Advanced WebApplications using

Reinhard Pötz, Apache Cocoon Comitter

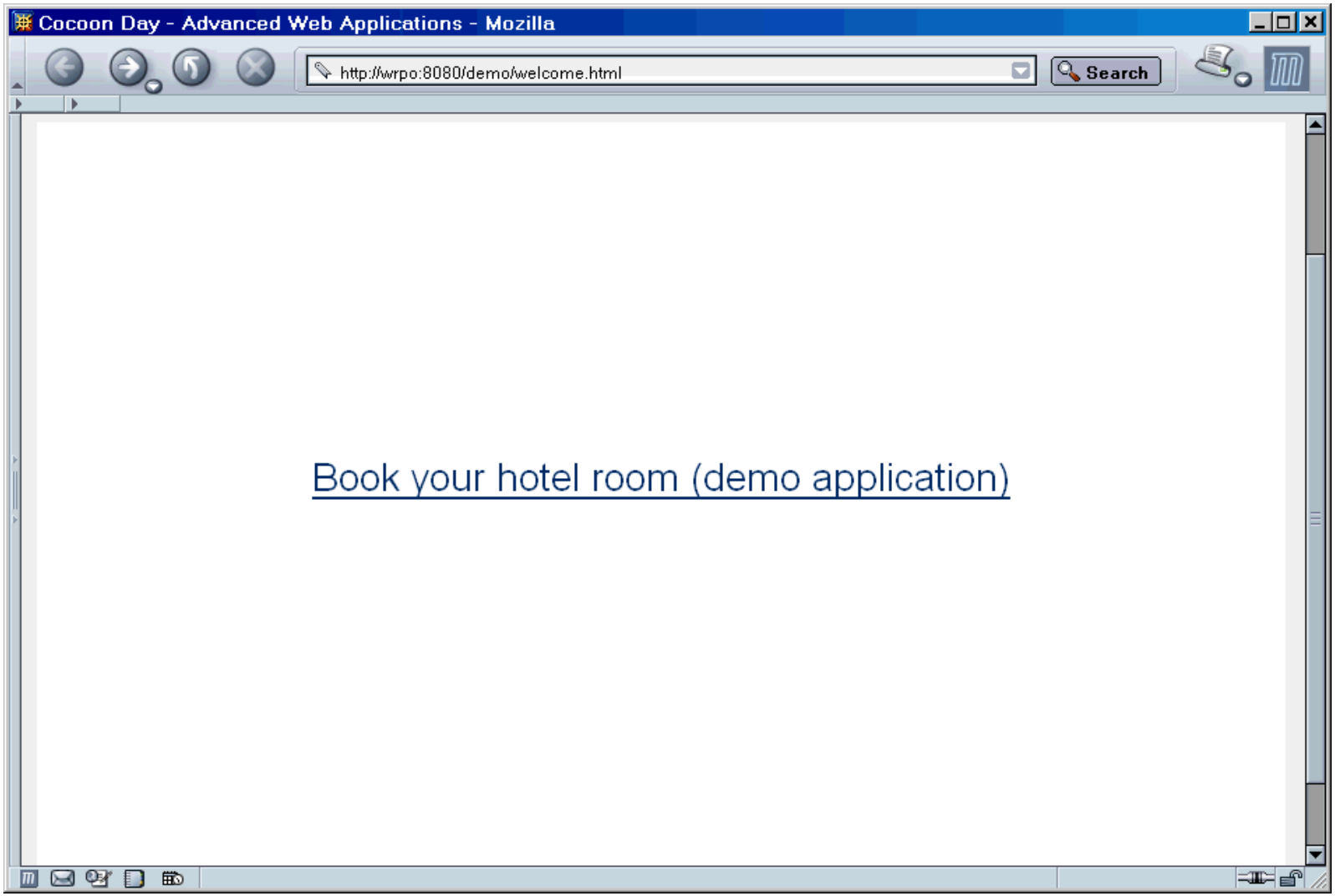
reinhard@apache.org
http://www.poetz.cc

CocoonDay, 2003-11-18, Vienna

Goals

- give an **overview** how to implement Web Applications with Cocoon2
- show **design patterns**
- focus on the **Cocoon Control Flow**
- **no details** on related aspects but hints
- after this presentation you should have a starting point for **your first application** using the Cocoon Control Flow

