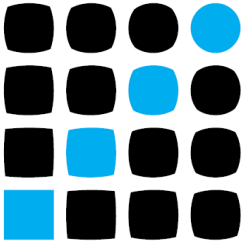


adaptTo()

APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 22-24 SEPTEMBER 2014

Sling Rookie Session

Sebastian Schlick, pro!vision GmbH

About the Speaker

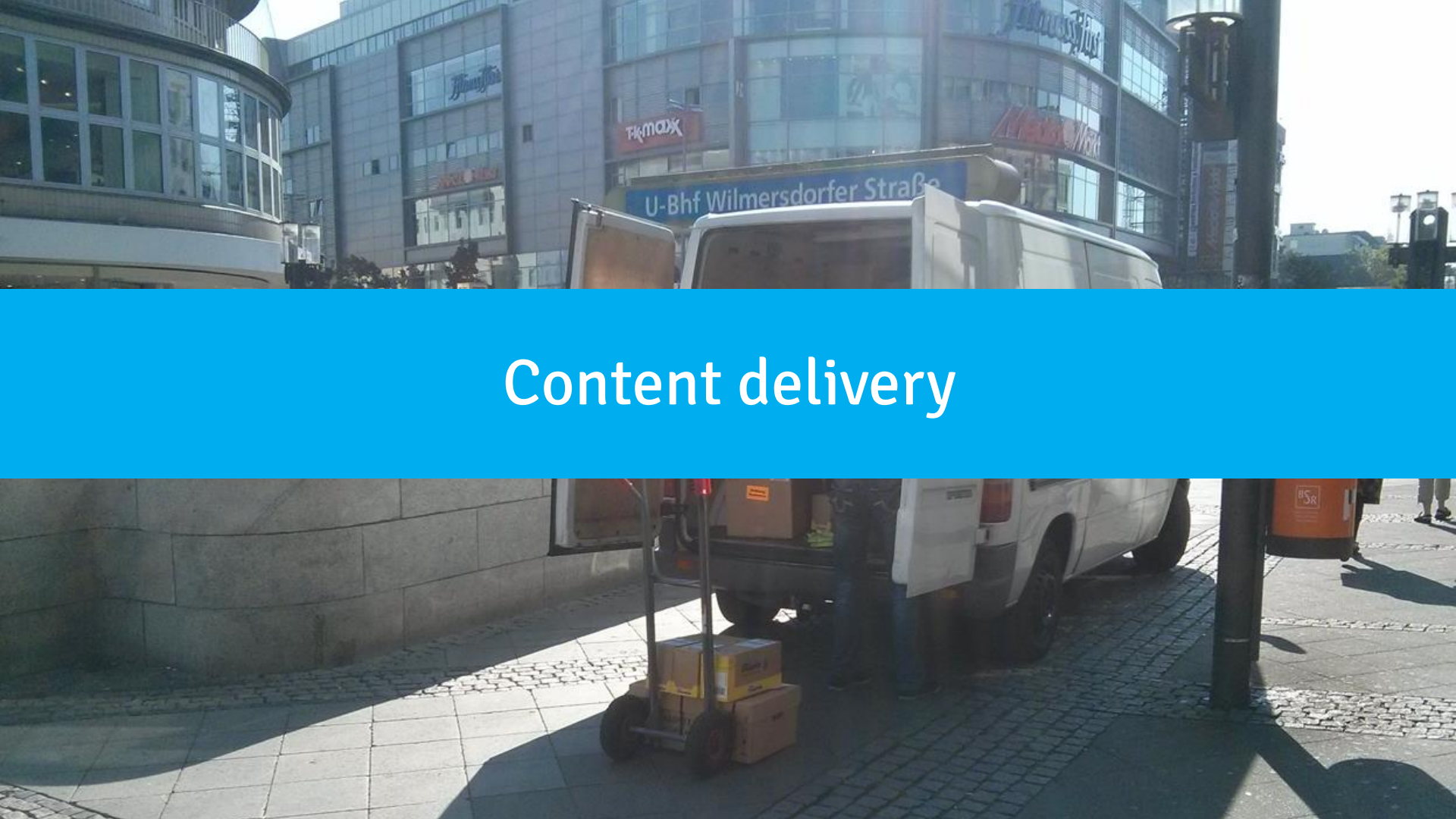
- CQ5/AEM6 Developer
- Apache Sling User
- Lead dev pro!vision GmbH

