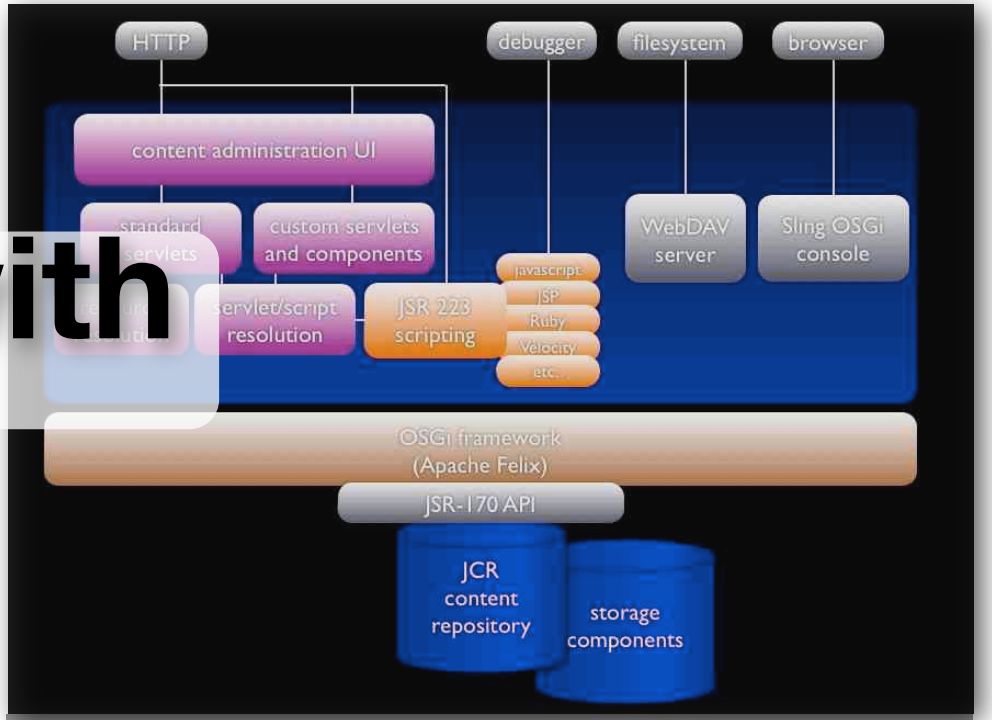
RESTful web applications with Apache Sling



Bertrand Delacrétaz

Senior Developer, R&D, Day Software, now part of Adobe

Apache Software Foundation Member and Director

http://grep.codeconsult.ch - twitter: @bdelacretaz - bdelacretaz@apache.org

ApacheCon NA 2010, Atlanta

slides revision: 2010-11-02

Day Software is now part of Adobe

APACHE felix

Jackrabbit

Sling

ApacheCon NORTH AMERICA 2010

everything is content

Sling

JCR API

The Java Content Repository
Tree of nodes and properties
JSR-170, JSR-283

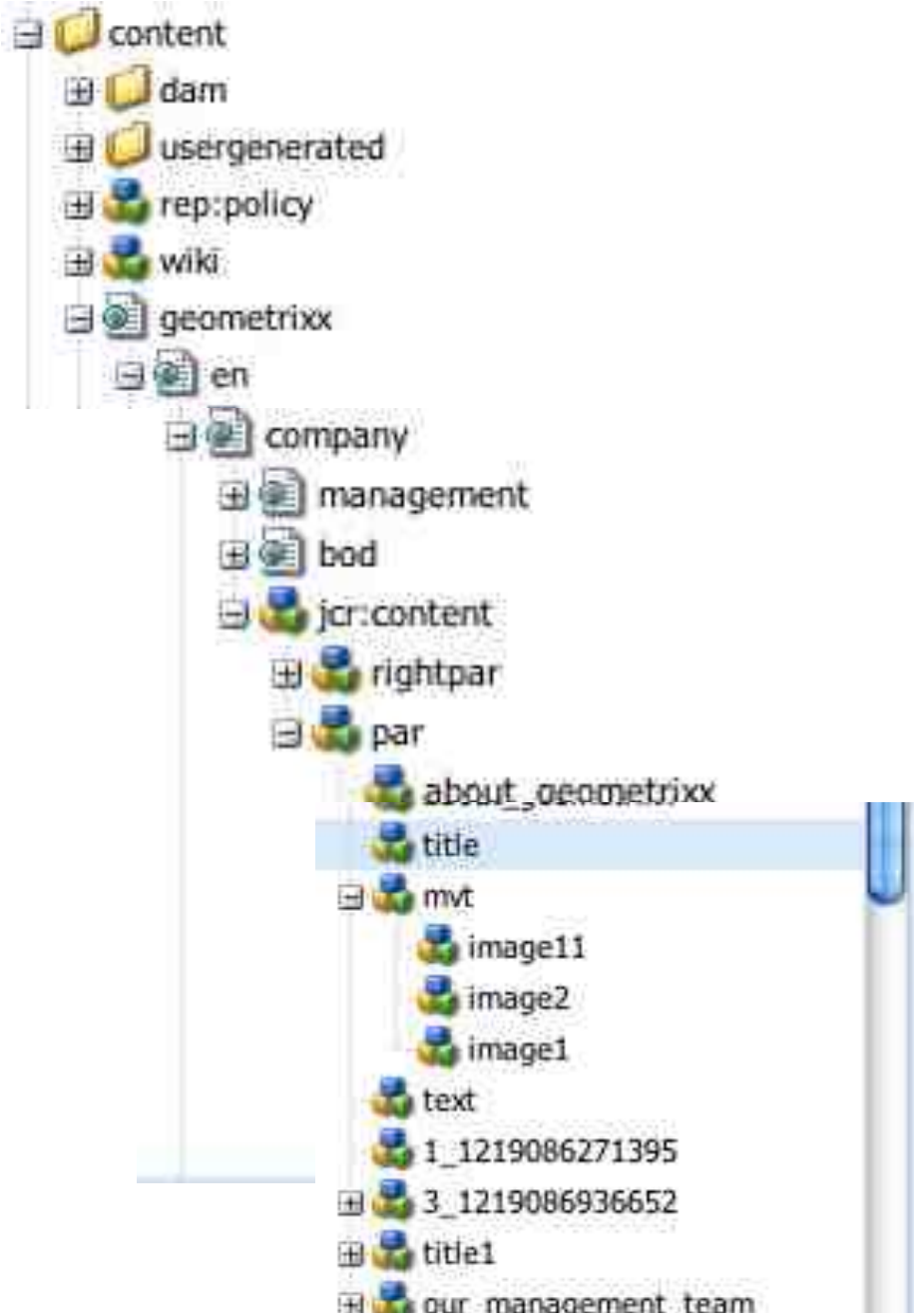
Text, images, media,
configurations, code, binary
OSGi bundles, etc...