Demo application



Thinking in layers ...

goals

- easily maintainable
- reuseable

reached by

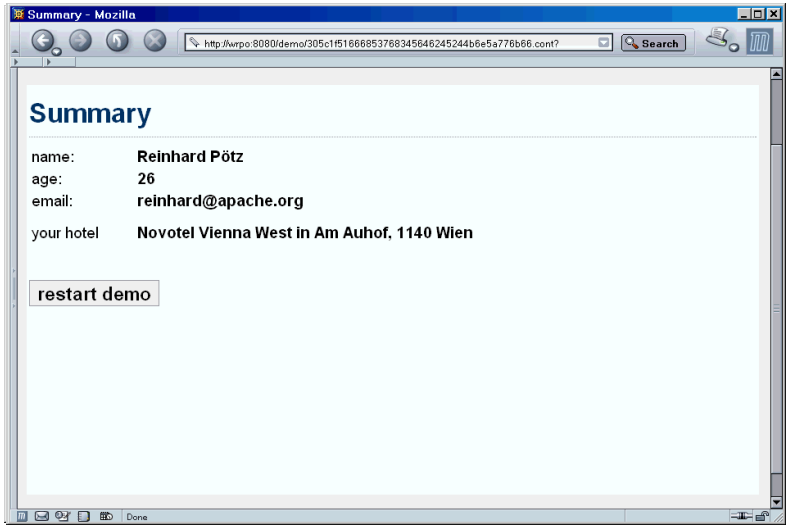
... separation of

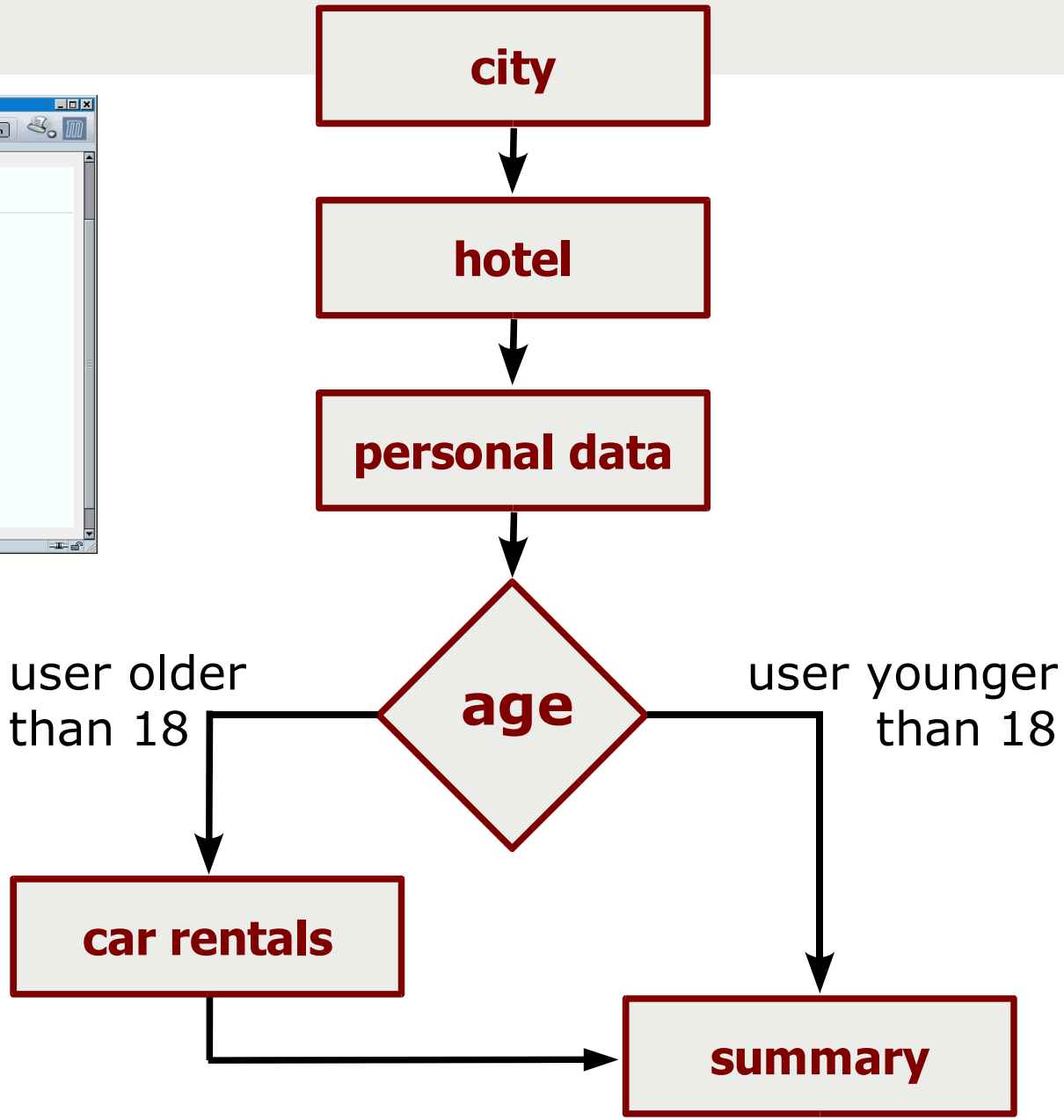
- data **model**
- **view** layer
- application **flow**

