APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 23-25 SEPTEMBER 2013

Rookie Session: JCR & Sling

Andres Pegam, Stefan Seifert

pro!vision GmbH

JCR

What is a JCR?

Content Repository API for Java

1.0

2.0

JCR

JCR History

- JCR 1.0
  - JSR-170
    http://www.day.com/specs/jcr/1.0/index.html

- JCR 2.0
  - JSR-283
    http://www.day.com/specs/jcr/2.0/index.html

What is a JCR?

Content Repository API for Java

1.0

2.0

JCR

Motto:

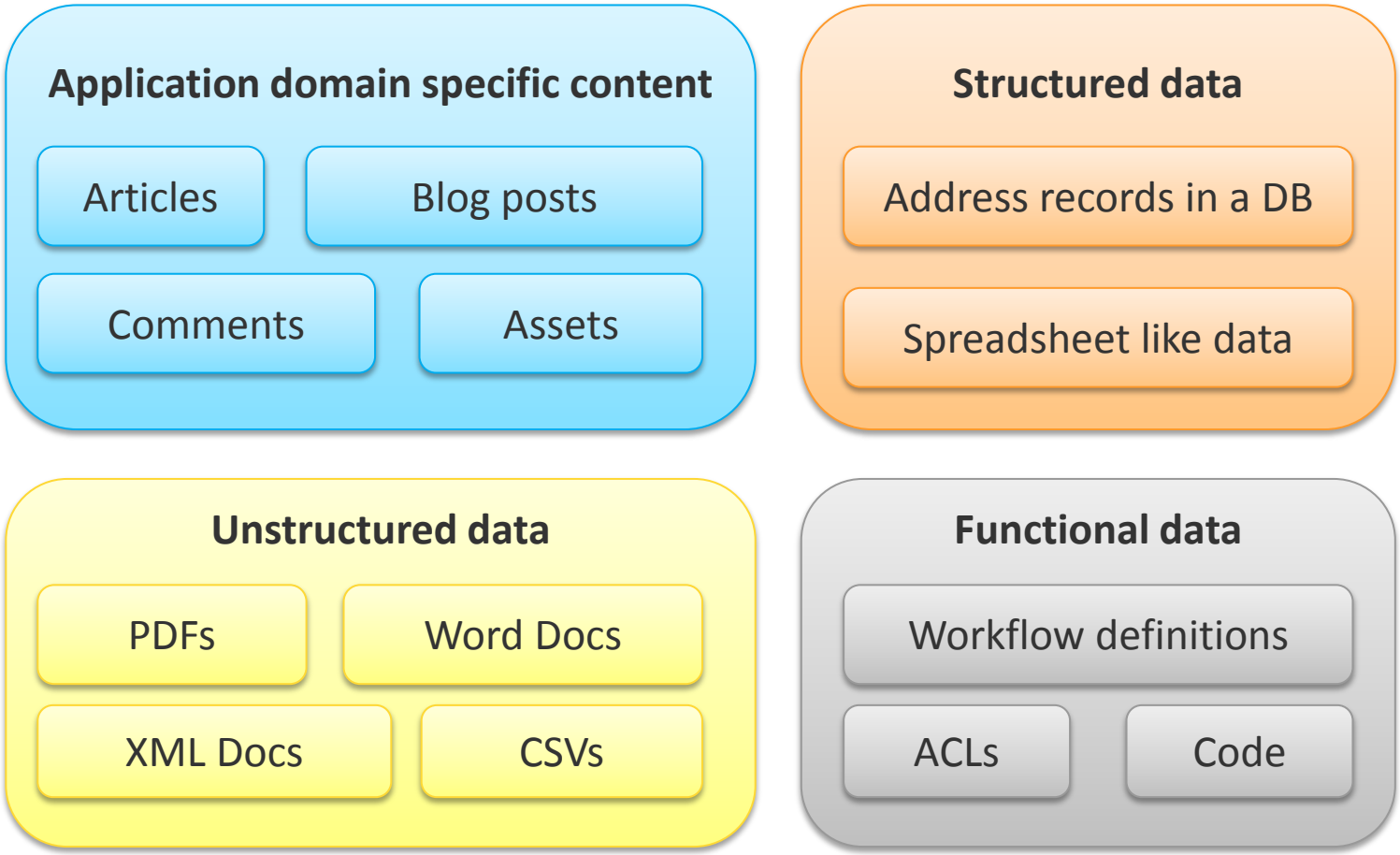
Everything Is Content

Goal:

A Unified interface to Content Repositories.