URLs map to
resources, not
commands

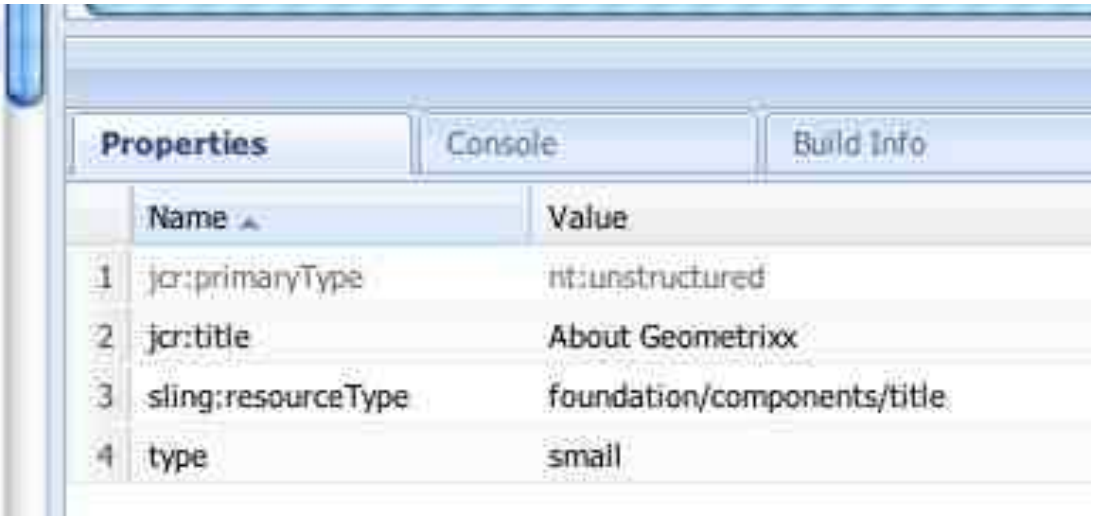
SLING RESTFUL WEB APPS

RESTful Web

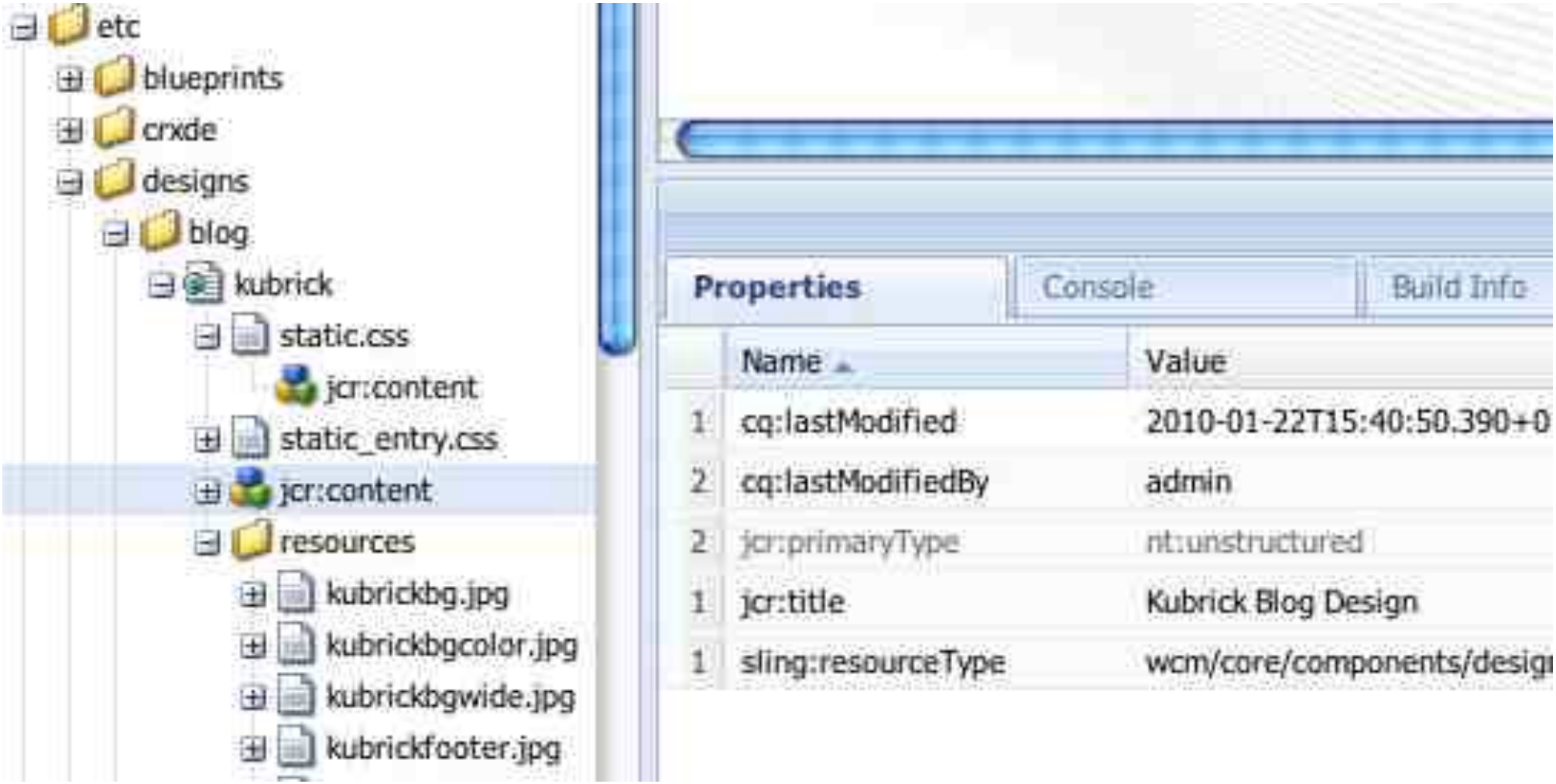
website content



/content
/geometrixx
/en
/company
/jcr:content
/par
/title



blog design content



/etc/designs/blog/kubrick/jcr:content

code content!

