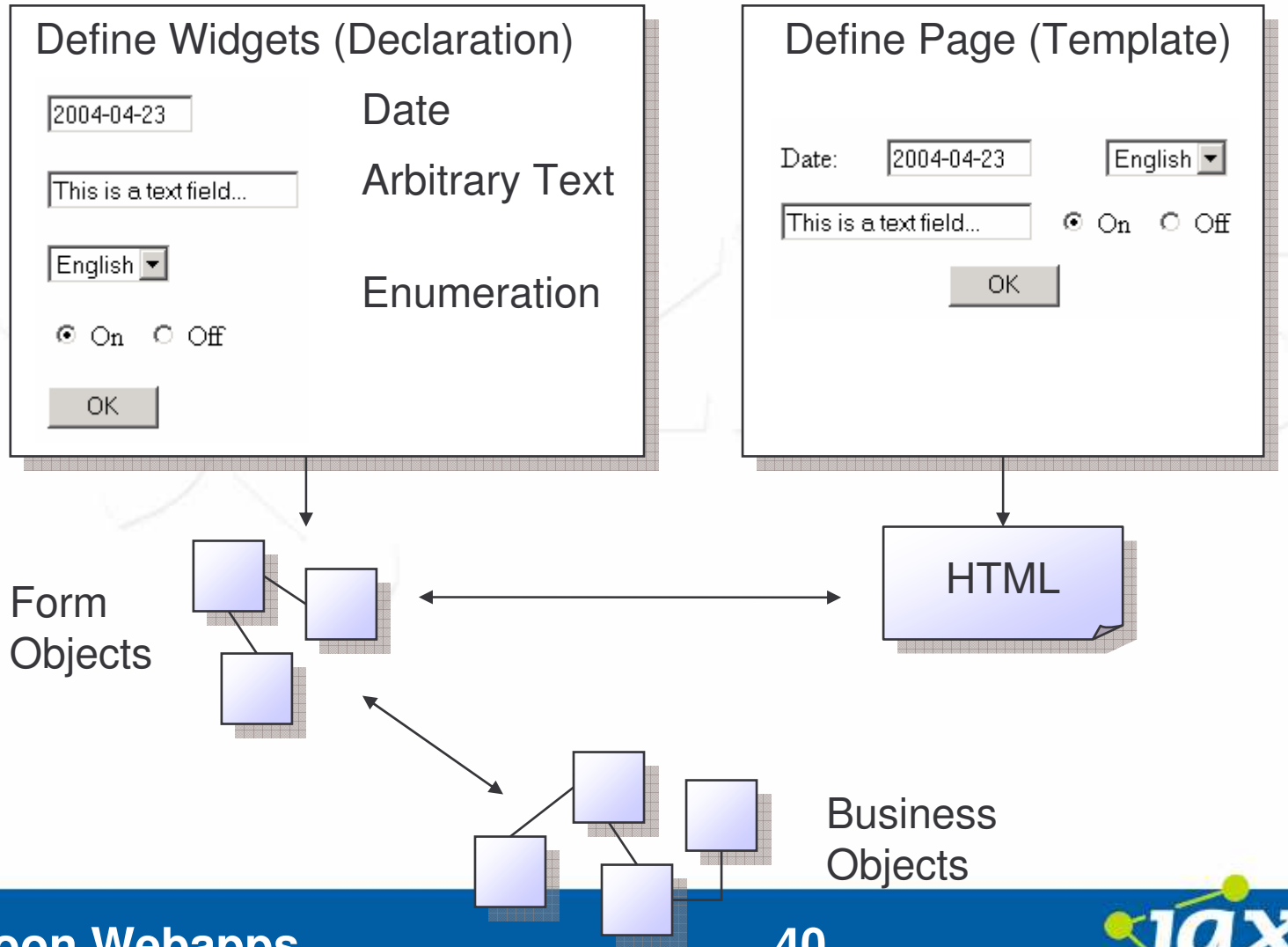
Cocoon Forms

- Official Cocoon Form framework “CForms”
- Fundamentals
  - Very little Java coding required
  - Strong datatyping & type conversion capabilities
  - Offers a growing set of ready-made widgets
  - Integrates with various development styles (flow)
  - Server-side form model plus event model
  - Separation of Concerns
  - Optional Binding (XML, Beans)