PRO!VISION
SOFTWARE CRAFTSMANSHIP

http://www.pro-vision.de

Overview

- Content delivery
- Sling Basics
- Sling for real
- Sling advanced

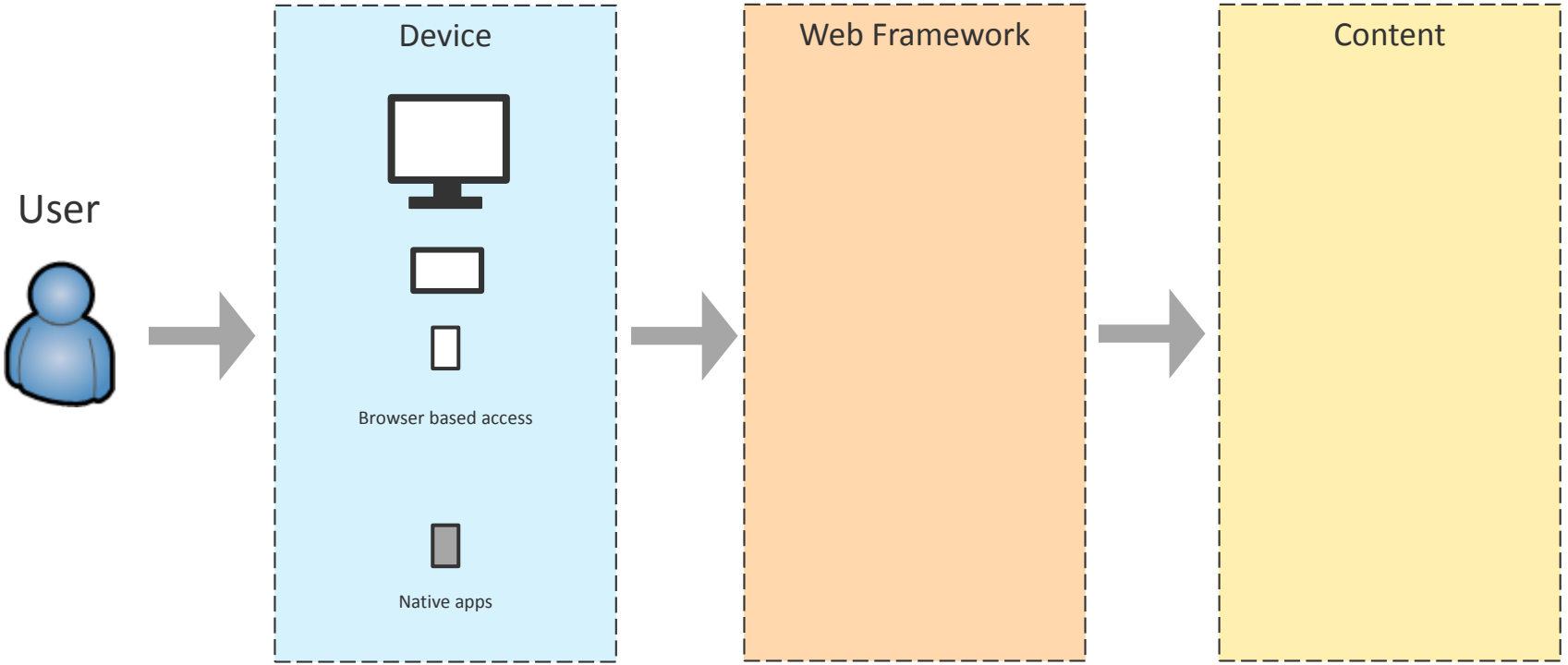


Content delivery

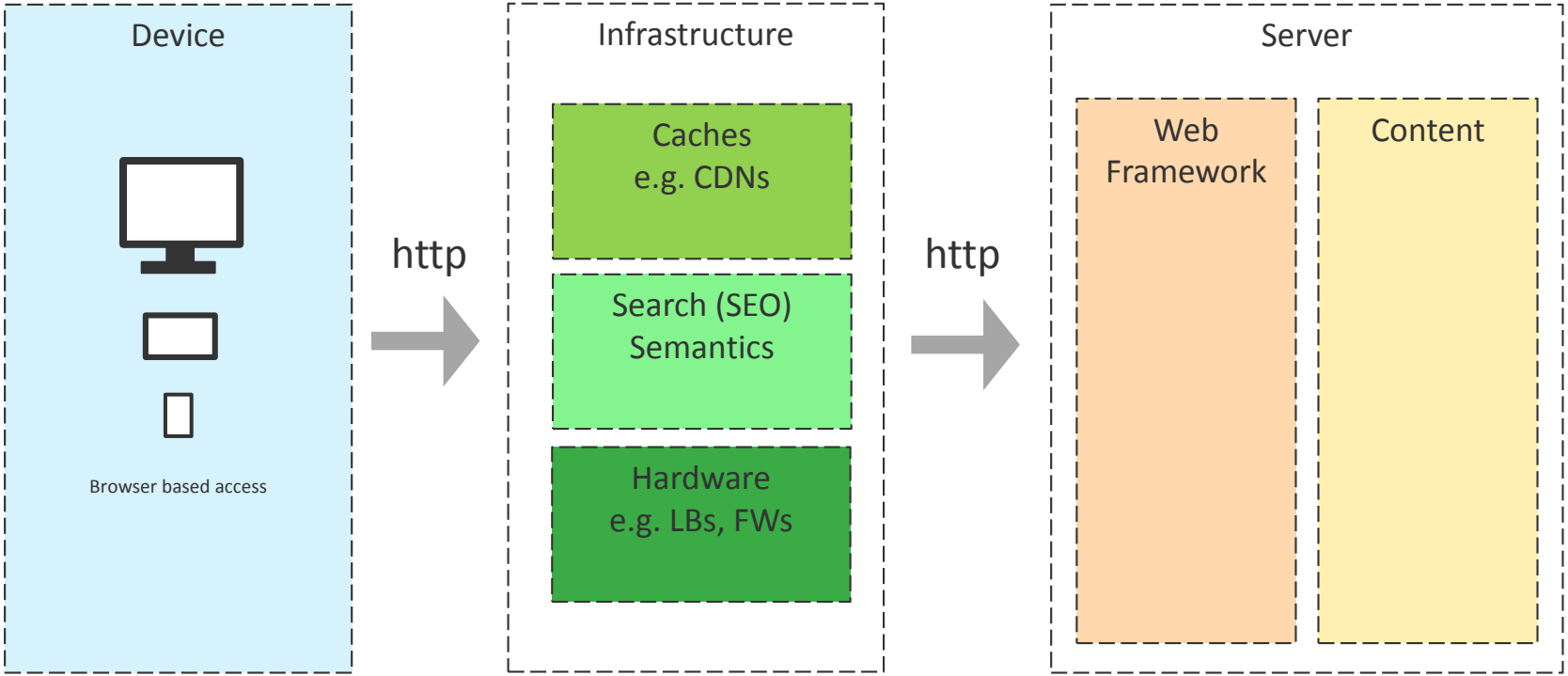
Content delivery

- Deliver content to the user
  - Different types of content
  - Everything is content (even code)
- Content representation
- Access control
- Eventing, job execution, et cetera

Content delivery



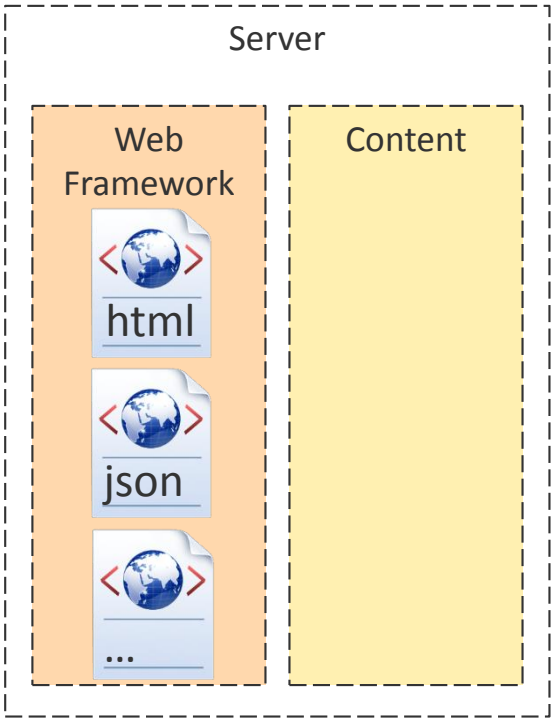
Content delivery



Content delivery

http

http



REST

- Representational State Transfer
  - Uniform Interface
  - Stateless Interactions
  - Cacheable
  - Client-Server separation
  - Layered System

(coined by Roy Fielding)

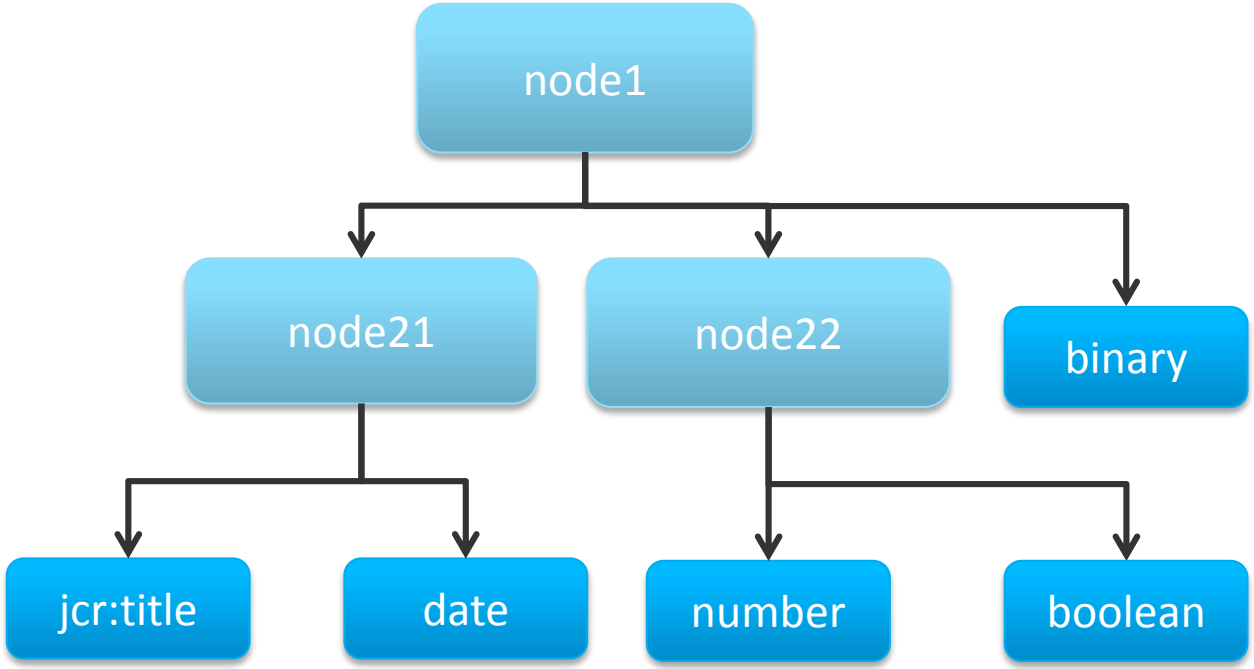
Fyi:
http://roca-style.org/

REST (how to achieve)

- Start early (yes, at persistance layer)
  - Relation vs. Hierarchy
  - Typisation
  - Versioning
- Keep transformations simple

Hierarchical structure of resources

0. Node
1. Node
2. Property

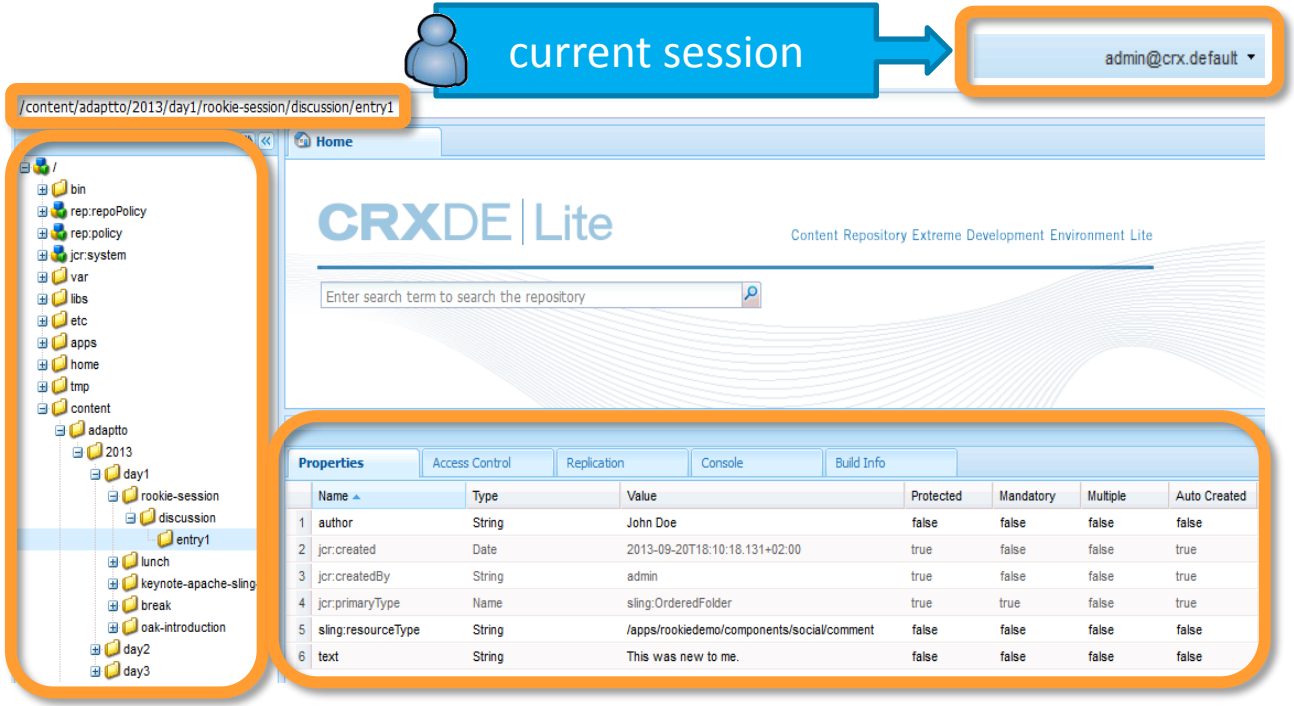


Dive into JCR with CRX DE

path to current node

Workspace tree

current session



properties of current node

Content Repository (Example)