/libs
/cq
/code
/install
/xyz.jar

powered by

Apache Sling

Applications layer for JCR repositories

resource-based

script == servlet

«any» scripting language

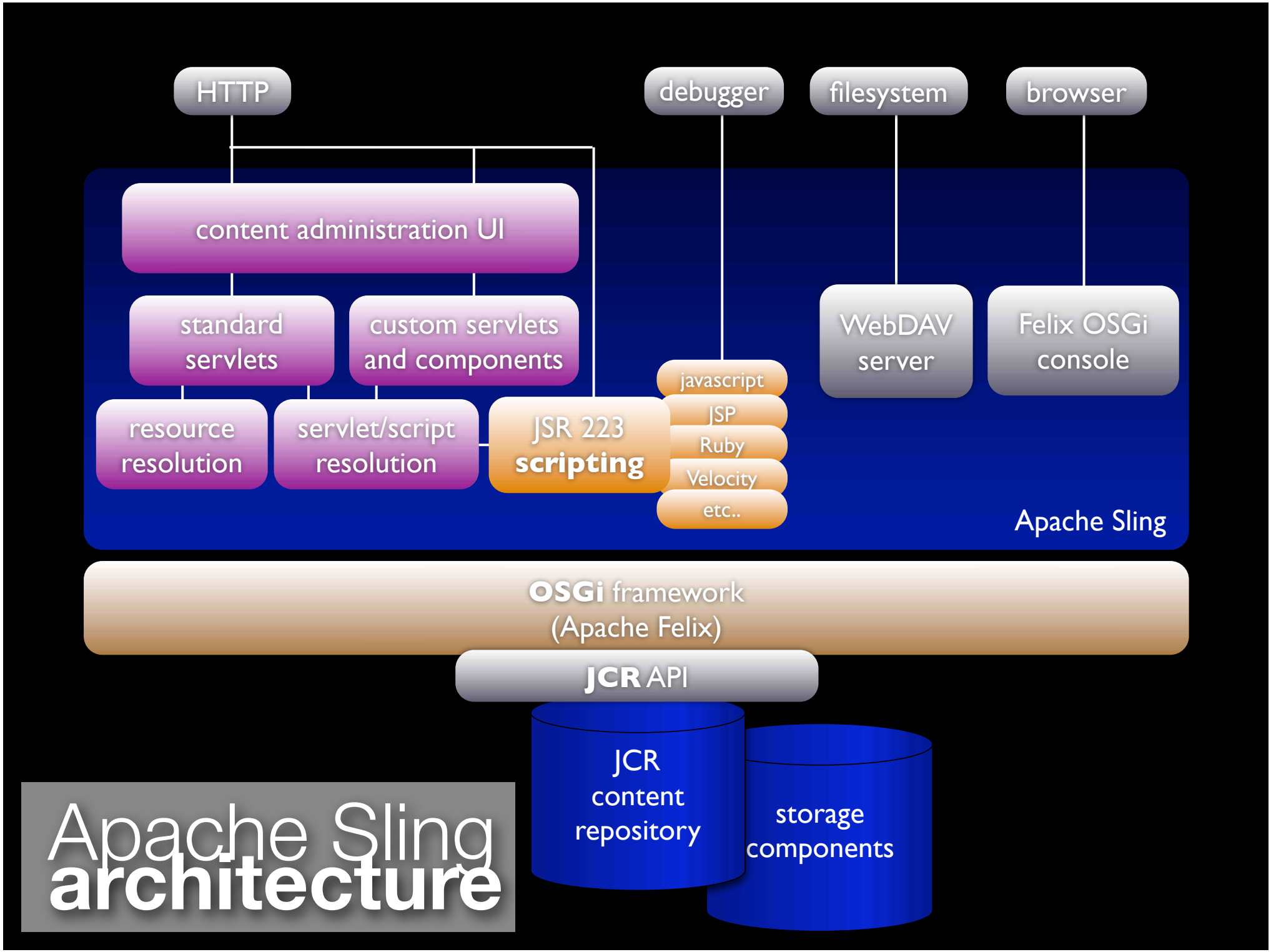
powerful default servlets

OSGi-based

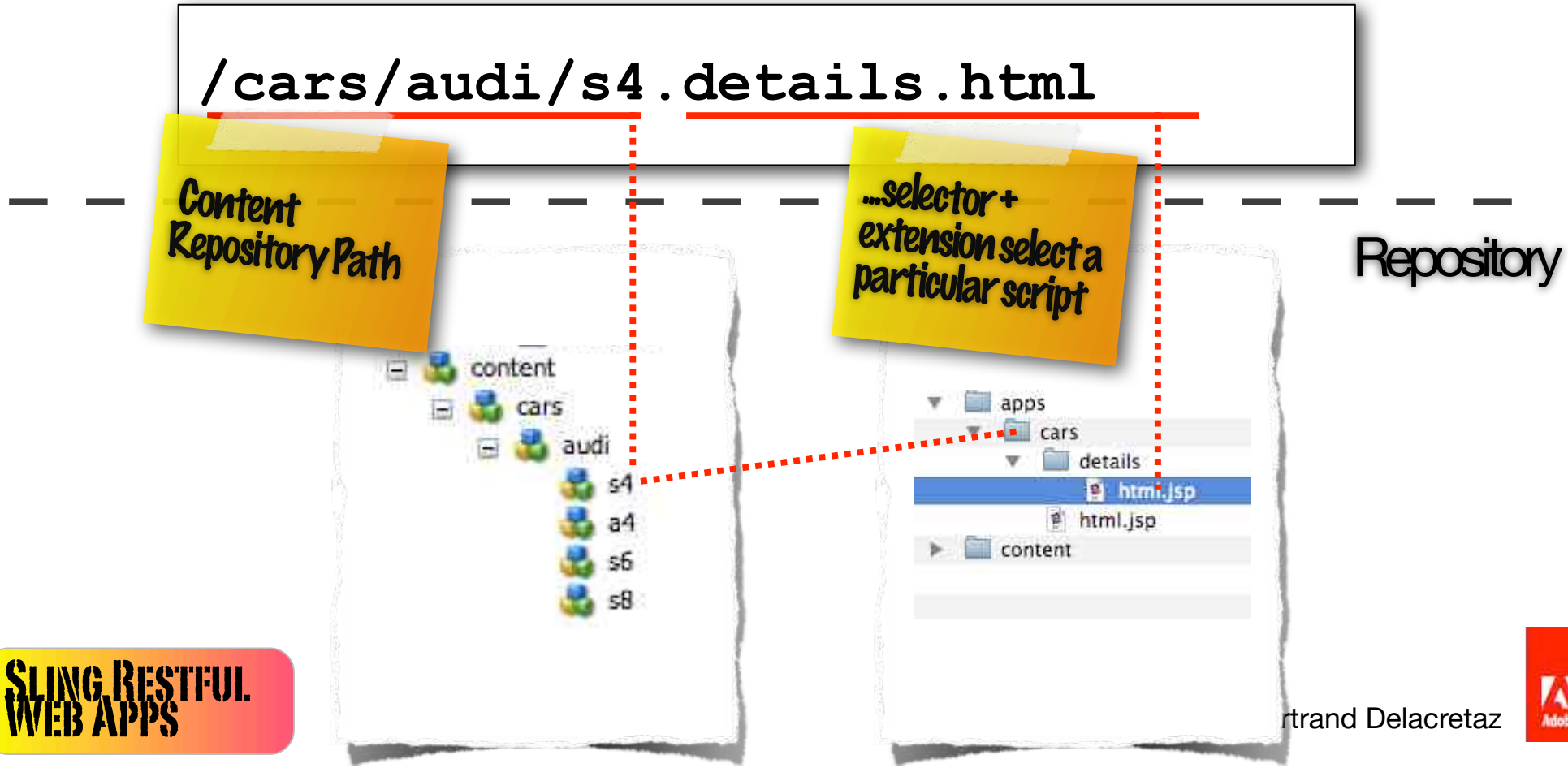
APACHE felix

Jackrabbit

Sling


HTTP
debugger
filesystem
browser
content administration UI
standard servlets
custom servlets and components
resource resolution
servlet/script resolution
JSR 223 scripting
javascript
JSP
Ruby
Velocity
etc..
WebDAV server
Felix OSGi console
Apache Sling
OSGi framework (Apache Felix)
JCR API
JCR content repository
storage components
Apache Sling architecture

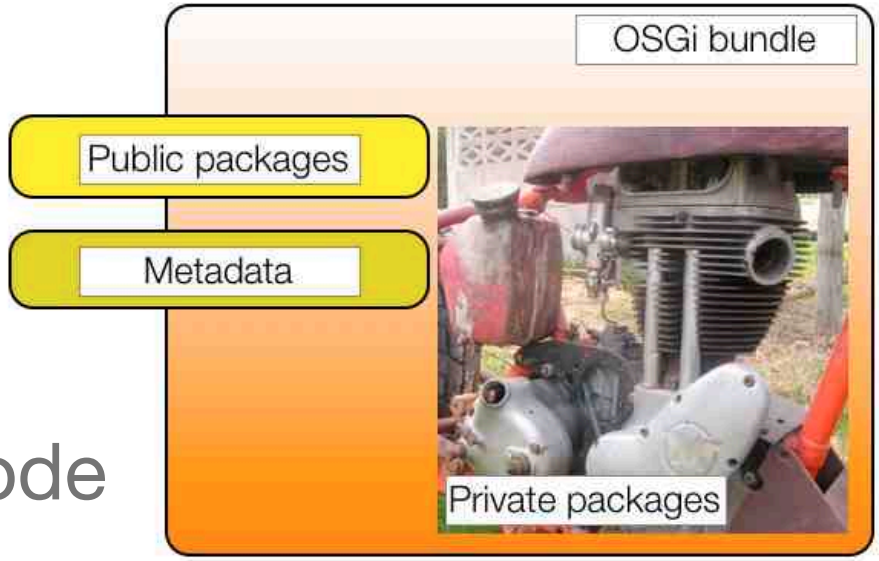
Reclaiming the Web:
Sling URL decomposition



SLING RESTFUL WEB APPS

Adobe

OSGi?