Widget Oriented Approach

- Set of standard widgets. For example:
  - Field (input, listbox, radiobuttons, textarea)
  - Booleanfield (checkbox)
  - Multivalue field (listbox, checkboxes)
  - Aggregated field
  - Action (button)
  - Upload
  - Repeater
    - Nested elements
    - Produces tabular forms with functions for adding/deleting rows

How it works


Define Widgets (Declaration)
2004-04-23
Date
This is a text field...
Arbitrary Text
English
Enumeration
On
Off
OK
Define Page (Template)
Date:
2004-04-23
English
This is a text field...
On
Off
OK
Form Objects
HTML
Business Objects

Step 1: Widget Definitions

```
<fd:field id="name" required="true">
  <fd:label>Name:</fd:label>
  <fd:datatype base="string">
    <fd:validation>
      <fd:length min="2"/>
    </fd:validation>
  </fd:datatype>
</fd:field>

<fd:field id="email" required="true">
  <fd:label>Email address:</fd:label>
  <fd:datatype base="string">
    <fd:validation>
      <fd:email/>
    </fd:validation>
  </fd:datatype>
</fd:field>

<fd:field id="age">
  <fd:label>Your age:</fd:label>
  <fd:datatype base="long">
    <fd:validation>
      <fd:range min="0" max="150"/>
    </fd:validation>
  </fd:datatype>
</fd:field>
```

Unique ID (used in template)

Datatype

Validation rule

Step 1: Datatypes / Validation

- Datatypes
  - String, Boolean, Integer, Long, Enum, Float, Date
  - Validation
    - Length, Range, Assert, RegExp, Email, Mod10
    - Combined validation rules
    - Validation may depend on other widgets‘ values
    - Custom failure messages
    - JavaScript (per widget and form)

Step 2: Widget Template

```
<page xmlns:ft="http://apache.org/cocoon/forms/1.0#template"
      xmlns:fi="http://apache.org/cocoon/forms/1.0#instance">
 <title>Registration</title>
 <content>
   <ft:form-template action="registration" method="POST">
     <ft:continuation-id/>
     <fi:group><fi:styling layout="columns"/>
       <fi:items>
         <ft:widget id="name"/>
         <ft:widget id="email"/>
         <ft:widget id="age"/>
         <ft:widget id="password"/>
         <ft:widget id="confirmPassword"/>
       </fi:items>
     </fi:group>
     <ft:widget id="spam"/>
     <ft:widget-label id="spam"/>
     <ft:widget id="submit"/>
   </ft:form-template>
 </content>
</page>
```