jcr:baseVersion	Reference	f4f181bf-3be6-455d-a4e5-dbe5bc3b28ed
jcr:created	Date	2013-08-23T13:36:04.796+02:00
jcr:createdBy	String	admin
jcr:isCheckedOut	Boolean	true
jcr:mixinTypes	Name[]	mix:versionable, cq:ReplicationStatus
jcr:predecessors	Reference[]	f4f181bf-3be6-455d-a4e5-dbe5bc3b28ed
jcr:primaryType	Name	nt:unstructured
jcr:title	String	de
jcr:uuid	String	18ce6c92-c947-4f75-b8b4-a13246017981
jcr:versionHistory	Reference	2a514f97-bd8f-4131-b1c7-0e6ed8922f05
noPageDisclaimers	Boolean	false
pageDisclaimers	String[]	Kraftstoffverbrauch, Beispiel-Disclaimer
pageTitle	String	Das WeltAuto.
sling:resourceType	String	/apps/vwd4_dwa/components/editorial/page/dwa_e0_homepage
trackingOnLoad	String	activated

Summing up

- JCR is a good foundation for RESTful data access
- REST is good for content delivery
  - Not all Frameworks achieve this OOTB
- Infrastructure likes REST too
- In the net, infrastructure matters

Sling basics

SLING per cURL: POST

- From: http://sling.apache.org/documentation/getting-started/discover-sling-in-15-minutes.html

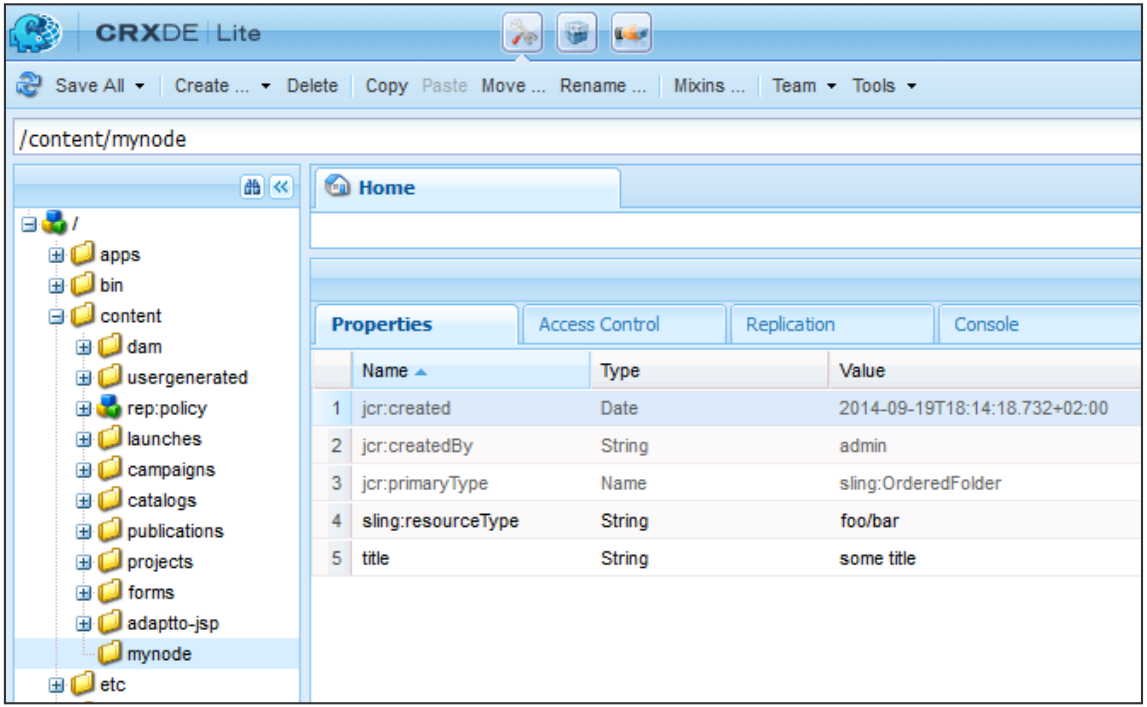
HTTP POST: Create a content node (nodes are a JCR concept, a unit of storage) with cURL

curl is a tool to transfer data from or to a server

- -F, --form <name=content>
- (HTTP) This lets curl emulate a filled-in form in which a user has pressed the submit button. This causes curl to POST data using the Content-Type multipart/form-data according to RFC 2388.

```
$ curl    -u admin:admin
          -F"sling:resourceType=foo/bar"
          -F"title=some title" http://localhost:4502/content/mynode
```

SLING per cURL: POST



SLING per: cURL GET

- http://sling.apache.org/documentation/getting-started/discover-sling-in-15-minutes.html

HTTP GET: The resulting node can be seen also with cURL:

```
$ curl -u admin:admin http://localhost:4502/content/mynode.json
```

```
{"title":"some title","sling:resourceType":"foo/bar","jcr:primaryType":"nt:unstructured"}
```

SLING per cURL: PUT a render script

- http://sling.apache.org/documentation/getting-started/discover-sling-in-15-minutes.html

HTTP POST: Create two sling folders with cURL:

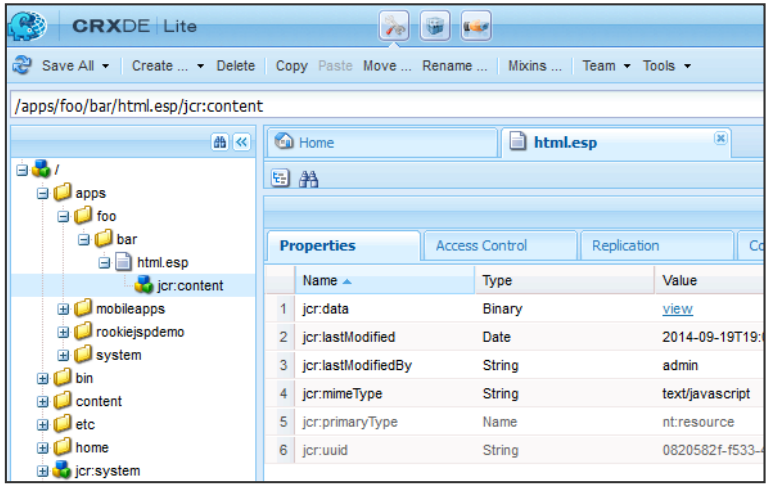
```
curl -u admin:admin -F"jcr:primaryType=sling:Folder" http://localhost:4502/apps/foo
curl -u admin:admin -F"jcr:primaryType=sling:Folder" http://localhost:4502/apps/foo/bar
```

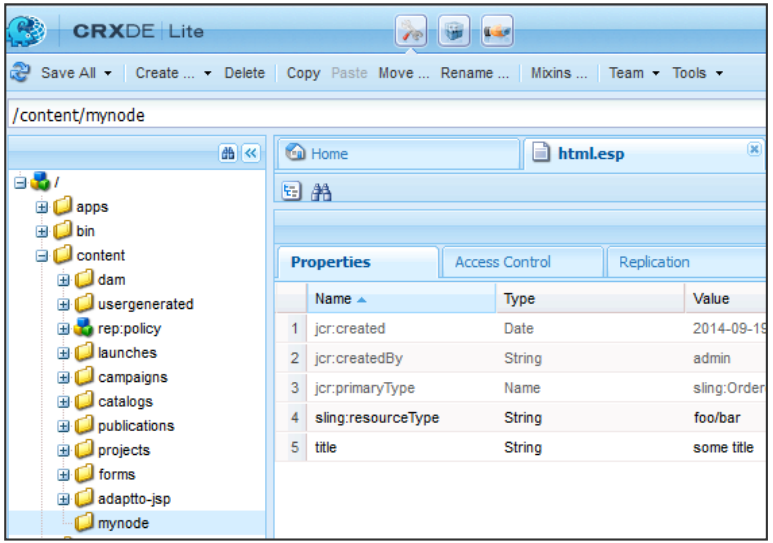
HTTP PUT: upload the script

```
curl -u admin:admin -T html.esp http://localhost:4502/apps/foo/bar/html.esp
```

Any **http get** with protocol **html** pointing to a **node** with `sling:resourceType` of `foo/bar` is now rendered with the **html.esp**