Great for **modularity**

Fosters **better structured** code

Dynamic **services** and **plugins**

Tooling needs to improve, but usable

OSGi **skills?** - OSGi way of thinking is new...

Asynchronous startup can be problematic if using declarative services

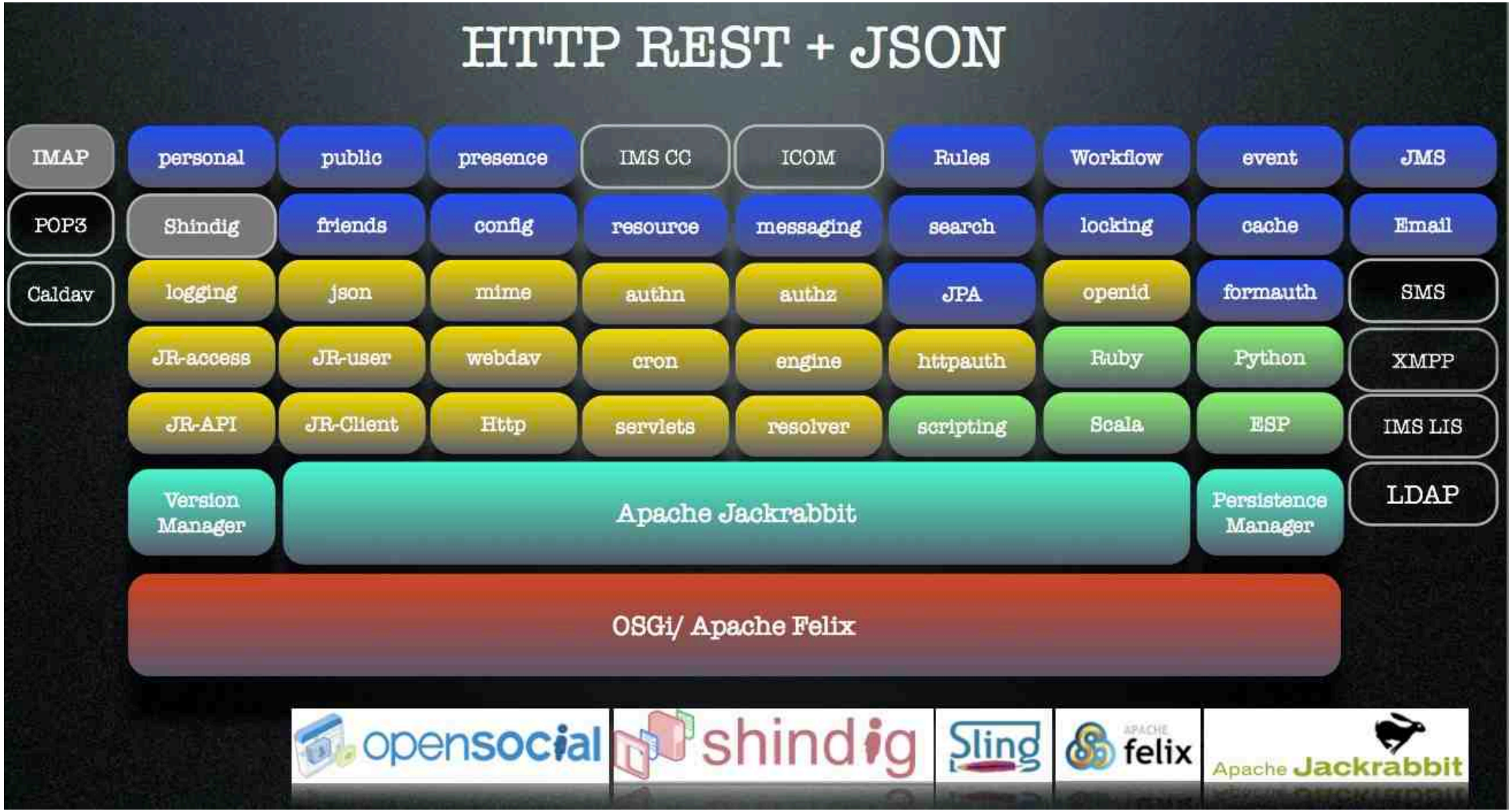
What

can you do with Sling?

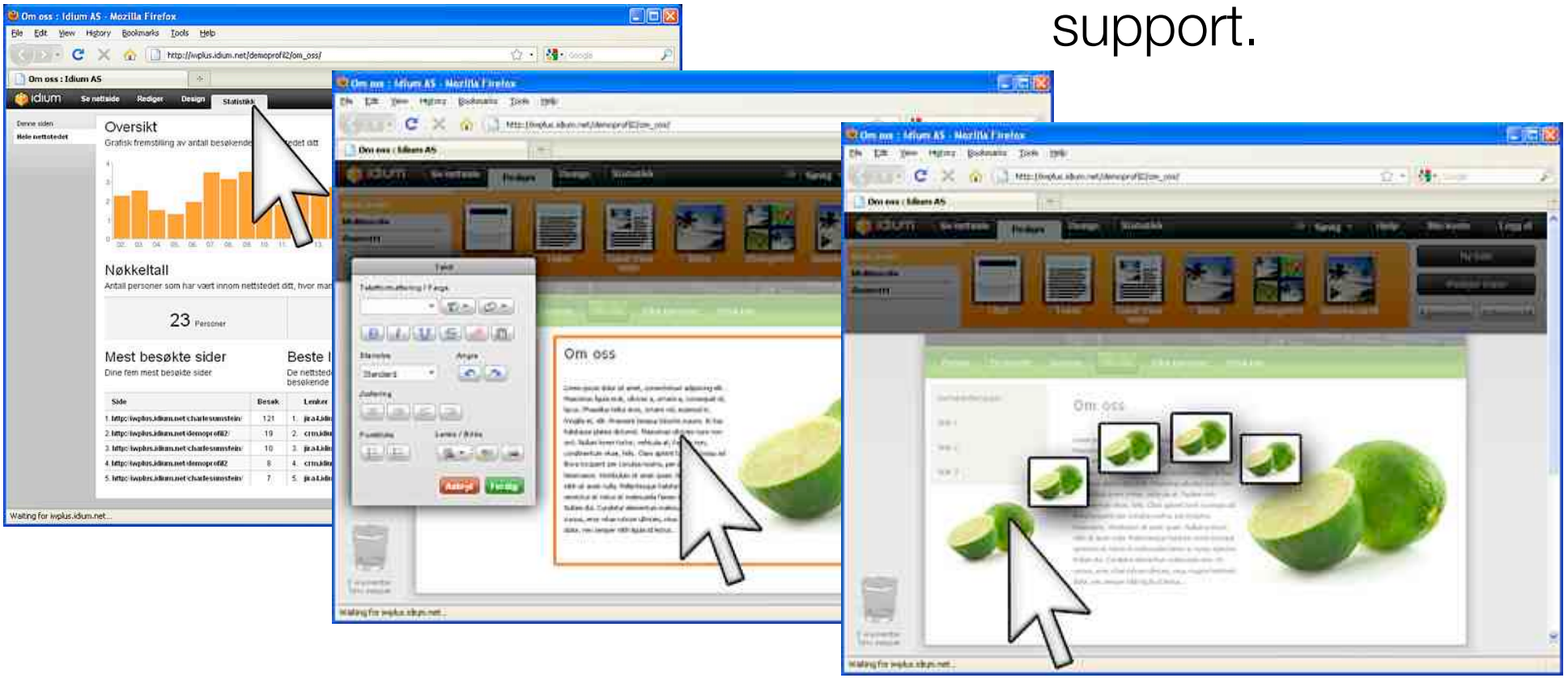
Sakai

Next-generation RESTful kernel for permeable, social, personal and remixable collaboration/ learning environment.