CONTENT REPOSITORY
FEATURES OF AN
RDBMS
Transactions, Query, Structure, Integrity
FEATURES OF A
FILESYSTEM
Binaries, Hierarchy, Locking, Access Control
+
ALL THE OTHER
GOOD STUFF
YOU ALWAYS WANTED
Unstructured, Versioning, Full-text,
Multi-Value, Sort-Order, Observation

What Is Content?



Comparison: JCR vs. RDBMS

	JCR	RDBMS
Structure	Unstructured, semi-unstructured, Structured	Structured
Navigation	Navigation API, traversal access, direct access, write access	Not supported
Access control	Record level	Table and column level
Version control	Supported	Not supported
Code complexity	Simple for navigation Complex for operations	Complex for navigation Simple for operations
Changeability	More agile Decoupled from the application	More rigid Coupled with the application

JCR vs. RDBMS

~~Better or Worse?~~

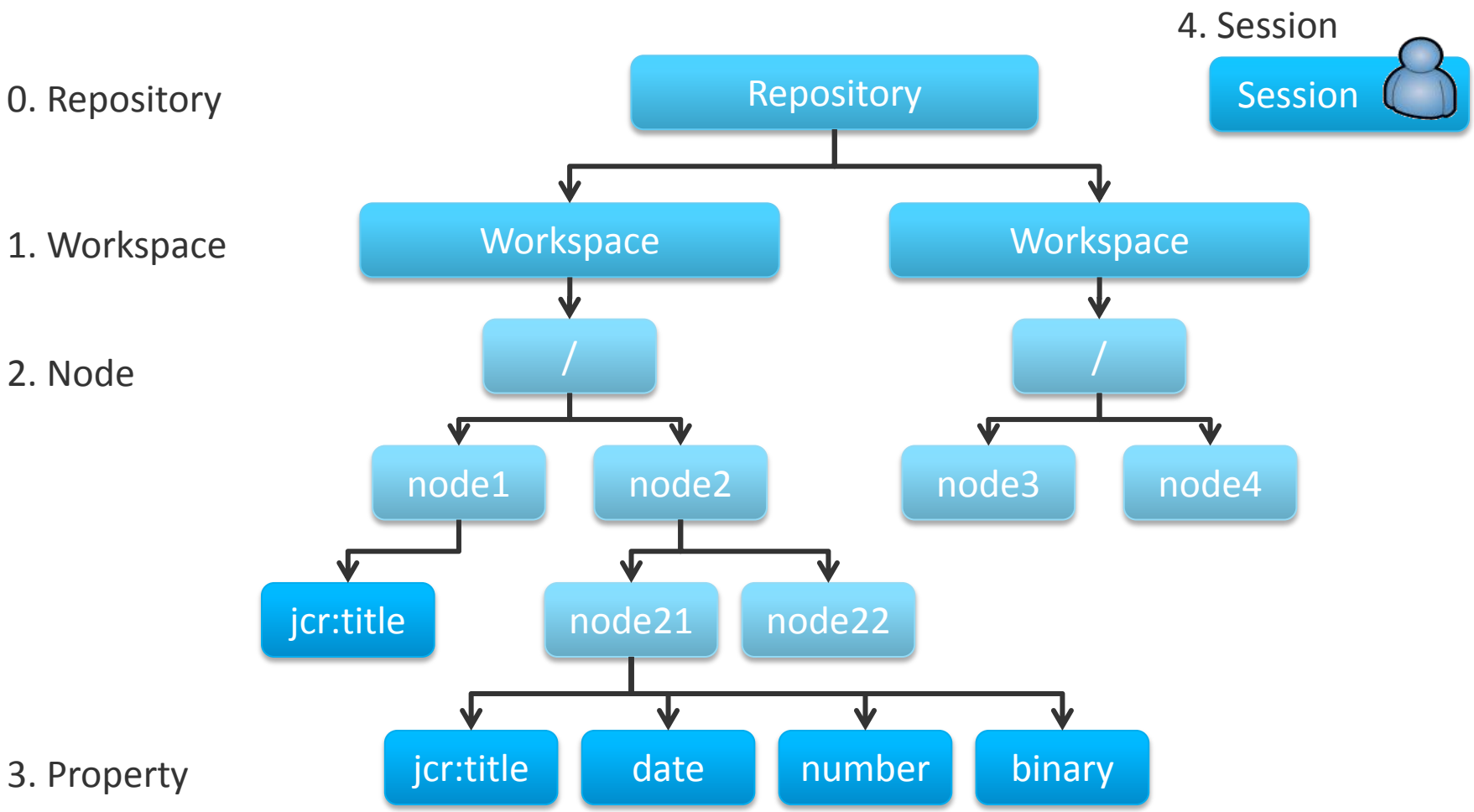
~~JCR replaces RDBMS~~

Best solution for given scenario and requirements

JCR provides an alternative data model for specific requirements of CMS and collaborative applications

The choice of technology should be dictated by Requirements and best possible fit for problem solving.