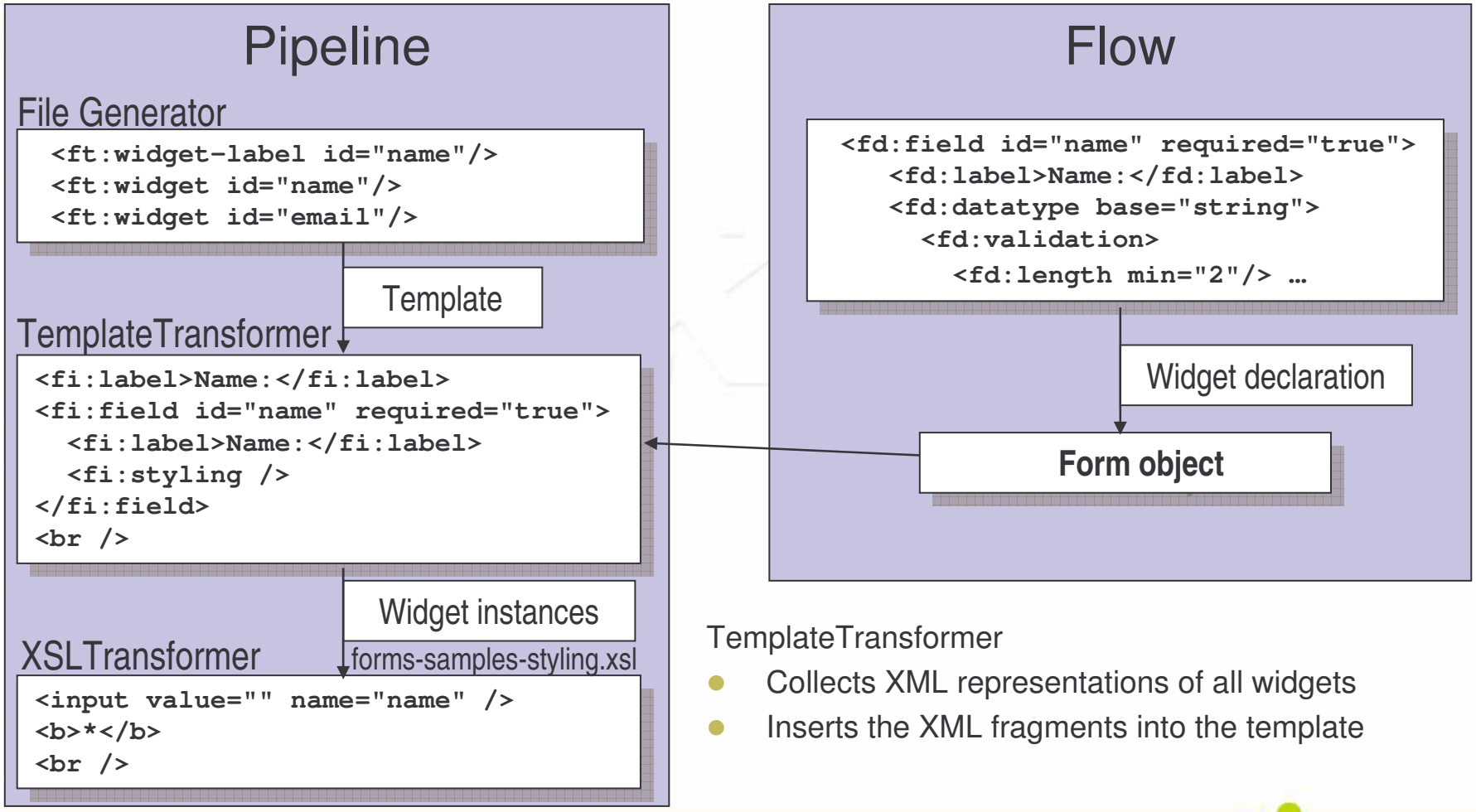
Call flow

Layout

Unique ID

Widget and label can be positioned independently

Step 3: Template Pipeline



TemplateTransformer

- Collects XML representations of all widgets
- Inserts the XML fragments into the template