enforces

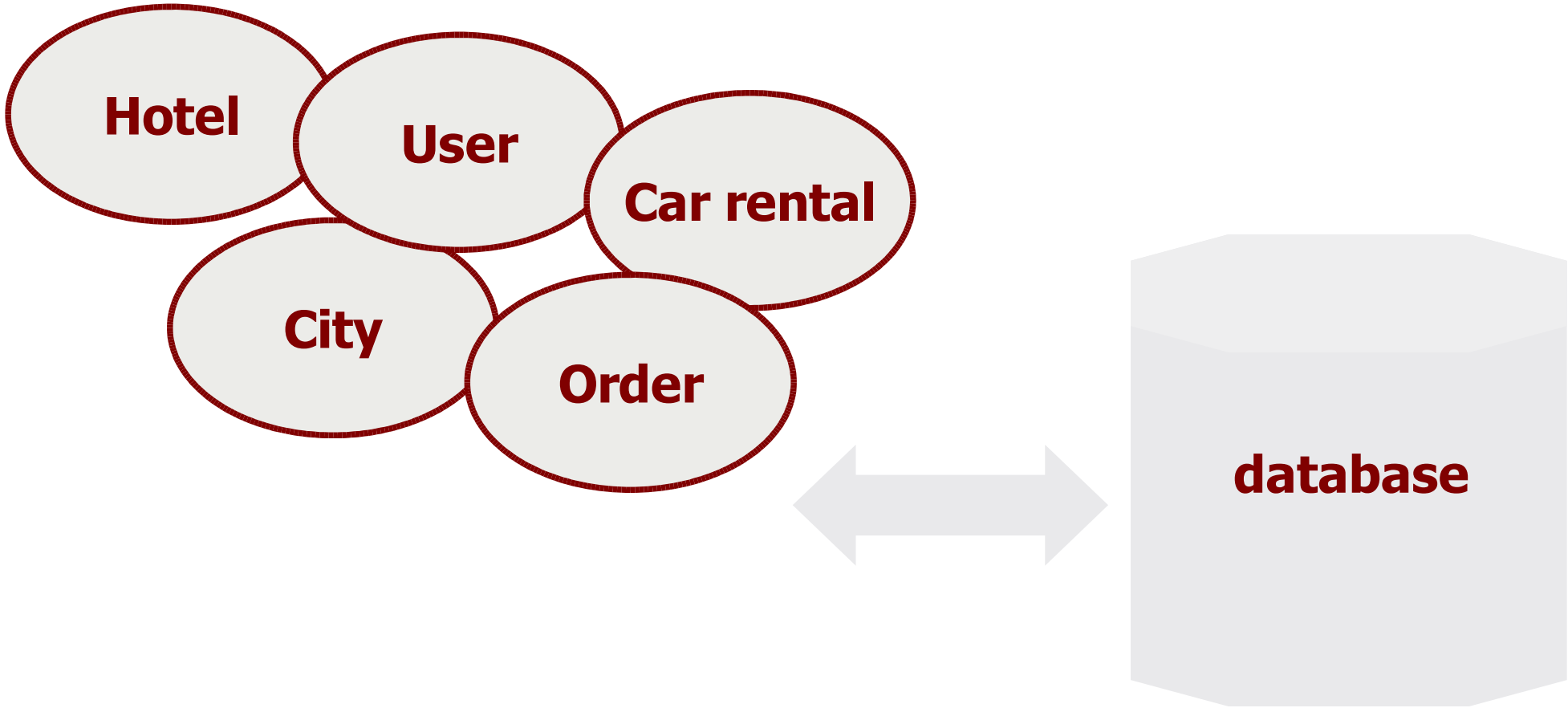
clear contracts

Page flow

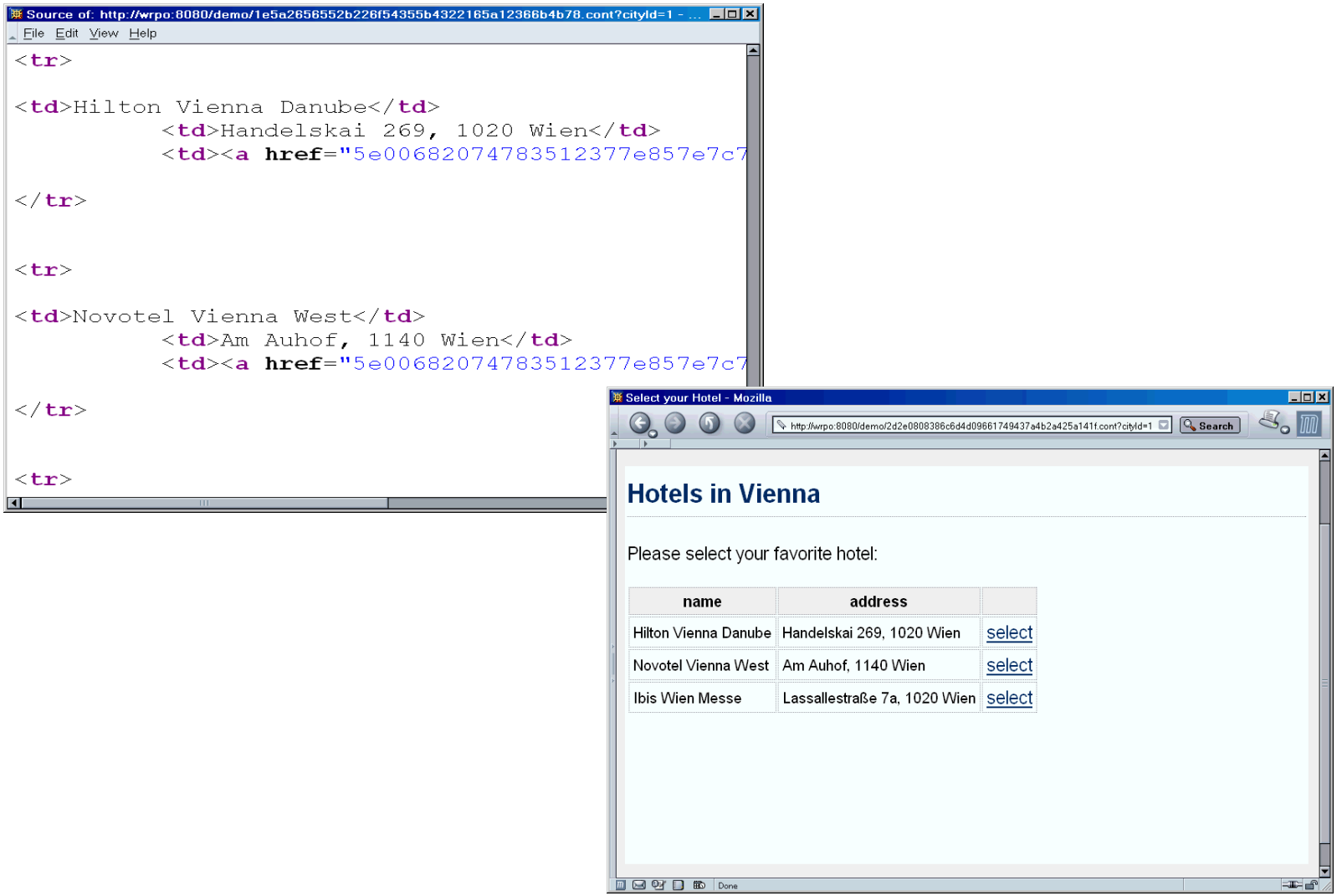

Summary - Mozilla
Search
Summary
name: Reinhard Pötz
age: 26
email: reinhard@apache.org
your hotel Novotel Vienna West in Am Auhof, 1140 Wien
restart demo
Done


city
hotel
personal data
age
user older than 18
user younger than 18
car rentals
summary

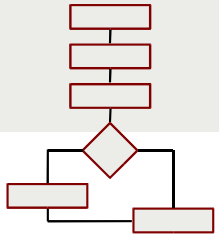
Data model



View layer

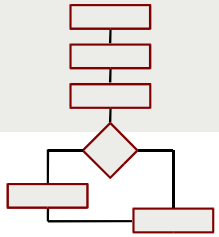

Source of: http://wrpo:8080/demo/1e5a2656552b226f54355b4322165a12366b4b78.cont?cityId=1 - ...
File Edit View Help
<tr>
<td>Hilton Vienna Danube</td>
<td>Handelskai 269, 1020 Wien</td>