SLING per cURL: PUT a render script






```
<html>
 <body>
  <h1><%= currentNode.title %></h1>
 </body>
</html>
```

Apache Sling

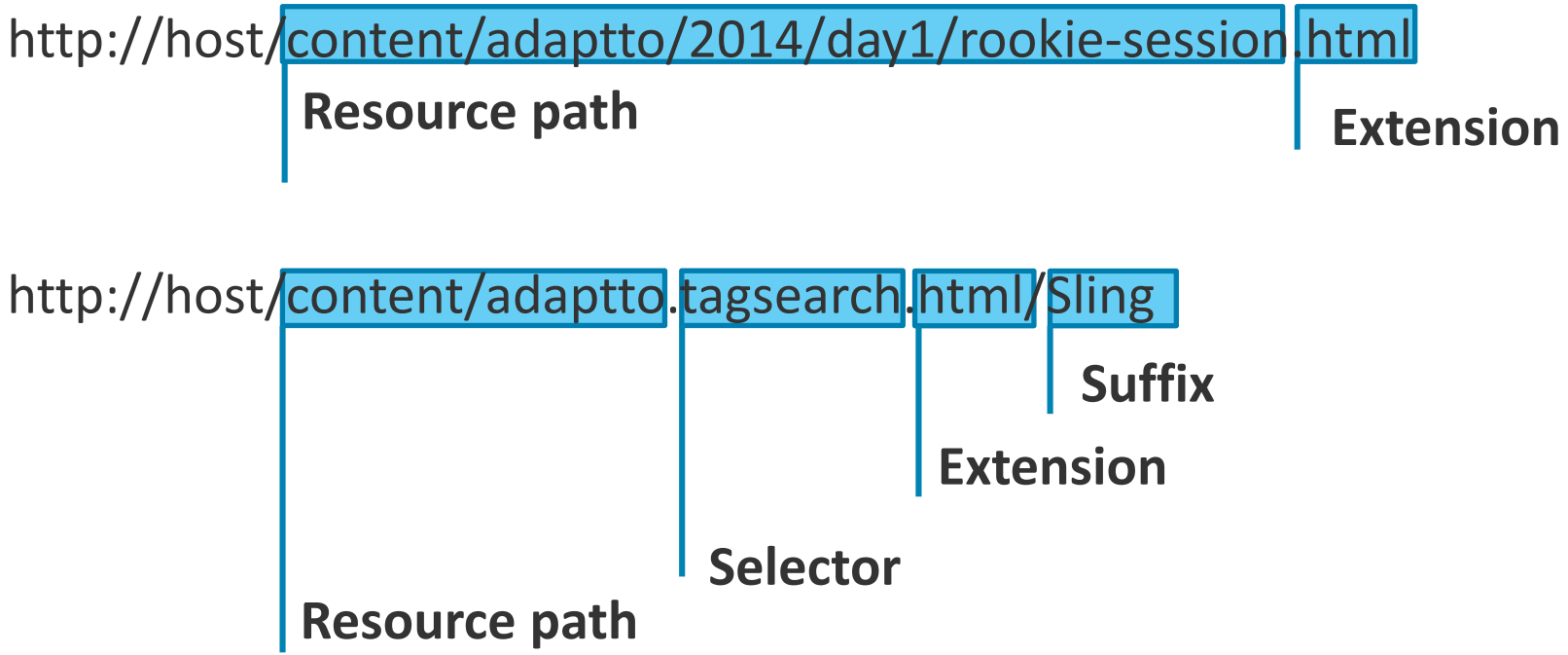
- REST based Framework on top of JCR
- Apache top level since 2009
- OSGI driven
- JVM based (JSP, Scala, …)
- Renders JCR Nodes using Scripts in JCR
- Maps URLs to content representations

Resource Hierarchy

- Node in JCR has properties

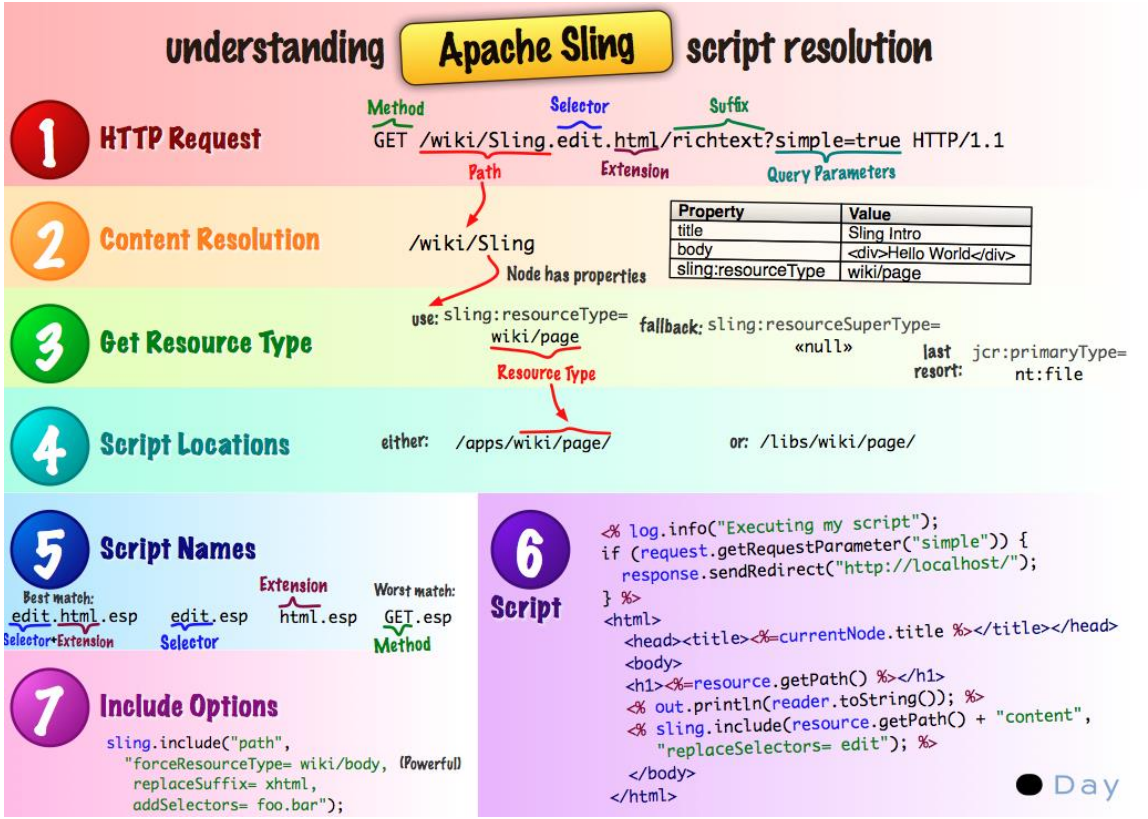
- Node may be a resource if it has the property
  - sling:resourceType
- Inheritance.
  - sling:resourceSuperType

URL decomposition

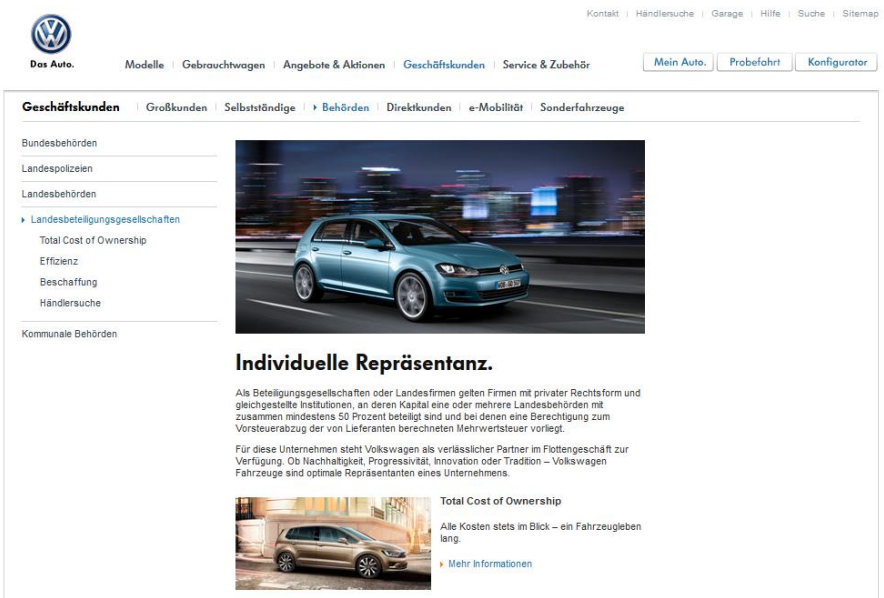


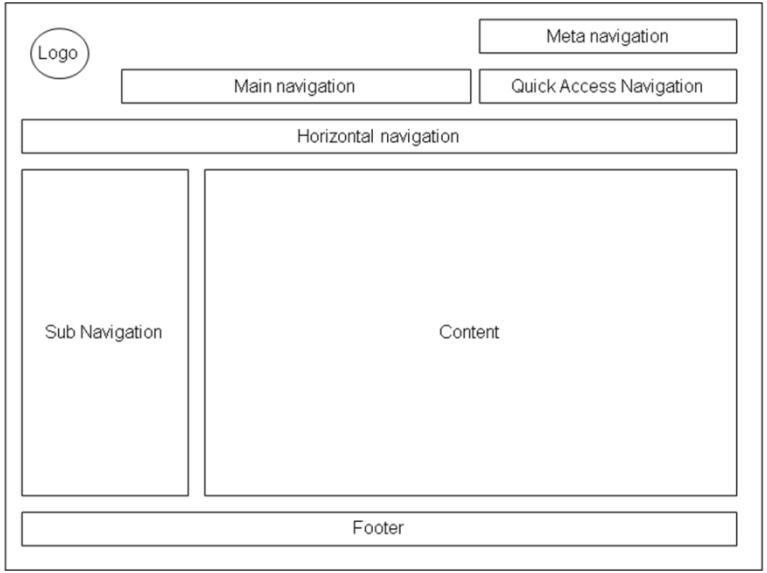
http://sling.apache.org/documentation/the-sling-engine/url-decomposition.html