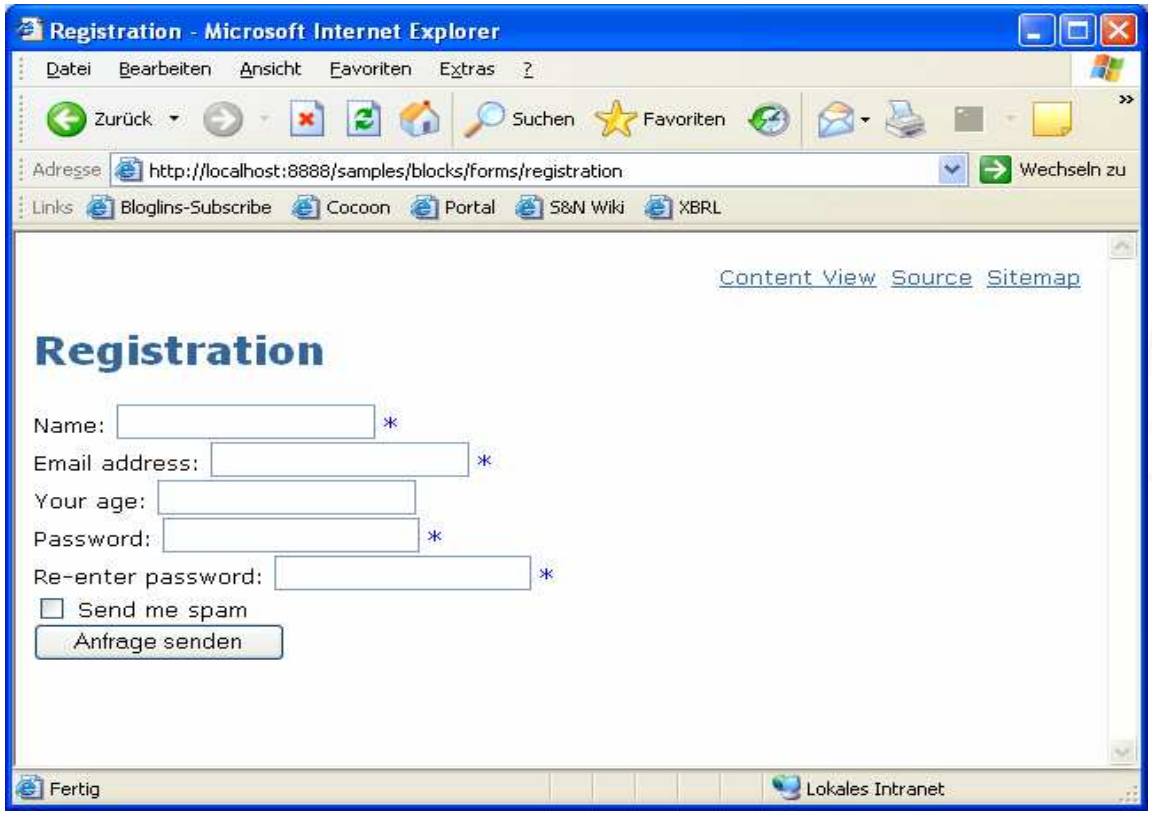
Step 4: Forms and Flow

- Use Cocoon Flow as the controller or „glue“
- Java Script on the server
- Has access to Forms model
- Forms Framework takes care of validation
- Predefined Functionality