<td><a href="5e006820747835123 77e857e7c7
</tr>
<tr>
<td>Novotel Vienna West</td>
<td>Am Auhof, 1140 Wien</td>
<td><a href="5e006820747835123 77e857e7c7
</tr>
<tr>
Select your Hotel - Mozilla
http://wrpo:8080/demo/2d2e0808386c6d4d09661749437a4b2a425a141f.cont?cityId=1
Search
Hotels in Vienna
Please select your favorite hotel:
name
address
Hilton Vienna Danube
Handelskai 269, 1020 Wien
select
Novotel Vienna West
Am Auhof, 1140 Wien
select
Ibis Wien Messe
Lassallestraße 7a, 1020 Wien
select
Done

Implementing the Flow layer



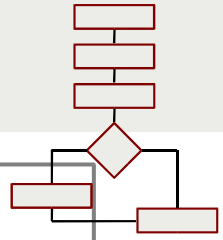
- **distributed over all pages**
  page flow is embedded in the hard wired links between the pages
  (often used in PHP/ASP/JSP applications)
  - hard to understand spathetti code, not maintainable, sideeffects
- **MVC (model view controller)**
  the controller is used to process requests and to select the views
  (e.g. implemented by the Struts framework or solutions based on Actions in Cocoon)
  - application is fragemented and often it is difficult to understand the logic
  - explicit state handling
- **MVC+**
  page flow is described as a sequential program using continuations
  (e.g. Cocoon Advanced Control Flow – using Flowscript)
  - very easy to understand the page flow
  - implicit state handling (continuations)