See http://sakaiproject.org/case-studies



idium.no - full-service hosted CMS integrated with CRM. Autoscales on Amazon EC based on JCR clustering and Sling event support.



Day CQ5

Web Content Management

Multi-channel + mobile

Customer Engagement Management

e-commerce

BIG websites

Digital Assets Management

Content Applications Platform

www.day.com

powered by

APACHE felix

Jackrabbit

Sling

Sling blog
46 lines of code

http://x42.ch/05.40.01

APACHE felix

Jackrabbit

Sling

Sling POST Servlet: create content

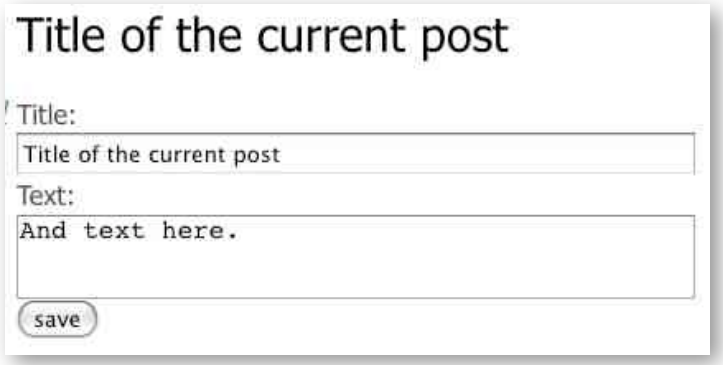
```
# POST to Sling
curl -F title=hi http://localhost:8888/foo
-> 200 OK

# GET created node in json format
curl http://localhost:8888/foo.tidy.json
{
   "jcr:primaryType": "nt:unstructured",
   "title": "hi"
}
```

HTML form for editing

```
<form method="POST">
  Title:
  <input type="text" name="title"/>
  Text:
  <textarea name="text"></textarea>
  <input type="submit" value="save"/>
  <input type="hidden"
    name=":redirect" value="*"/>
</form>
```

sling.js: initializes form fields

```
<script src="/system/sling.js"></script>
<form method="POST">
   ...(as in step 1)...
</form>

<!-- set form fields to current node values --
<script>Sling.wizard();</script>
```



sling.js: generate navigation

```
<ul>
  <li>
    <a href="/content/blog/*">[Create post]</a>
  </li>
  <script>
    var posts = Sling.getContent("/content/blog", 2);
    for(var post in posts) {
      document.write(
        "<li><a href='" + post + "'>"
        + posts[post].title + "</a></li>");
     }
  </script>
</ul>
```

we got a blog!

html form + Sling wizard() + Sling.getContent()