Forms and Flow

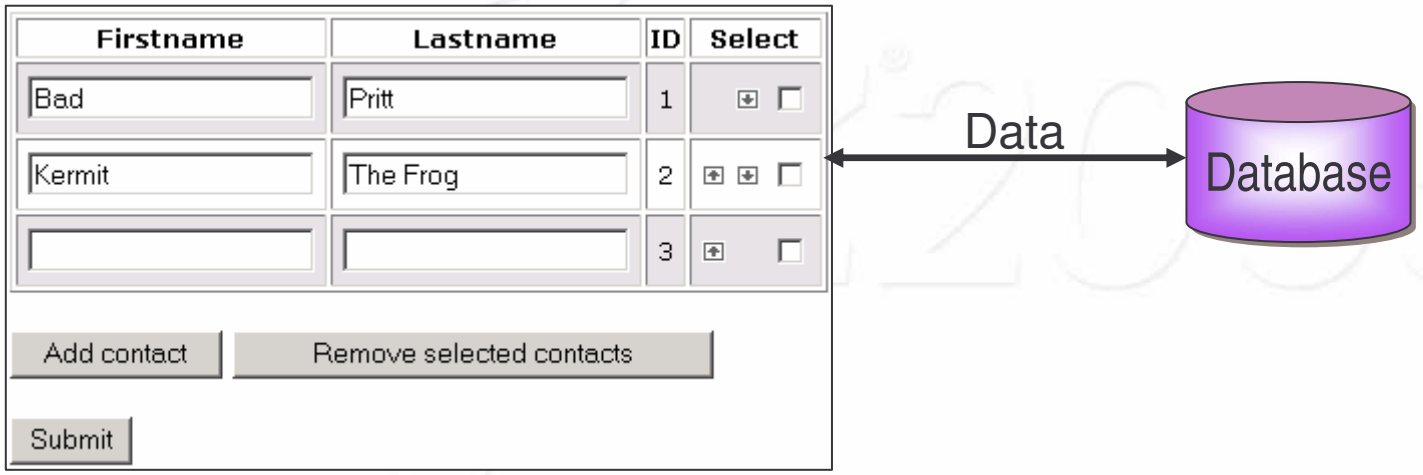
```
cocoon.load("resource://org/apache/cocoon/forms/flow/javascript/Form.js");

function formsample() {
    var form = new Form("formsample_model.xml");
    form.showForm("formsample-display-pipeline");
    cocoon.sendPage(
                    "formsample-success-pipeline.jx");
}
```

Success


Registration - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten
Adresse http://localhost:8888/samples/blocks/forms/registration
Wechseln zu
Links Bloglins-Subscribe Cocoon Portal S&N Wiki XBRL
Content View Source Sitemap
Registration
Name: *
Email address: *
Your age:
Password: *
Re-enter password: *
Send me spam
Anfrage senden
Fertig
Lokales Intranet

Web Applications
Business Layer

Simple Web Application



Business Layer: it is up to you!

- Use FlowScript all-the-way (bad!)
- Use some O/R mapping (OJB, Hibernate) from inside FlowScript (slightly better)
- FlowScript + Java facade, providing access to BO and methods on back-end
- (opt.) Avalon (configuration, monitoring, logging)

Binding

- Forms usually used for editing data
- Declare binding between widgets and business objects instead of implementing it
- Support for binding to JavaBeans and XML documents

Binding

```
<fd:field id="name" required="true">
  …
</fd:field>
<fd:field id="email" required="true">
  …
</fd:field>
<fd:field id="age">
  …
</fd:field>
```

Definitions

```
<fb:context xmlns:fb="http://apache.org/cocoon/forms/1.0#binding" path="/" >
    <fb:value id="name" path="name"/>
    <fb:value id="email" path="email"/>
    <fb:value id="age" path="age"/>
</fb:context>
```

Binding

Bean:

```
public class PersonBean {
    public void setName(String name);
    public String getName();
    public void setEmail(String email);
    public String getEmail();
    public void setAge(long age);
    public long getAge();
}
```

Binding

```
<fd:field id="name" required="true">
  …
</fd:field>
<fd:field id="email" required="true">
  …
</fd:field>
<fd:field id="age">
  …
</fd:field>
```

Definitions

```
<fb:context xmlns:fb="http://apache.org/cocoon/forms/1.0#binding" path="/" >
    <fb:value id="name" path="name"/>
    <fb:value id="email" path="email"/>
    <fb:value id="age" path="age"/>
</fb:context>
```

Binding

XML:

```
<person>
    <name></name>
    <email></email>
    <age></age>
</person>
```

Binding – Load/Save from Flow

```
cocoon.load("resource://org/apache/cocoon/forms/flow/javascript/Form.js");
function formsample() {
    var form = new Form("formsample_model.xml");

    form.createBinding("formsample_binding.xml");

    var bean = …

    form.load(bean);

    form.showForm("formsample-display-pipeline");
    form.save(bean);

    cocoon.sendPage("formsample-success-pipeline.jx");
}
```