Hierarchical Structure of JCR



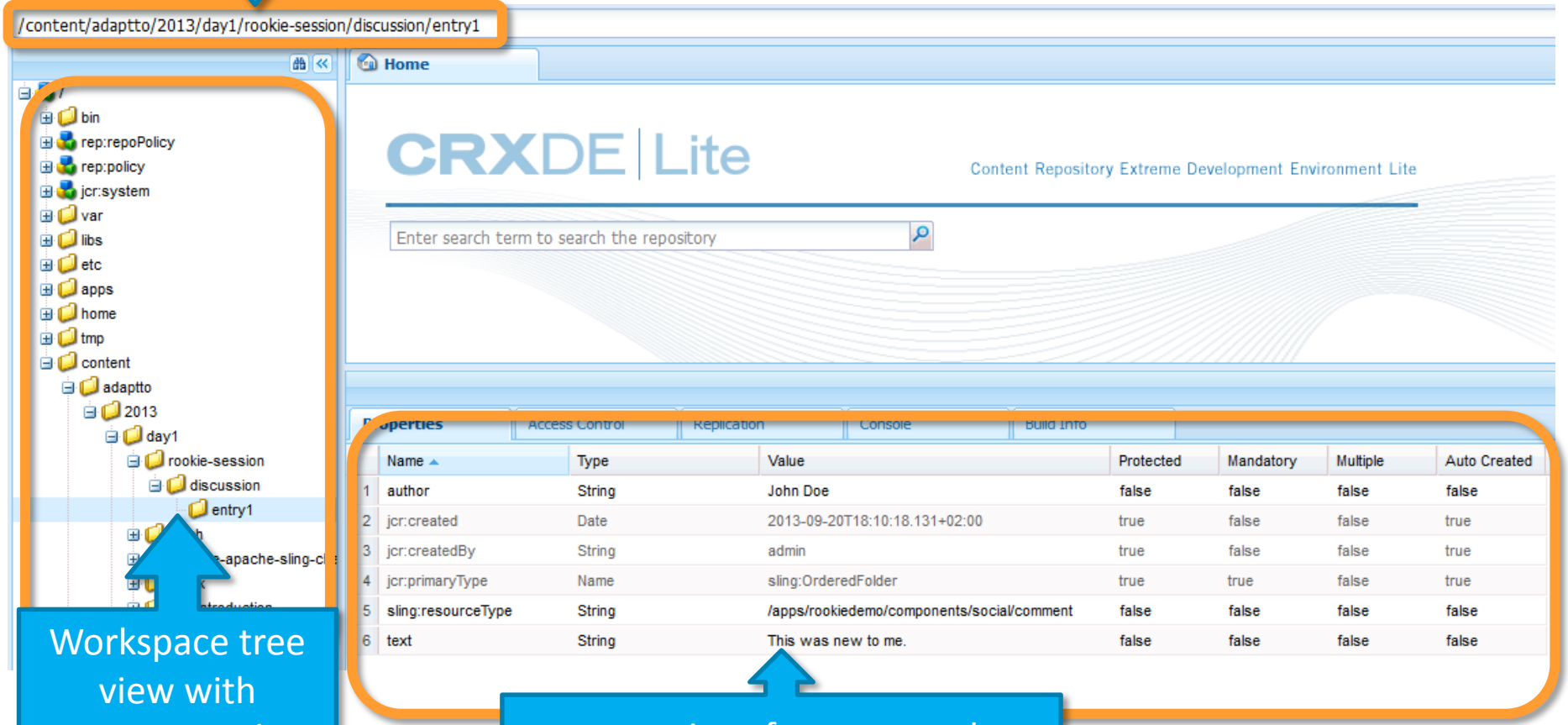
Dive into JCR with CRX DE

path to current node

/content/adaptto/2013/day1/rookie-session/discussion/entry1

current session

admin@crx.default



	Name	Type	Value	Protected	Mandatory	Multiple	Auto Created
1	author	String	John Doe	false	false	false	false
2	jcr:created	Date	2013-09-20T18:10:18.131+02:00	true	false	false	true
3	jcr:createdBy	String	admin	true	false	false	true
4	jcr:primaryType	Name	sling:OrderedFolder	true	true	false	true
5	sling:resourceType	String	/apps/rookiedemo/components/social/comment	false	false	false	false
6	text	String	This was new to me.	false	false	false	false

Workspace tree view with current node highlighted

properties of current node

JCR – Node

lunch

- Every Node has a **Primary Nodetype**
  - Nodetypes define mandatory or optional properties, data types and subnodes
  - Default: **nt:unstructured** – Does not define properties or subnodes = free data structure
  - **nt:folder** – comparable to folder in file system
  - **nt:file** and **nt:resource** – files/binary data
- Additionally a node can have an arbitrary number of Mixins:
  - Examples: **mix:title**, **mix:referencable**
- Hint: **Nodetypes** are useful in Queries

JCR – Property

jcr:title

- Contains the data of the nodes
- Data Types
  - STRING
  - LONG
  - DOUBLE
  - DECIMAL
  - BOOLEAN
  - DATE
  - BINARY
  - REFERENCE
  - …
- Multivalued Data Type
  - STRING[ ]

JCR – Code Examples

Read

```
String readJcrContent(Session pSession) throws RepositoryException {

  // get node directly
  Node day1 = pSession.getNode("/content/adaptto/2013/day1");