more at http://x42.ch/05.40.01

APACHE felix

Jackrabbit

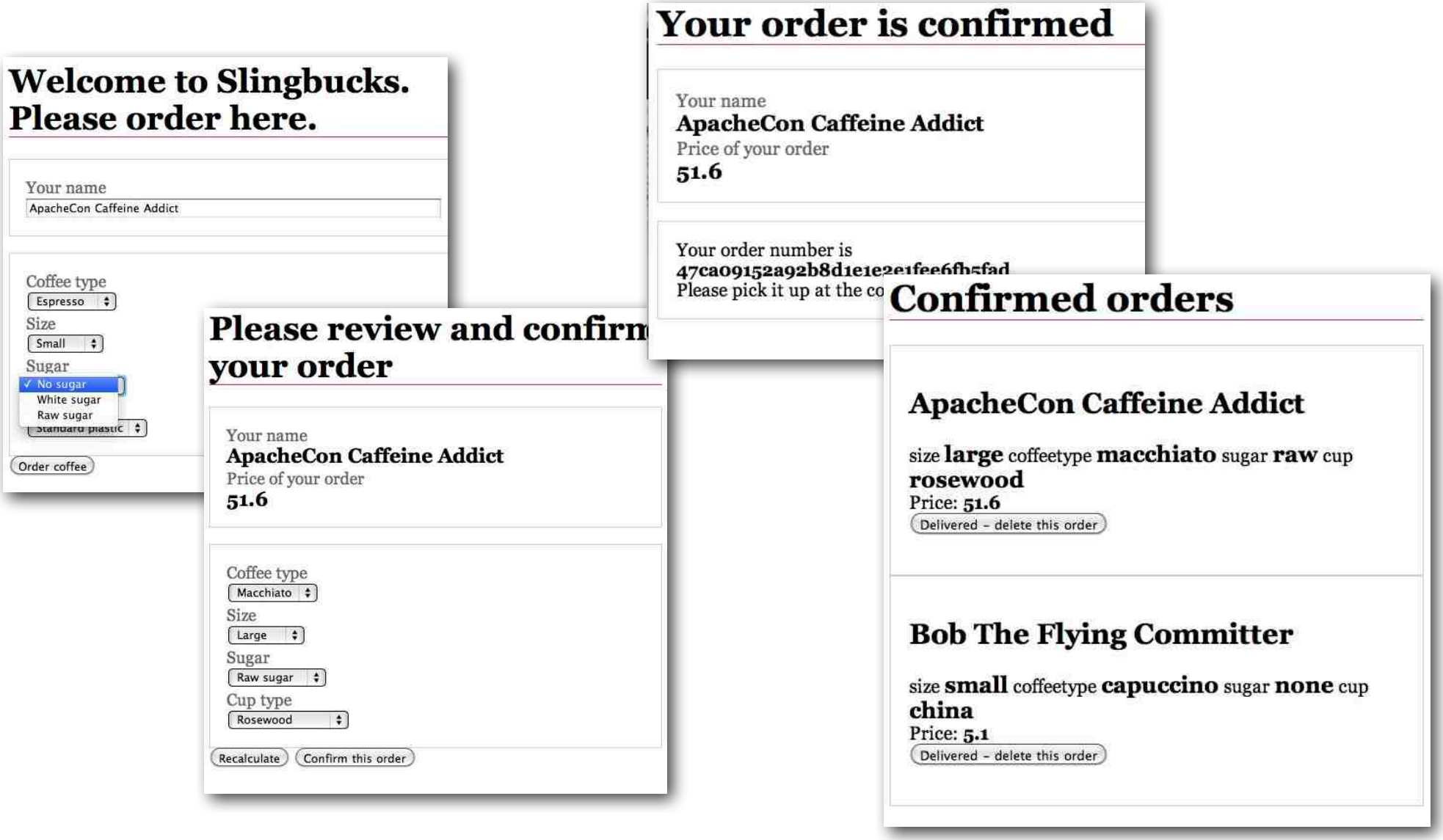
Sling

Slingbucks

RESTful coffee orders

Still a basic app, a bit more realistic

Slingbucks demo



SLING RESTFUL WEB APPS

Bertrand Delacretaz

Adobe

Slingbucks use case #1: order coffee

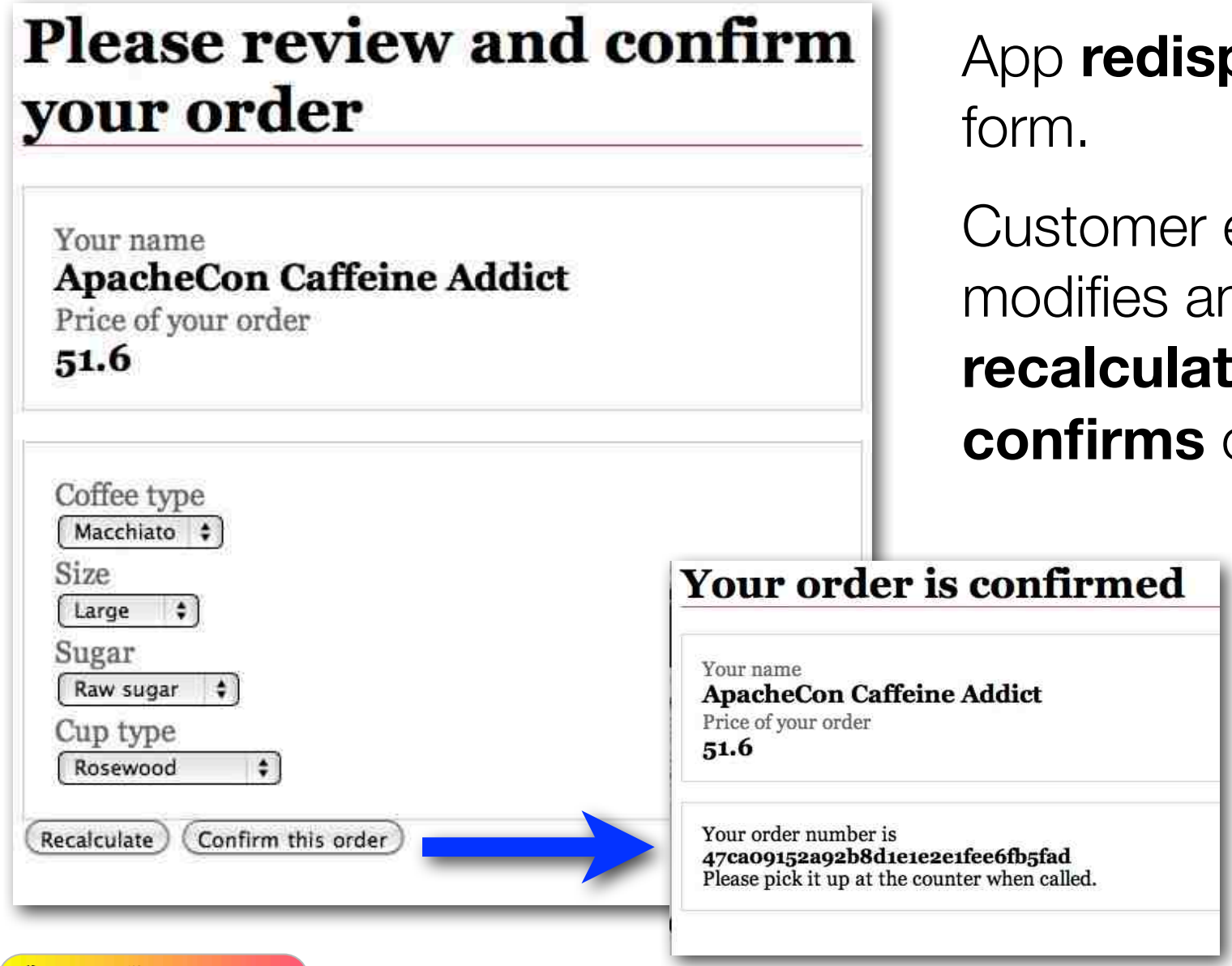
App displays **order form** with configurable options.

Customer indicates their name, options, **submits order.**

App generates **hard to guess** order ID.

App redisplays order for **confirmation** (use-case #2).

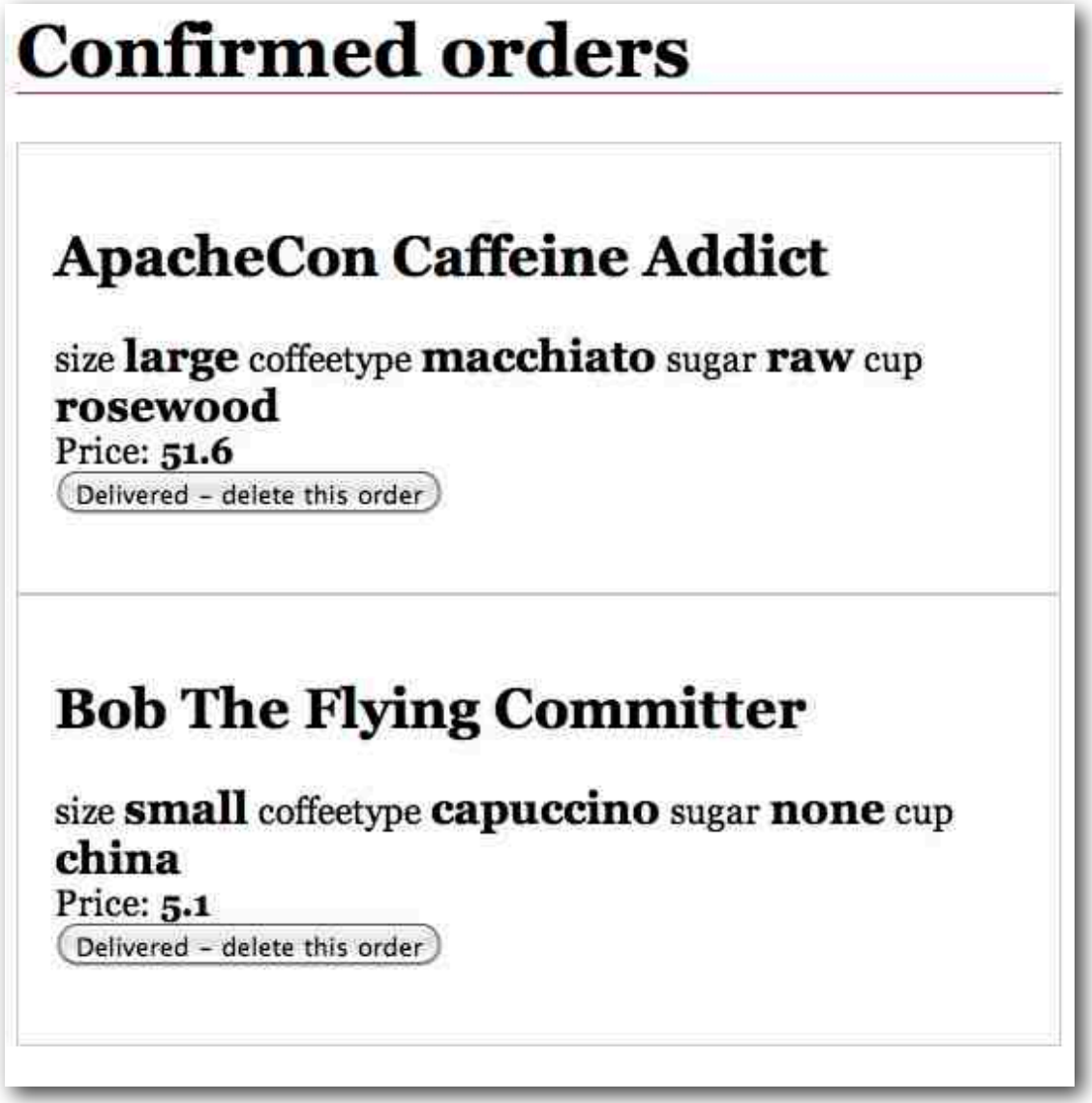
Slingbucks use case #2: confirm order



App **redisplays** order form.

Customer either modifies and **recalculates** price, or **confirms** order.

Slingbucks use case #3: process order



Order **moves** to the **private** Slingbucks employees area.

App displays a **list** of confirmed orders.

Employee delivers order and deletes it from list.

OO design:
Highlight names to find objects

App displays **order form** with configurable **options**.

Customer indicates their **name**, **options**, submits **order**.