Flowscript – Integration into the Cocoon world [1]



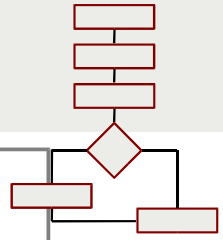
```
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
  <map:flow language="javascript">
    <map:script src="flow.js"/>
  </map:flow>
  <map:pipelines>
    <map:pipeline>
      <map:match pattern="booking.html">
        <map:call function="booking"/>
      </map:match>
    </map:pipeline>
</map:sitemap>
```

Flowscript – Example (flow.js)



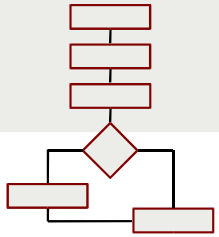
```
function booking() {
...

  var cities = getAllCities();
  cocoon.sendPageAndWait( "screens/destination.html",
     {
        cities : cities
     }
  );

  var city = getCity( cocoon.request.cityId, cities );
  var hotels = city.getHotels();
  cocoon.sendPageAndWait( "screens/hotellist.html",
     {
       cityName : city.getName(),
       hotels   : hotels
     }
   );

...
}
```

Continuations ... what? [1]



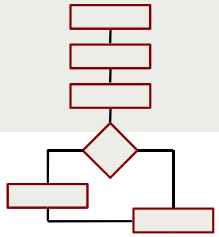
```
function booking() {
...

  var cities = getAllCities();
  cocoon.sendPageAndWait( "screens/destination.html",
    {
       cities : cities
    }
  );

  var city = getCity( cocoon.request.cityId, cities );
  var hotels = city.getHotels();
  cocoon.sendPageAndWait( "screens/hotellist.html",
    {
      cityName : city.getName(),
      hotels   : hotels
    }
   );

...
}
```

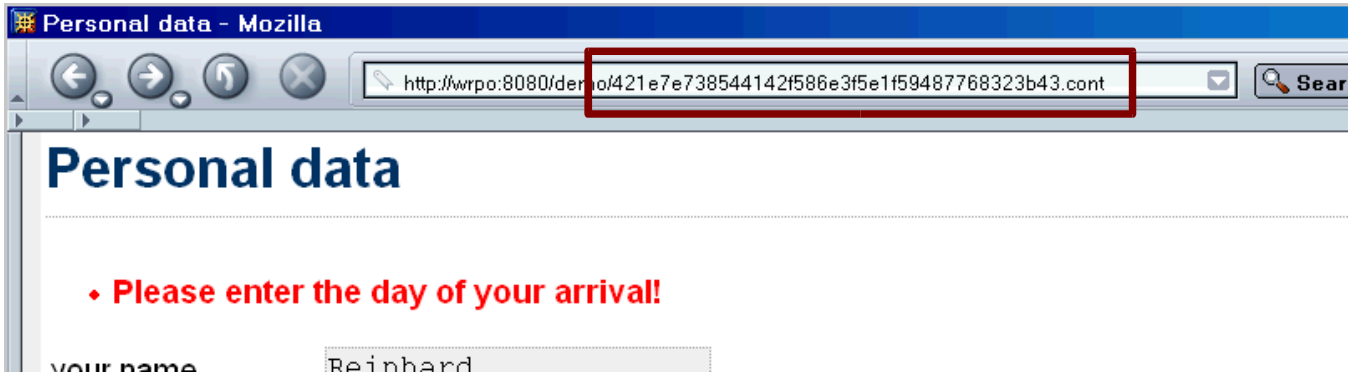
Continuations ... what? [2]