Sling cheat sheet



Example: modularization of markup





Summing up

- Sling is http based and as RESTful as it gets
- Sling can GET and POST (CRUD complete)
- Uses http headers and protocols
- Maps URLs to Scripts in JCR
- Everything is content with Sling
- Sling includes for markup modularization

The sample app

Sling Rookie Session – Demo application

- Available at github

https://github.com/adaptto-conf/2014-sling-rookie-session

- JSP
- Sighly

JSP Example: Simple HTML view

Resource Type: **/apps/rookiedemo/components/talk**

JSP Script in JCR: **/apps/rookiedemo/components/talk/html.jsp**

Script type

Extension resolution mapping

```
<!doctype html>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sling" uri="http://sling.apache.org/taglibs/sling"%>
<sling:defineObjects/>
<sling:adaptTo var="props" adaptable="${resource}"
adaptTo="org.apache.sling.api.resource.ValueMap"/>
<html>
  <body>

    <%-- Output talk properties --%>
    <h1><c:out value="${props['jcr:title']}"/></h1>
    <p><c:out value="${props['jcr:description']}"/></p>
    <p><em><c:out value="${props.speaker}"/>, ${props.durationMin} min</em></p>
  </body>
</html>
```

JSP Example: vCalendar view

Resource Type: **/apps/rookiedemo/components/talk**

JSP Script in JCR: **/apps/rookiedemo/components/talk/vcs.jsp**

Script type

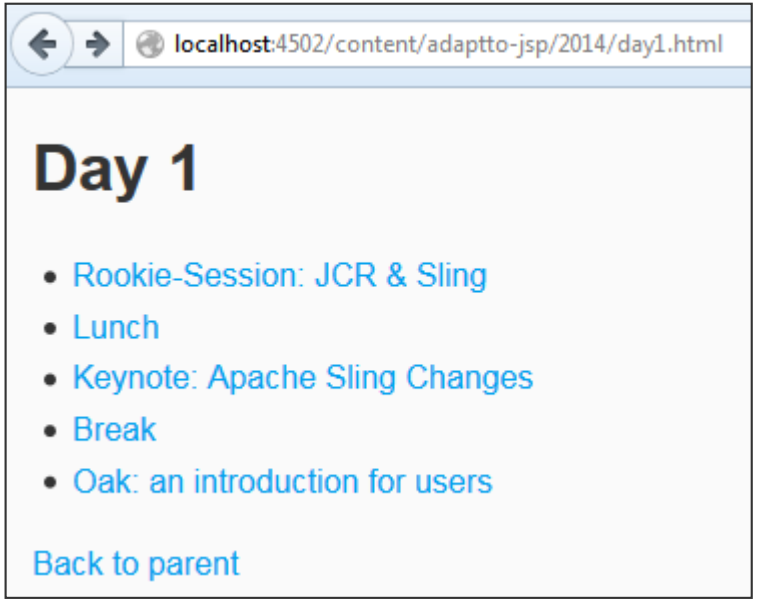
Extension resolution mapping

```
<%@page contentType="text/calendar; charset=UTF-8" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sling" uri="http://sling.apache.org/taglibs/sling"%>
<sling:defineObjects/>
<sling:adaptTo var="props" adaptable="${resource}" adaptTo="org.apache.sling.api.resource.ValueMap"/>

<%-- Output talk details as vCalender to import in mail client --%>
BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
DTSTART:${props.startDate}
DTEND:${props.endDate}
DESCRIPTION;ENCODING=QUOTED-PRINTABLE:<c:out value="${props['jcr:description']}"/>
SUMMARY:<c:out value="${props['jcr:title']}"/>
PRIORITY:3
END:VEVENT
END:VCALENDAR
```

JSP Example: Iterate over resources

JSP Script in JCR: **/apps/rookiedemo/components/common/childlist.jsp**
(included in other views via sling:include)

```
<ul>
  <%-- Iterate over all child resources from current resource --%>
  <sling:listChildren var="children" resource="${resource}"/>
  <c:forEach var="child" items="${children}">
    <sling:adaptTo var="props" adaptable="${child}"
adaptTo="org.apache.sling.api.resource.ValueMap"/>
    <li>
      <a href="${child.path}.html"><c:out
value="${props['jcr:title']}"/></a>
    </li>
  </c:forEach>

</ul>
```



Sling script inclusion examples

Example for **sling:call**

```
<%-- Include html_head script inherited from super component "common" --%>
<sling:call script="html_head.jsp"/>
```

Example for **sling:include**: replace selectors to force rendering with different script

```
<%-- Include childlist via selector view inherited from super component "common" --%>
<sling:include replaceSelectors="childlist"/>
```

Example for **sling:include**: render current resource with different resource type

```
<%-- Integrate java-based sling component via it's resource type to render previous/next links --%>
<sling:include resourceType="/apps/rookiedemo/components/resourceSiblingNavigator"/>
```

Example for **sling:include**: iterate over children and render each child with it's own resource type

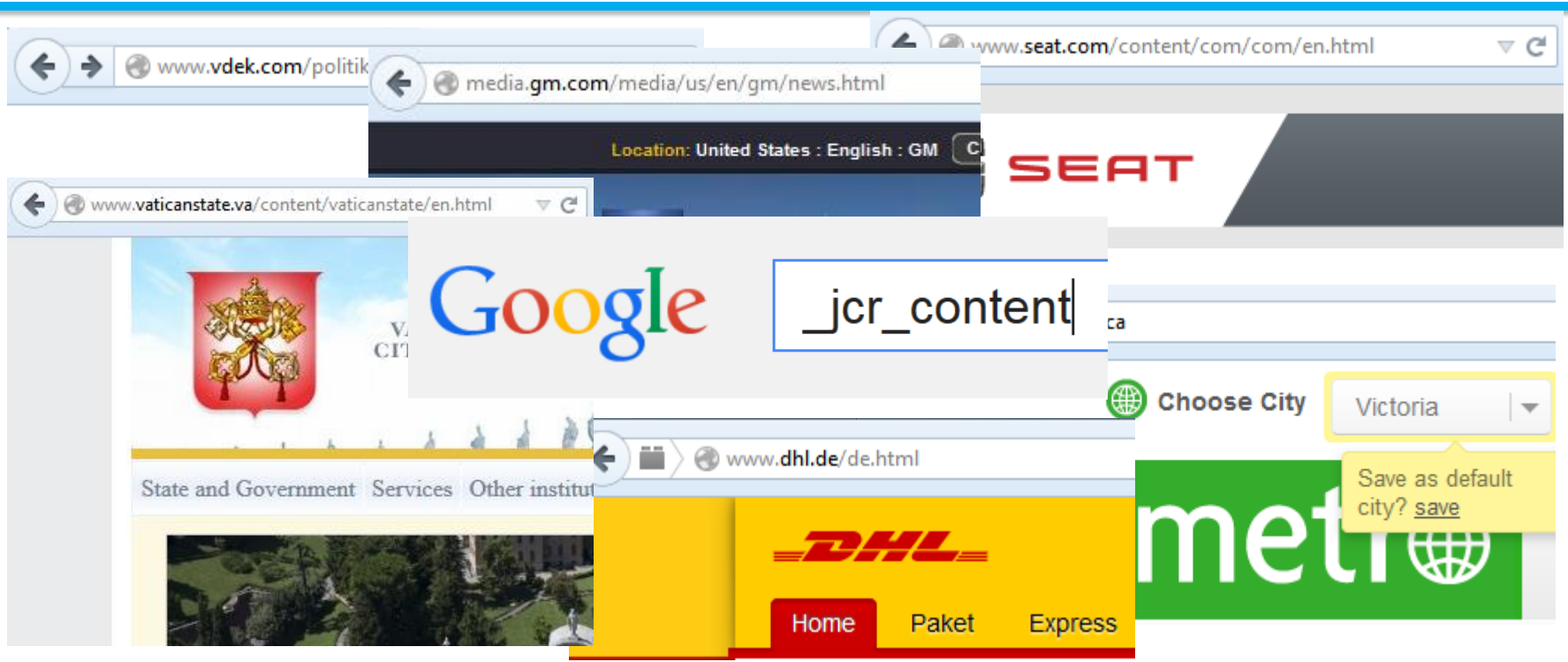
```
<%-- Render all existing comments --%>
<sling:getResource var="discussionResource" path="${resource.path}/discussion"/>
<sling:listChildren var="children" resource="${discussionResource}"/>
<c:forEach var="child" items="${children}">
  <sling:include resource="${child}"/>
</c:forEach>
```