Get form-object

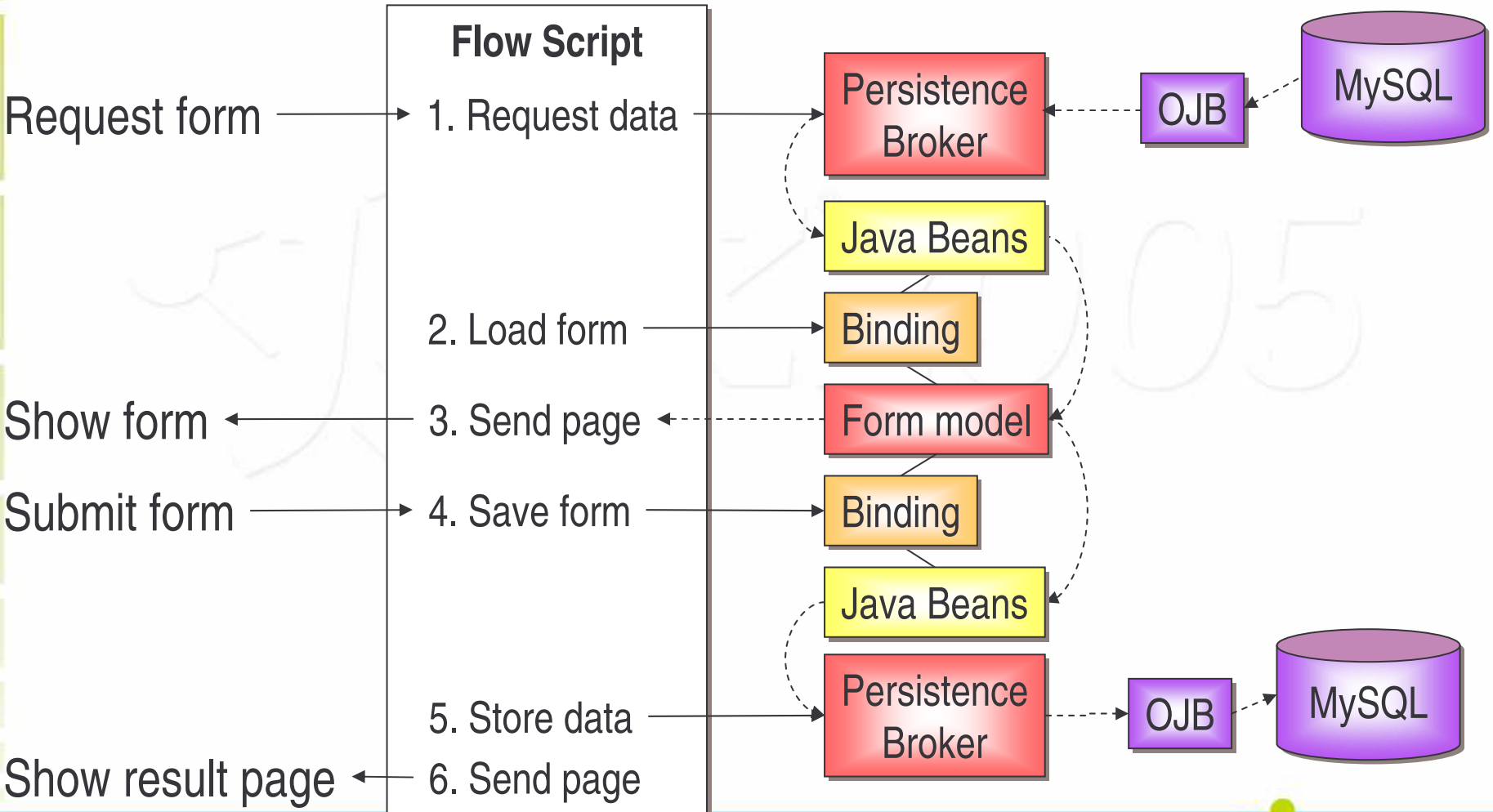
Get binding-object

Get bean

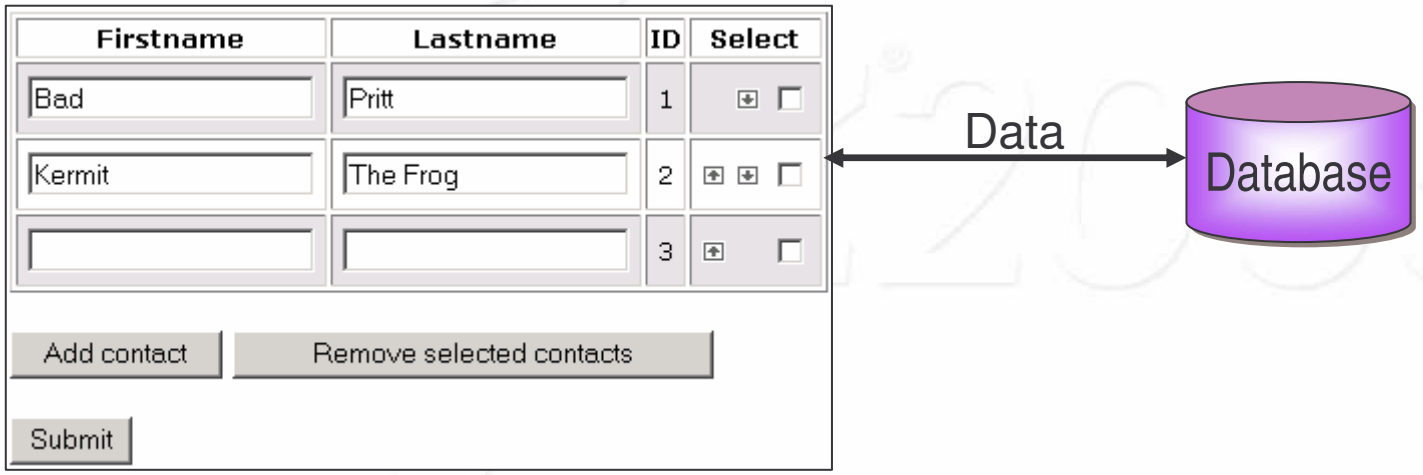
Fill form from bean

Fill bean from form

Sample Web Application



Simple Web Application



Building Web Applications

- The Easy Way
  - Use Cocoon ☺
  - With Flow and Forms
  - Add your own business layer
  - Benefit from separation of concerns
- But
  - The first steps might be hard!
  - Don‘t give up – the effort pays back

Building Web Applications

- There are many more features like
  - Dynamic form generation
    - Widgets depend on selected values/objects
    - Selection lists etc.
  - Hooks in the validation and binding framework
  - Prepared functionality for flow and jxtg
- You are not locked in – use what you need
  - Spring, Hibernate etc.

Outlook

Summary

- Stable platform (3 years+)
- Large community
  - Including large corporations
  - "Awareness" in the public is growing fast
- Separation of Concerns
  - Team Development
- Optimized for performance and stability
  - Caching, Pooling

Summary

- 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?
  - "Could be better" documentation
    - Books are available / Wiki
  - 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

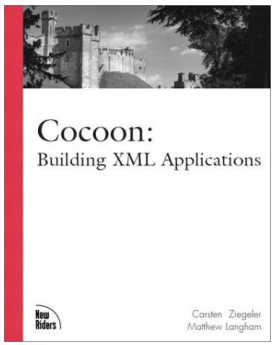
And Development Continues!

- 2.1.7 Released
- Release Early – Release Often ☺
- New features are constantly added
  - New Blocks
- Enhanced
  - Usability
  - Performance
- Bug Fixes
- Next major innovation: Cocoon 2.2

Further Information

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

Cocoon 2.2
Session WA17 / X8
Cocoon Portal
Session E28 / WA15 / X6



Thanks for your attention!

Questions?