  // get first child node
  Node firstTalk = day1.getNodes().nextNode();

  // read property values
  String title = firstTalk.getProperty("jcr:title").getString();
  long duration = firstTalk.getProperty("durationMin").getLong();

  // read multi-valued property
  Value[] tagValues = firstTalk.getProperty("tags").getValues();
  String[] tags = new String[tagValues.length];
  for (int i=0; i<tagValues.length; i++) {
    tags[i] = tagValues[i].getString();
  }

  return "First talk: " + title + " (" + duration + " min)\n"
      + "Tags: " + Arrays.toString(tags) + "\n"
      + "Path: " + firstTalk.getPath();
}
```

JCR – Code Examples

Write

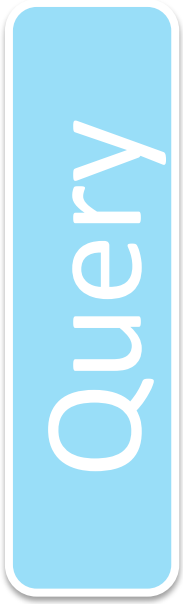
```
void writeJcrContent(Session pSession) throws RepositoryException {

  // get node directly
  Node talk = pSession.getNode("/content/adaptto/2013/day1/rookie-session");

  // write property values
  talk.setProperty("jcr:title", "My Rookie Session");
  talk.setProperty("durationMin", talk.getProperty("durationMin").getLong() + 10);
  talk.setProperty("tags", new String[] { "Sling", "JCR", "Rookie" });

  // save changes to repository (implicit transaction)
  pSession.save();
}
```

JCR - Query



- The structure and evaluation semantics of a query are defined by an *abstract query model* (AQM)
  - *JCR-SQL2*, which expresses a query as a string with syntax similar to SQL
  - *JCR-JQOM* (JCR Java Query Object Model), which expresses a query as a tree of Java objects.
  - *XPath* was introduced in JCR 1.0 and while deprecated in JCR 2.0 it is still well-supported by Jackrabbit/CRX.