Summing up

- Sample project on github
- Usecases:
  - Render as html or vcf, selected by URL extension
  - Iterate over resources
  - Scripts may include other scripts

Sling Real World Examples

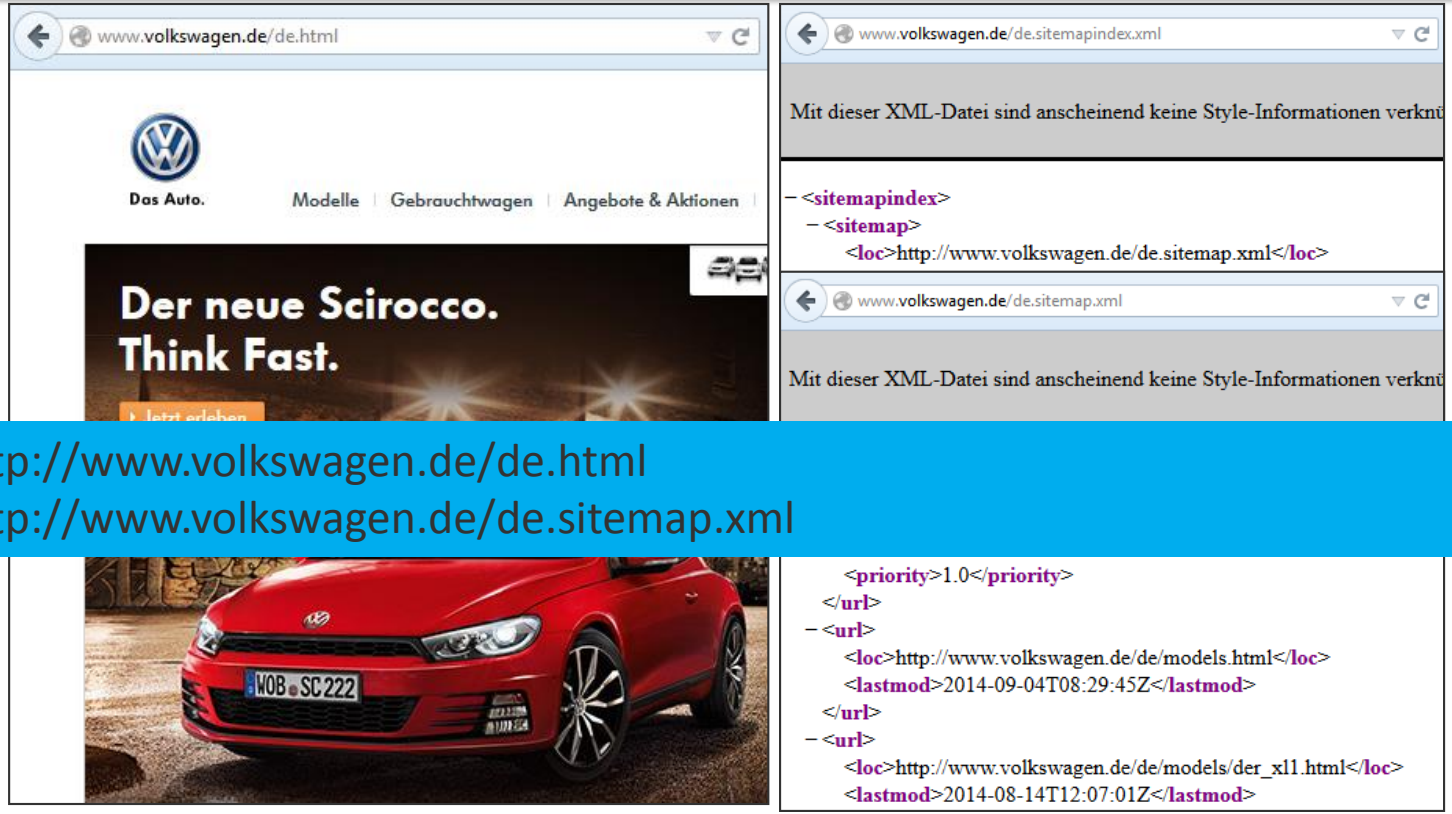
Example: JCR sites found with google



Example: HTML and RSS view

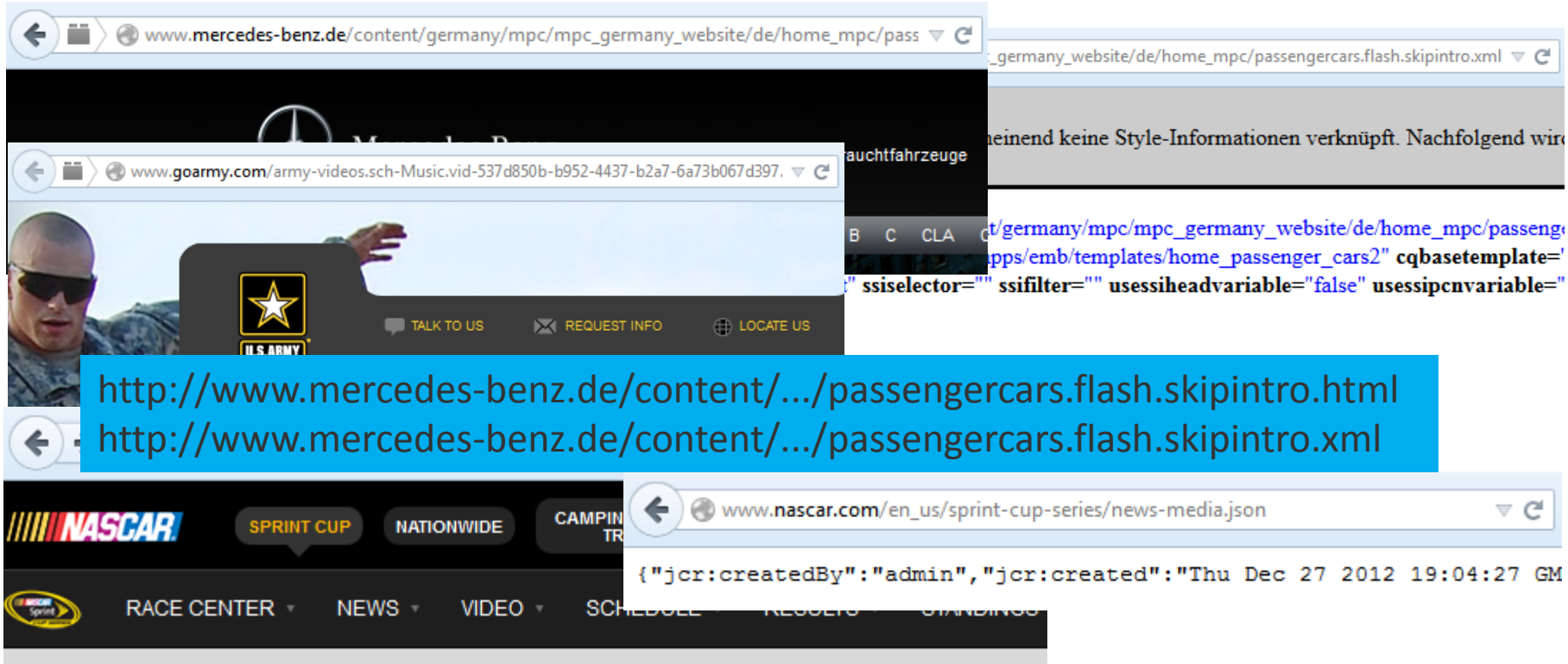
http://www.rwjf.org/en/blogs/culture-of-health.html
http://www.rwjf.org/en/blogs/culture-of-health/_jcr_content.rssfeed