- **Continuations ...**
  - **know where the program execution stopped**
  - are tied to a stack (which is shared between all continuations)
  - contain local variables
  - are created after cocoon.sendPage**AndWait**(...)
  - have an unique identifier
  - are light-weight Java objects

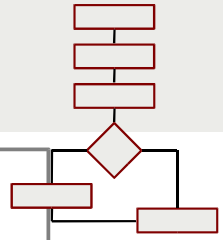
Flowscript – Integration into the Cocoon world [1]





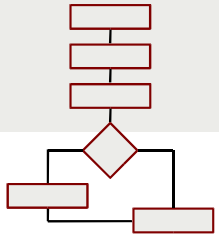

```
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
  <map:pipelines>
    <map:pipeline>
      <map:match pattern="*.cont">
        <map:call continuation="{1}"/>
      </map:match>
    </map:pipeline>
</map:sitemap>
```

Continuations ... what? [3]



```
function booking() {
...

  var cities = getAllCities();
  cocoon.sendPageAndWait( "screens/destination.html",
    {
       cities : cities
    }
  );

  var city = getCity( cocoon.request.cityId, cities );
  var hotels = city.getHotels();
  cocoon.sendPageAndWait( "screens/hotellist.html",
    {
      cityName : city.getName(),
      hotels   : hotels
    }
   );

...
}
```

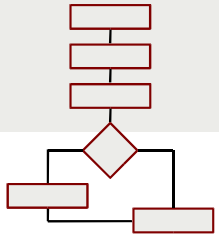
Flowscript supports ...



- **FOM (flow object model)**
  - access to the **environment** (request, response, session, context, sitemap parameters)
  - access to the **framework** (Avalon components, logging framework)
  - **page flow control** (cocoon.sendPage(), cocoon.sendPageAndWait(), cocoon.redirectTo()
  - access to the **continuations' tree**
- **Java Life Connect**
  - Access all Java classes available in the Cocoon classloader:

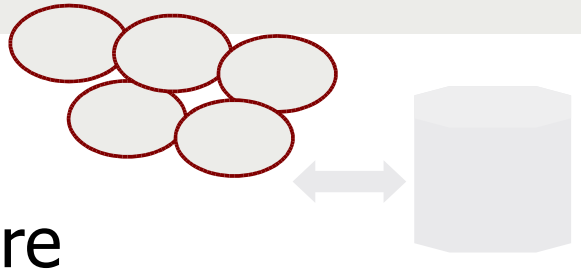
```
var map = new Packages.java.util.HashMap();
```

Why Javascript?



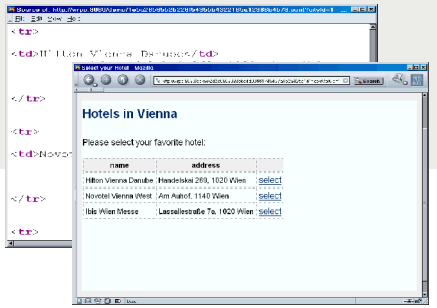
- known by many developers worldwide
- simpler than Java but nearly as powerful
- support for continuations
- faster roundtrips (save & reload)
- integrates well (see Live-Connect)
- not verbose

Integrate your backend



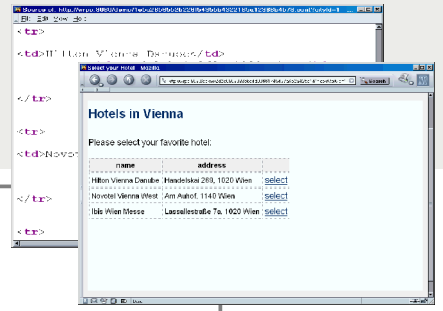
- because of the highly flexible architecture everything that Cocoon/Java/J2EE world offers is possible, e.g.
  - XML documents (e.g. generated by Cocoon pipelines)
  - EJB
  - Beans via O/R mapping tools
- the demo application uses OJB (PB API) – the Apache O/R mapping tool
  (see http://db.apache.og/ojb)