App redisplays order for confirmation (use-case #2)

Just kidding ;-)

Everything is CONTENT

Slingbucks resources design

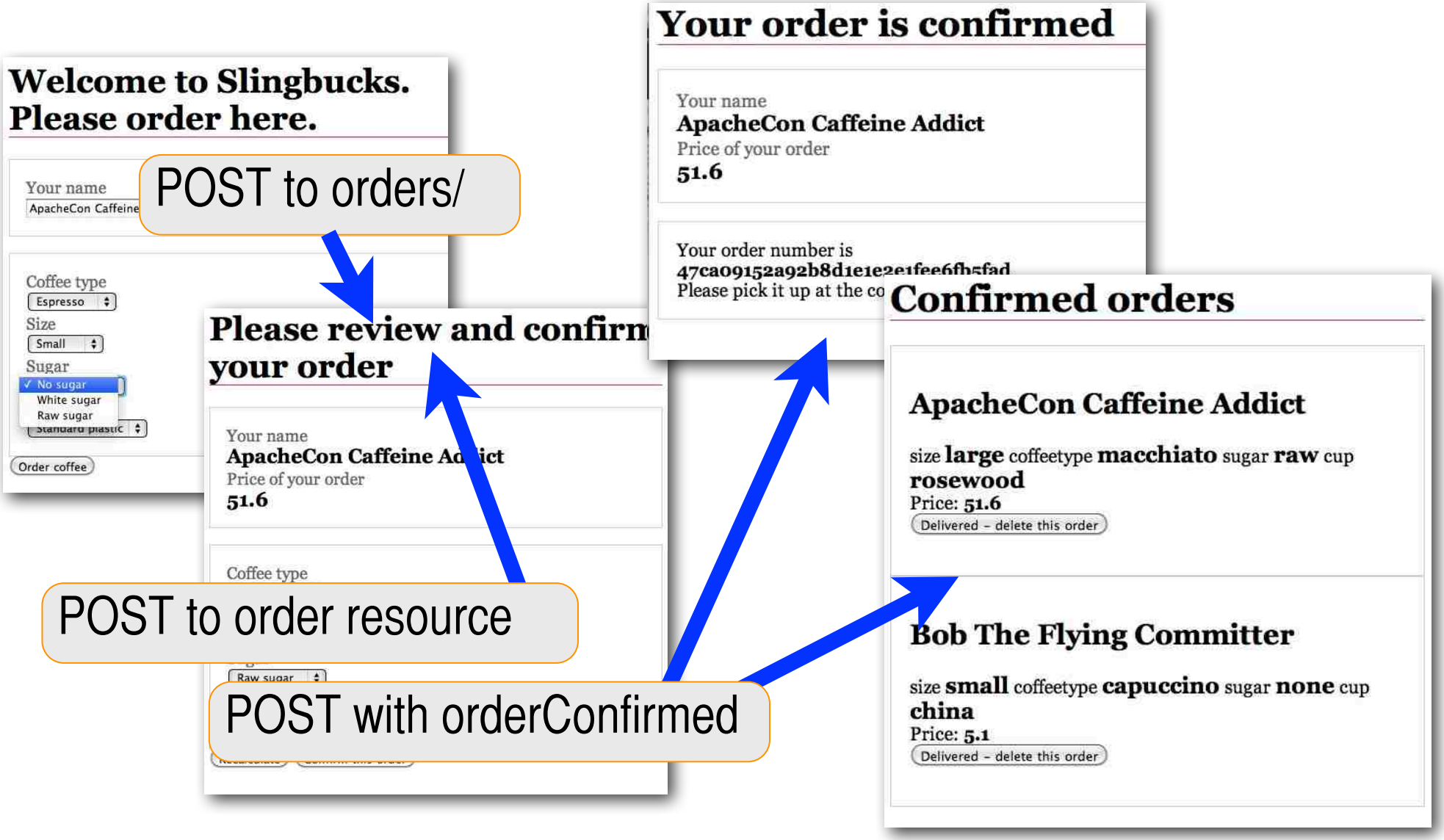
New order form:
http://slingbucks.com**/public/orders.html**

Order editing and confirmation (example):
http://slingbucks.com**/public/orders/54494da6029.html**

Price of an order:
Same but ending with **.price.html** selector

List of confirmed orders:
http://slingbucks.com**/private/confirmed.html**

Slingbucks demo

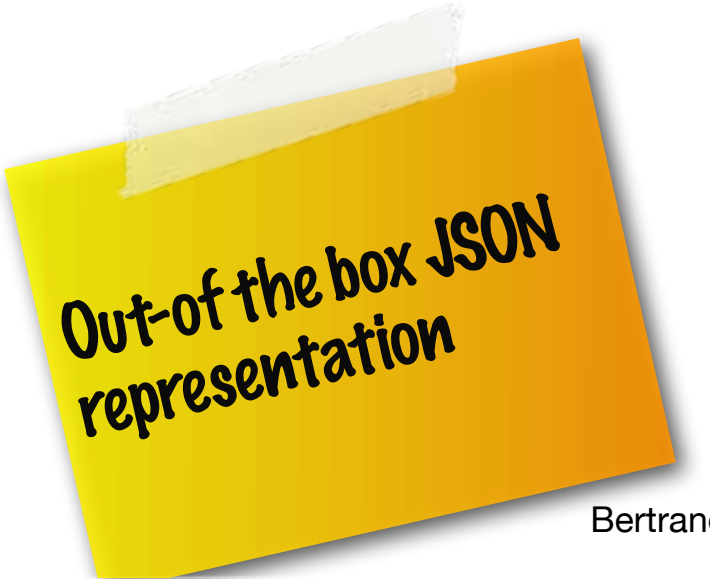


Geeks order coffee with *curl*... What else?


```
$ curl -D -
-F "customerName=Bob The Geek"
-F sling:resourceType=slingbucks/order
-F lastModified=""
-F opt_coffeetype=capuccino
-F opt_size=medium
-F opt_sugar=raw
-F opt_cup=rosewood
http://admin:admin@localhost:8080/content/slingbucks/public/orders/

HTTP/1.1 201 Created
Location: /slingbucks/public/orders/117936075d4de452cbba5b468
```

Coffee node content

```
$ curl http://localhost:8080/content/slingbucks/public/orders/
fad01d62f50aaca54209ae14c9505e3b.tidy.json

{
  "opt_size": "small",
  "customerName": "Anonymous Coffee Drinker",
  "opt_coffeetype": "espresso",
  "opt_sugar": "none",
  "sling:resourceType": "slingbucks/order",
  "opt_cup": "plastic",
  "lastModified": "Mon Nov 01 2010 18:31:01 GMT+0100",
  "jcr:primaryType": "nt:unstructured"
}
```



Slingbucks

code walkthrough

Hypermedia? Self-documenting?

```
<link
rel="slingbucks/options"
href="/content/slingbucks/readonly/
options.tidy.infinity.json"/>
...
<form method="POST"
action="/content/slingbucks/public/orders/"
...
<select name="opt_coffeetype">
<options>
<option value="espresso">Espresso</option>
<option value="capuccino">Capuccino</option>
<option value="macchiato">Macchiato</option>
</select>
...
```

public/orders.html
form provides all
required «API» info.