Example: Sling selector used for sitemap



http://www.volkswagen.de/de.html
http://www.volkswagen.de/de.sitemap.xml

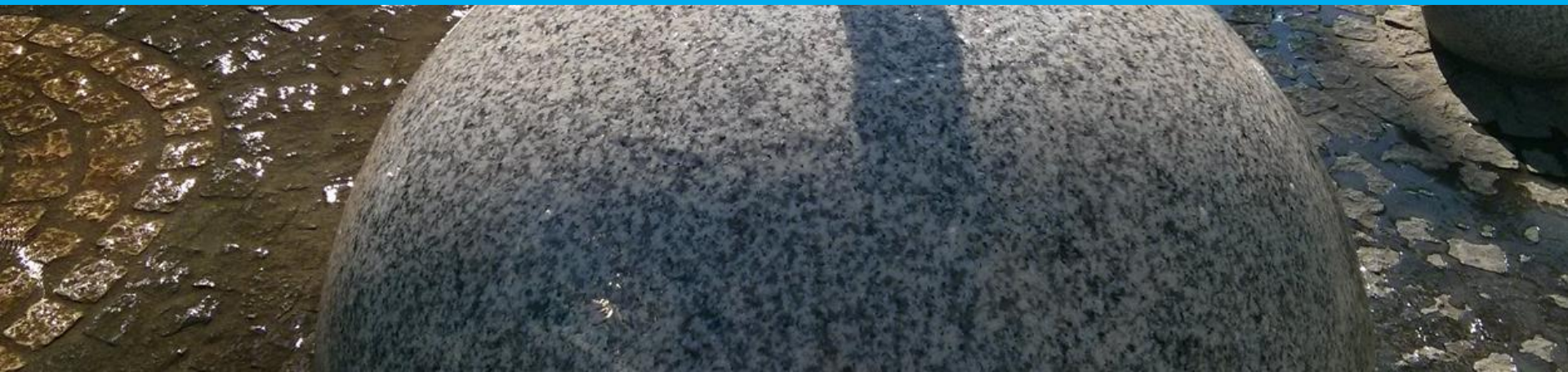
Example: Content exposure



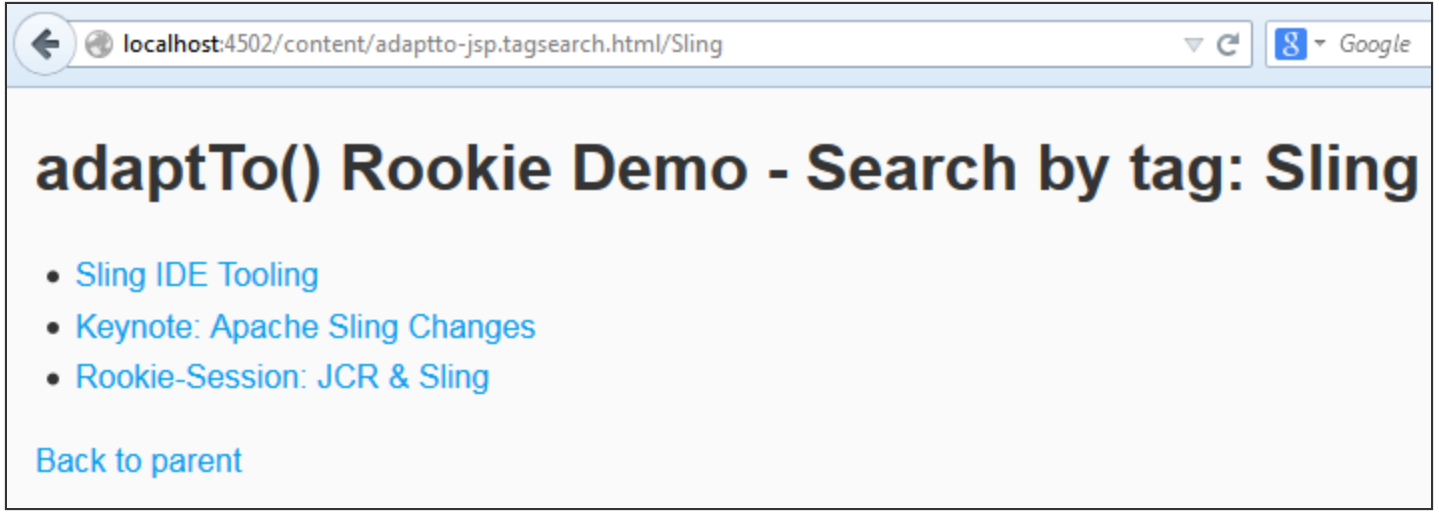
Summing up

- JCR is well established
- Sling delivers JCR content very easily
- Sling delivers content in many forms OOTB
  - It is a good thing
  - Just be aware of it

More sling features



Sling model


localhost:4502/content/adaptto-jsp.tagsearch.html/Sling
Google
adaptTo() Rookie Demo - Search by tag: Sling
Sling IDE Tooling
Keynote: Apache Sling Changes
Rookie-Session: JCR & Sling
Back to parent

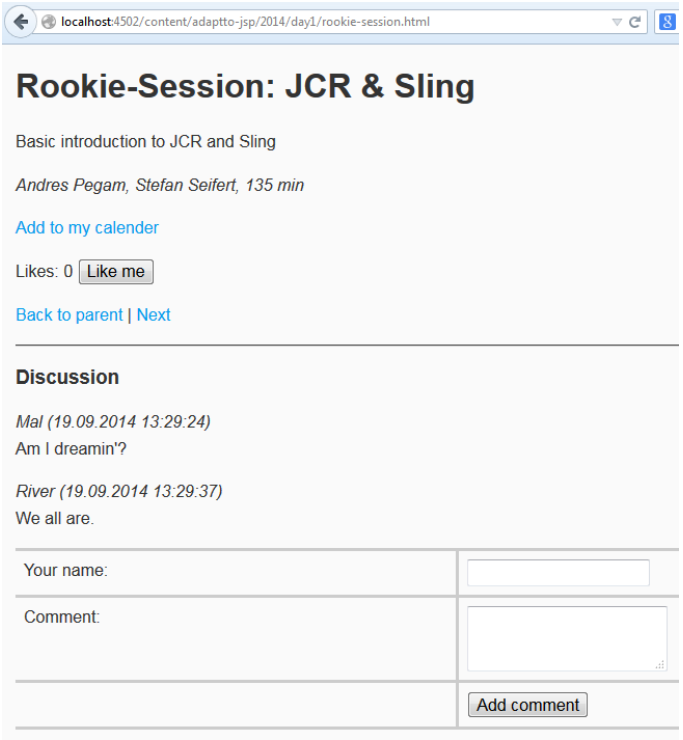
Sling model

```java
/**
 * Controller model that implements a search for all talks with the
tag name given as suffix.
 */
@Model(adaptables=SlingHttpServletRequest.class)
public class TagSearchController {

  private final String tag;
  private final List<Resource> result;

public TagSearchController(SlingHttpServletRequest request) {
    Resource resource = request.getResource();
    ResourceResolver resolver = request.getResourceResolver();

    // get tag name to search for form suffix
    String suffix = request.getRequestPathInfo().getSuffix();
    this.tag = StringUtils.substringAfter(suffix, "/");

    // execute JCR query via Sling API
    String xpathQuery = "/jcr:root" + resource.getPath() +
"//*[tags='" + this.tag + "']";
    this.result =
IteratorUtils.toList(resolver.findResources(xpathQuery, "xpath"));
  }


}
```

```
<%-- Search all talks with the given tag name using a Sling Model --%>
<sling:adaptTo var="search" adaptable="${slingRequest}"
adaptTo="org.adaptto.rookie.jspdemo.models.TagSearchController"/>

<html>

  <sling:call script="html_head.jsp"/>

  <body>

    <h1>adaptTo() Rookie Demo - Search by tag: <c:out
value="${search.tag}"/></h1>

    <%-- Display search results --%>
    <ul>
      <c:forEach var="child" items="${search.result}">
        <sling:adaptTo var="props" adaptable="${child}"
adaptTo="org.apache.sling.api.resource.ValueMap"/>
        <li>
          <a href="${child.path}.html"><c:out
value="${props['jcr:title']}"/></a>
        </li>
      </c:forEach>
    </ul>

    <p><a href="${resource.path}.html">Back to parent</a></p>
```