JCR – Code Examples

Query

```
String queryJcrContent(Session pSession) throws RepositoryException {

  // get query manager
  QueryManager queryManager = pSession.getWorkspace().getQueryManager();

  // query for all nodes with tag "JCR"
  Query query = queryManager.
      createQuery("/jcr:root/content/adaptto//*[tags='JCR']", Query.XPATH);

  // iterate over results
  QueryResult result = query.execute();
  NodeIterator nodes = result.getNodes();
  StringBuilder output = new StringBuilder();
  while (nodes.hasNext()) {
    Node node = nodes.nextNode();
    output.append("path=" + node.getPath() + "\n");
  }

  return output.toString();
}
```

More info on Queries:

Specification: http://www.day.com/specs/jcr/2.0/6_Query.html

JBoss DNA: http://docs.jboss.org/modeshape/0.7/manuals/reference/html/jcr-query-and-search.html

David's model

- David's model:
  http://wiki.apache.org/jackrabbit/DavidsModel
  - Content first, structure later, maybe.
  - Drive the content hierarchy, don't let it happen.
  - Files are files are files.

Tools

- Tools to use to access JCR
  - Free
    - Jackrabbit explorer
      - https://code.google.com/p/jackrabbitexplorer/
    - Eclipse plugins
      - http://jcrbrowser.sourceforge.net/
  - Commercial
    - CRX DE Lite
      - http://dev.day.com/docs/en/crx/current/developing/development_tools/developing_with_crxde_lite.html