Slingbucks code: order ID generation

```
// Just provide an OSGi service that implements NodeNameGenerator
@Component
@Service
public class HexNodeNameGenerator
implements org.apache.sling.servlets.post.NodeNameGenerator {
...
    public String getNodeName(
            SlingHttpServletRequest request,
            String parentPath,
            ...)
    {
        if(SlingbucksConstants.ORDERS_PATH.equals(parentPath)) {
            return computeHardToGuessNodeName();
        }
        return null;
    }
}
```

Slingbucks code: move confirmed orders 1/2

```
@Component
public class ConfirmedOrdersObserver
implements EventListener, Runnable {

  @Reference
  private SlingRepository repository;

  /** Called by OSGi framework when component starts */
  protected void activate(ComponentContext context){
    session = repository.loginAdministrative(null);
    om = session.getWorkspace().getObservationManager();
    String path = "/content/slingbucks/orders";
    om.addEventListener(this,
       Event.PROPERTY_CHANGED | Event.PROPERTY_ADDED,
       path...);
         }
```

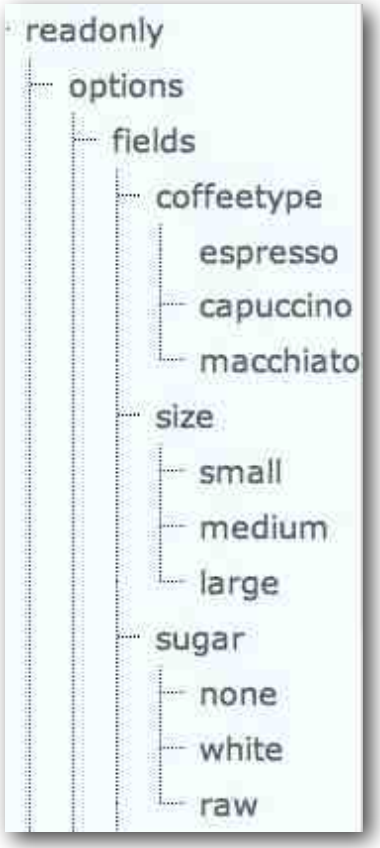
Slingbucks code: move confirmed orders 2/2

```
@Component
public class ConfirmedOrdersObserver
implements EventListener, Runnable {

... code from page 1

public void onEvent(EventIterator it) {
  while (it.hasNext()) {
    if(path.endsWith(«orderConfirmed») {
      ... note property change and
      ... if confirmed move node later
      ... using session.getWorkspace().move(srcPath, destPath);
    }
}
```

Slingbucks «code»: initial content for options

src/main/resources/SLING-CONTENT/content/slingbucks/readonly/options.json:

```
    "fields" : {
        "coffeetype" : {
            "jcr:title" : "Coffee type",
            "espresso" : {
                "jcr:title" : "Espresso",
                "jcr:description" : "The Italian job",
                "priceOffset" : 2.20
            },
            "capuccino" : {
                "jcr:title" : "Capuccino",
                "jcr:description" : "The one with cream on top",
                "priceOffset" : 3.40
            }
        },
        "size" : {
            "jcr:title" : "Size",
            "small" : {
                "jcr:title" : "Small",
                "jcr:description" : "1dl",
                "priceFactor" : 1
            },
            "large" : {
                "jcr:title" : "Large",
                "jcr:description" : "5dl",
                "priceFactor" : 2
            }
        },
```



Loaded by Sling as nodes/properties when OSGi bundle is loaded

Slingbucks code: content -> options form

src/main/resources/SLING-CONTENT/apps/slingbucks/options/options.esp:

```
<%
// Make sure current node has a "fields" subnode,
// and visit it (duck typing content!)
if(currentNode["fields"]) {
  var fields = currentNode["fields"];
  for(i in fields) {
    var f = fields[i];

    // If field has a jcr:title property, we can use it
    if(f["jcr:title"]) {
      %>
      // Generate HTML <select> for our field
      <select name="<%= fieldName %>">
        <options>
          <%
            for(j in f) {
              var opt = f[j];
              if(opt["jcr:title"]) {
                %>
                <option value="<%= j %>"><%= opt["jcr:title"] %></option>
                <%
              }
            }
          %>
...
// Set appropriate resource type on created coffee order
// and let Sling set lastModified property
<input type="hidden" name="sling:resourceType" value="slingbucks/order"/>
<input type="hidden" id="lastModified" name="lastModified" value=""/>
```

Slingbucks LOC

Java code: 250

```
171 src/main/java/org/apache...ConfirmedOrdersObserver.java
 57 src/main/java/org/apache...HexNodeNameGenerator.java
 28 src/main/java/org/apache...SlingbucksConstants.java
```

HTML representation scripts: 250

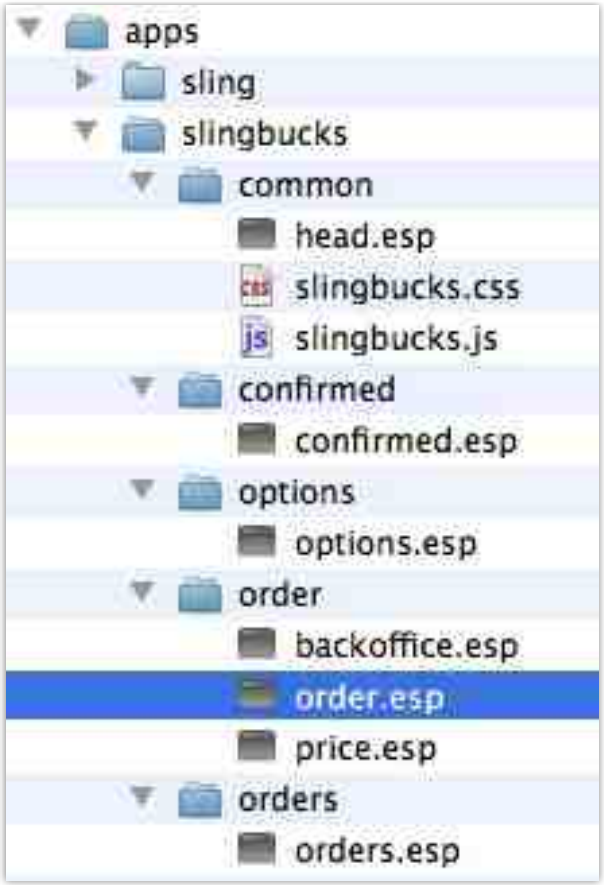
```
  2 /apps/slingbucks/common/head.esp
 15 /apps/slingbucks/confirmed/confirmed.esp
 47 /apps/slingbucks/options/options.esp
 25 /apps/slingbucks/order/backoffice.esp
 58 /apps/slingbucks/order/order.esp
 58 /apps/slingbucks/order/price.esp
 32 /apps/slingbucks/orders/orders.esp
```

Initial content: 85

```
 73 /content/slingbucks/readonly/options.json
  6 /content/slingbucks/private.json
  6 /content/slingbucks/public.json
```

Style etc: 43

```
 38 /apps/slingbucks/common/slingbucks.css
  5 /apps/slingbucks/common/slingbucks.js
```



Content-driven app: new field

```
$ cat /tmp/logo.json
{
  "jcr:title": "Cup Logo",
  "slingbucks": {
    "priceFactor": 1,
    "jcr:title": "Slingbucks",
  },
  "apache": {
    "priceFactor": 1.5,
    "jcr:title": "Apache Software Foundation",
  },
  "swissflag": {
    "priceFactor": 4.5,
    "jcr:title": "Swiss flag",
  }
}
```


```
curl -F:operation=import -F:contentType=json -F:contentFile=@/
tmp/logo.json http://admin:admin@127.0.0.1:8080/content/
slingbucks/readonly/options/fields/logo
```

Slingbucks next steps

Security:
Setup ACL on /public, /readonly, /private.
All done.

Scalability:
Built-in. No HTTP sessions. Cache-friendly.
RESTful.

End-to-end testing:
Easy using HTTP/JSON and HTTP/HTML scenarios.

That was not too hard, was it?

Code online soon,
stay tuned to twitter @bdelacretaz
or Sling mailing lists

RESTful apps with Sling

Built-in RESTful content creation/ editing

Easy to plugin scripts, (OSGi-based) servlets and extensions.

URLs map to resources, not commands

RESTful Web

SLING RESTFUL WEB APPS

http://sling.apache.org
http://dev.day.com
http://grep.codeconsult.ch
@bdelacretaz on Twitter

Sling Restful Web Apps

ApacheCon
North America 2010

Adobe