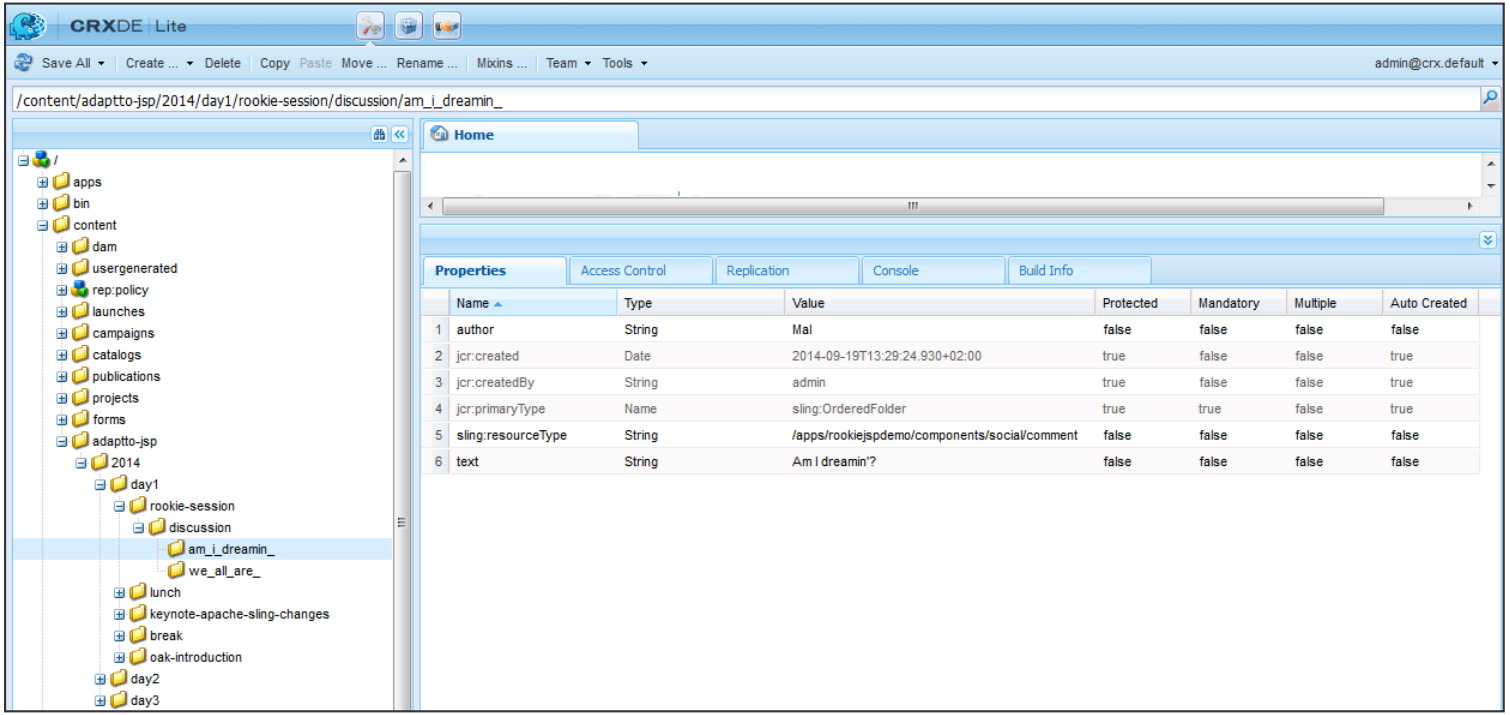
POST a comment

(as seen in slide „SLING per cURL“)




```
 <%-- Post to "*" which means create a new resource with unique name --%>
  <form action="${resource.path}/discussion/*" method="POST"
enctype="multipart/form-data">

    <%-- Define resource type for new node --%>
    <input type="hidden" name="sling:resourceType"
value="/apps/rookiejspdemo/components/social/comment"/>

    <%-- Ensure proper charset encoding --%>
    <input type="hidden" name="_charset_" value="UTF-8"/>

    <%-- Redirect to main view after writing content --%>
    <input type="hidden" name=":redirect" value="${resource.path}.html"/>

    <%-- Post to properties "author" and "text" in repository --%>
    <table>
      <tr>
        <td>Your name:</td>
        <td><input type="text" name="author"/></td>
      </tr>
      <tr>
        <td>Comment:</td>
        <td><textarea name="text"/></textarea></td>
      </tr>
      <tr>
        <td></td>
        <td><input type="submit" value="Add comment"/></td>
      </tr>
    </table>

  </form>
```

POST a comment



adaptTo()

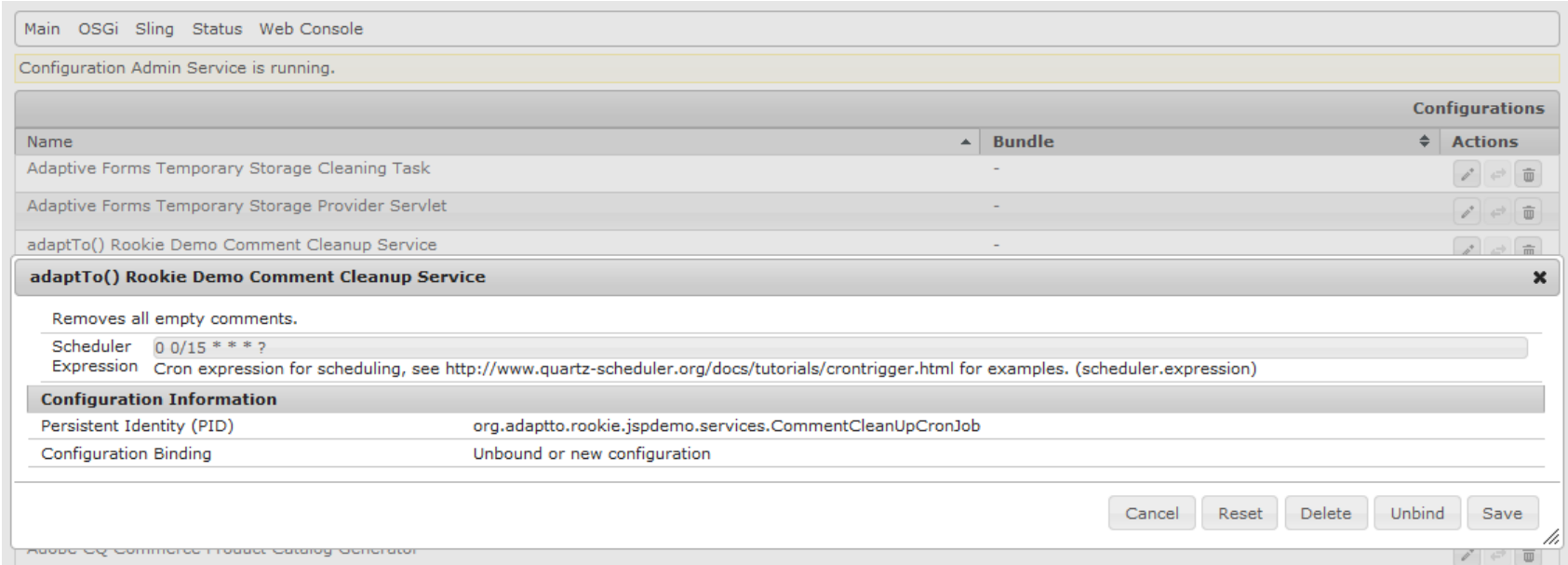
```java
/**
 * Servlet example comment for social comment entry
 */
@SlingServlet(resourceTypes="/apps/rookiejspdemo/components/social/comment")
public class DiscussionComment extends SlingSafeMethodsServlet {

  @Override
  protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)
throws ServletException, IOException {
    Writer out = response.getWriter();

    // read comment via Sling Model
    Comment comment = request.getResource().adaptTo(Comment.class);

    // output comment as HTML
    out.write("<p>");
    out.write("<em>" + escapeHtml(comment.getAuthor())
        + " (" + DateFormat.getDateTimeInstance().format(comment.getCreated()) +
")</em><br/>");
    out.write(escapeHtml(comment.getText()));
    out.write("</p>");

  }

}
```

```java
/**
 * Model mapping comment resource properties to getter
methods.
 */
@Model(adaptables=Resource.class)
public class Comment {

  @Inject
  @Optional
  private String author;

  @Inject
  @Named("jcr:created")
  private Date created;

  @Inject
  @Optional
  private String text;

  public boolean isEmpty() {
    return StringUtils.isEmpty(getText());
  }

}
```

OSGi ready

```
/**
 * Background job to automatically remove empty comments.
 */
@Component(immediate = true, metatype = true, label = "adaptTo() Rookie Demo Comment Cleanup Service",
    description = "Removes all empty comments.")
@Service(value = Runnable.class)
public class CommentCleanUpCronJob implements Runnable {

  @Property(value = "0 0/15 * * * ?", // run every 15 minutes
  label = "Scheduler Expression",
      description = "Cron expression for scheduling, see http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html for examples.")
  private static final String PROPERTY_CRON_EXPRESSION = "scheduler.expression";

  @Reference
  private ResourceResolverFactory resourceResolverFactory;
```

OSGi ready


Main OSGi Sling Status Web Console
Configuration Admin Service is running.
Configurations
Name
Bundle
Actions
Adaptive Forms Temporary Storage Cleaning Task
-
Adaptive Forms Temporary Storage Provider Servlet
-
adaptTo() Rookie Demo Comment Cleanup Service
-
adaptTo() Rookie Demo Comment Cleanup Service
Removes all empty comments.
Scheduler Expression
0 0/15 * * * ?
Cron expression for scheduling, see http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html for examples. (scheduler.expression)
Configuration Information
Persistent Identity (PID)
org.adaptto.rookie.jspdemo.services.CommentCleanUpCronJob
Configuration Binding
Unbound or new configuration
Cancel
Reset
Delete
Unbind
Save

More examples in 2013 slides

- Sling Default JSON/XML Mapping
- JCR queries in Sling
- Custom POST, Sling CRUD
- …
- http://adapt.to/2013/en/schedule/01_rookiesession.html

Summing up

- Content delivery
- REST
- Java Content Repository
- Apache Sling
- Rookie Demo

The creation of Sling (video)

References

- **ROCA** Resource-oriented Client Architecture
  http://roca-style.org/
- **Representational State Transfer**
  http://en.wikipedia.org/wiki/Representational_state_transfer
- **Apache Jackrabbit JCR**
  http://jackrabbit.apache.org/jcr-api.html
- **Sling**
  http://sling.apache.org/documentation/getting-started.html
- **Sample app (JSP and Sighly Demo)**
  https://github.com/adaptto-conf/2014-sling-rookie-session
- **Last years rookie presentation**
  http://adapt.to/2013/en/schedule/01_rookiesession.html