View Layer - Details



- **Power of Cocoon pipelines**
  provided by Cocoon pipelines (full power of Cocoon multi-client/multi-user/multi-language support/caching)
- **Templating**
  there are a few generators that are „flow aware“
  recommended templating approach:
  JXTemplateGenerator/Transformer
- **Inversion of Control**
  the view layer doesn't know where to find data (operates on JavaBeans or XML-fragments) – data objects are passed by the flow layer

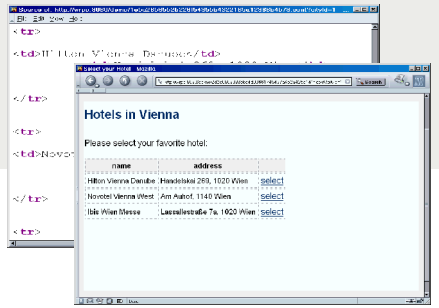
Passing Objects to the View




```
function booking() {
...

  var cities = getAllCities();
  cocoon.sendPageAndWait( "screens/destination.html",
     {
       cities : cities
     }
  );

  var city = getCity( cocoon.request.cityId, cities );
  var hotels = city.getHotels();
  cocoon.sendPageAndWait( "screens/hotellist.html",
     {
       cityName : city.getName(),
       hotels   : hotels
     }
   );

...
}
```

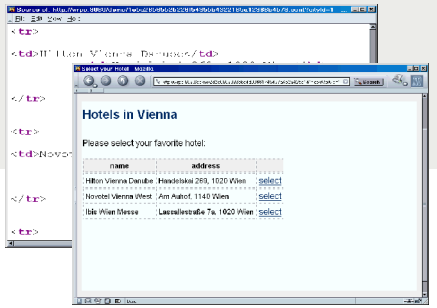
java.util.Collection

JXTemplateGenerator




```
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
  <map:components>
    <map:generators internal-only="file">
      <map:generator
        label="content"
        logger="sitemap.generator.jxpath"
        name="jx"
        src="org.apache.cocoon.generation.JXTemplateGenerator"/>
     </map:generators>
   </map:components>
   <map:pipelines>
    <map:pipeline internal-only="true">
      <map:match pattern="screens/*.html">
        <map:generate type="jx" src="screens/{1}.xml"/>
        <map:transform src="stylesheets/global.xsl"/>
        <map:serialize type="html"/>
      </map:match>
    </map:pipeline>
   </map:pipelines>
</map:sitemap/>
```

JXTemplateGenerator



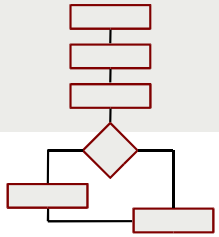
- access to the FOM objects
- support for JEXL and JXPath expressions
- very fast

```
<form action="${continuation.id}.cont">
    <select name="cityId">
      <jx:forEach var="city" items="${cities}">
         <option value="${city.id}">${city.name}</option>
      </jx:forEach>
    </select>
    <br/>
    <br/>
    <input type="submit" value="Make your choice"/>
  </form>
```

Cocoon Forms (aka Woody)

- get rid of all the explicit mappings from request parameters to beans (which could be harmful!)
- validation, strong datatyping, internationalization (i18n)
- following a widget approach
- special API for flowscript integration
- the community solution (will deprecate all former one-man shows like JXForms, XMLForms, ...)
- expect a release 2004/Q1 - Q2
  (personal opinoin of the author of this presentation ;-)

Conclusion



- **easily maintainable** page flow because of clean designs contracts and no fragmentation (find out in minutes what a web applications does!)
- **implicit state management** (you don't have to find out where the browser is – you always know it!)
- no back button problem
- **no trade-offs** between fast development cycles/rapid prototyping and clean designs
- perfectly fits into the Cocoon and J2EE world