Sling

Sling - History

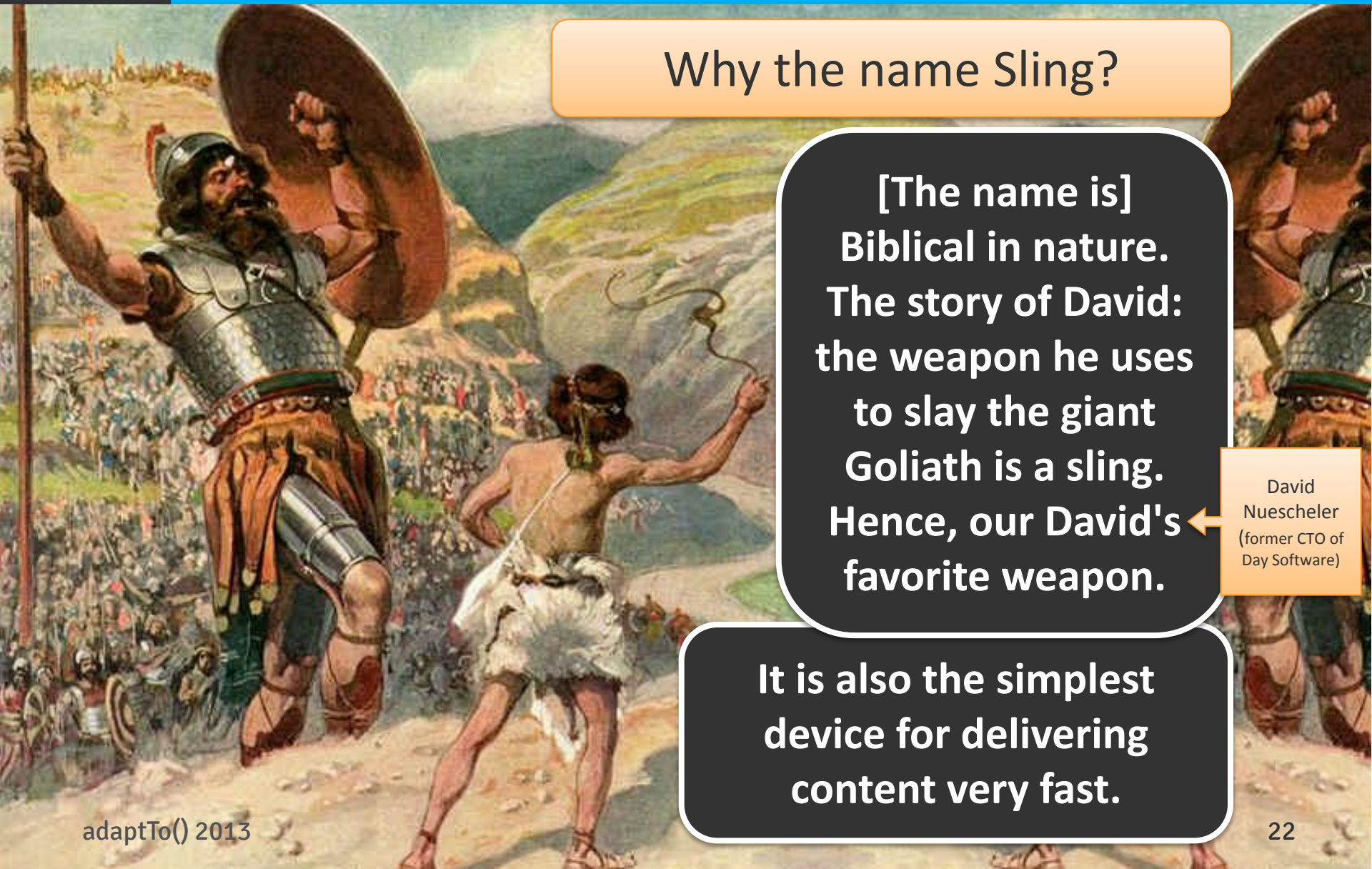






Why the name Sling?

Sling - History



Why the name Sling?

[The name is] Biblical in nature. The story of David: the weapon he uses to slay the giant Goliath is a sling. Hence, our David's favorite weapon.

David Nuescheler (former CTO of Day Software)

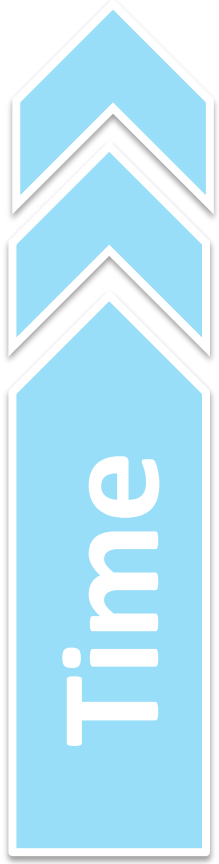
It is also the simplest device for delivering content very fast.

Sling - History

Since **June 2009** Apache top level project.

Since **September 2007** Apache Incubator project.

Started as an internal project at Day Software.



Apache Sling

- Web Application Framework
- JCR for content storage
  - or other data sources using ResourceProviders
- Process HTTP requests in a RESTful way
- Scripts or Servlets for processing
- OSGi for deploying modules at runtime

Sling encourages REST

- Thinking in resources (mapped to JCR Nodes)

adaptTo()

Sling encourages REST

- Thinking in resources (mapped to JCR Nodes)
- Representation selection (HTML, Atom, PDF...) via part of URI
- Uniform interface for content handling: GET + POST + PUT + DELETE

UnRESTful URL:

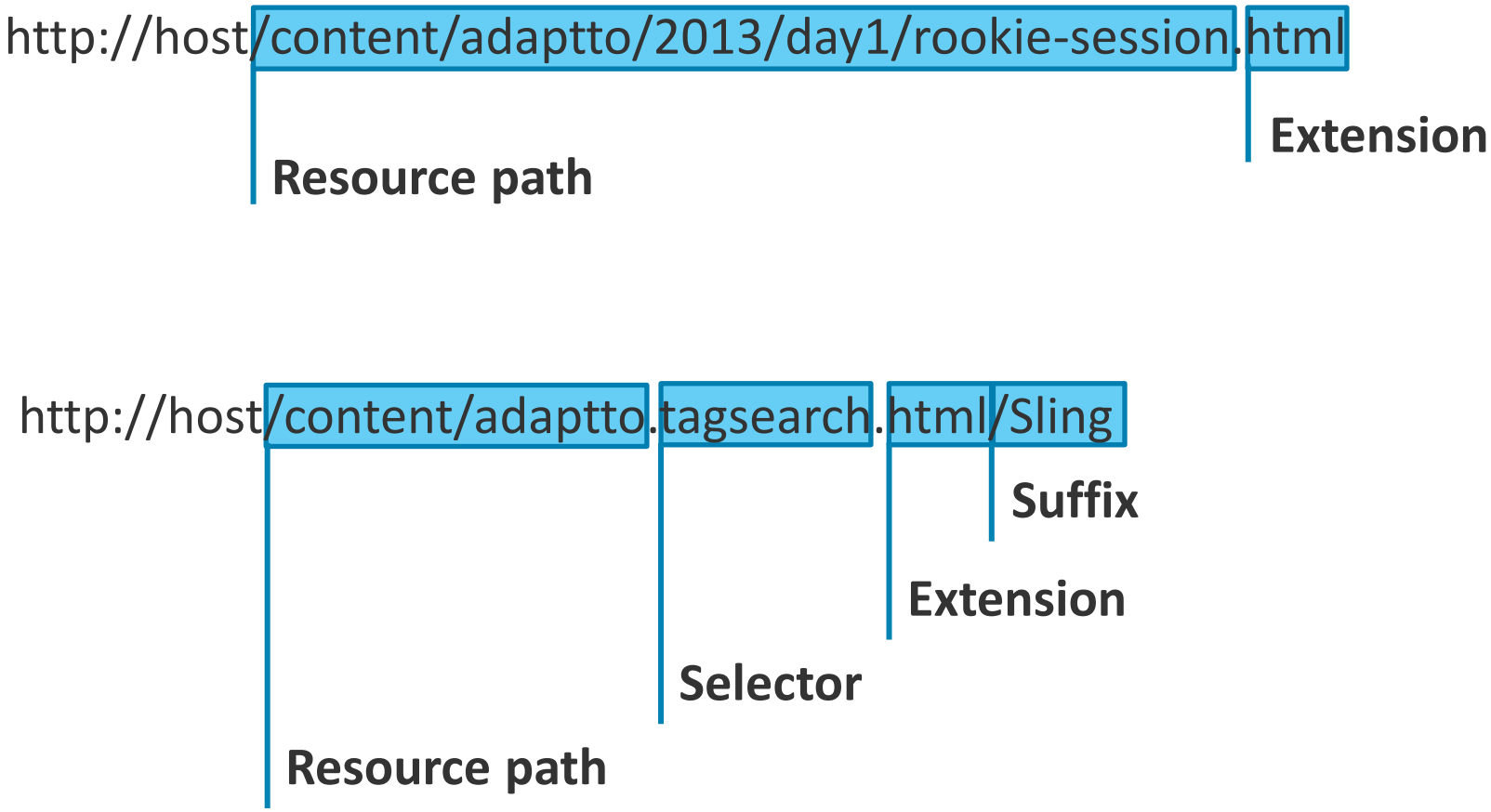
http://localhost/schedule.jsp?event=1&year=2013&id=1&format=JSON

RESTful URLs:

http://localhost:4502/content/adaptto/2013/day1/rookie-session.json

http://localhost:4502/content/adaptto/2013/day1/rookie-session.xml

URL Decomposition Examples



Sling Default JSON/XML Mapping

http://localhost:4502/content/adaptto/2013/day1/rookie-session.json


```
{
    durationMin: 135,
    jcr:description: "Basic introduction to JCR and Sling",
    speaker: "Andres Pegam, Stefan Seifert",
    sling:resourceType: "/apps/rookiedemo/components/talk",
    jcr:createdBy: "admin",
    jcr:title: "Rookie-Session: JCR & Sling",
  - tags: [
        "Sling",
        "JCR"
    ],
    startDate: "20130923T081500Z",
    jcr:created: "Sat Sep 21 2013 23:23:16 GMT+0200",
    endDate: "20130923T103000Z",
    jcr:primaryType: "sling:OrderedFolder"
}
```


http://localhost:4502/content/adaptto/2013/day1/rookie-session.xml


```
- <rookie-session jcr:primaryType="sling:OrderedFolder" durationMin="135" endDate="20130923T103000Z"
  jcr:created="2013-09-21T23:23:16.837+02:00" jcr:createdBy="admin" jcr:description="Basic introduction to JCR and Sling"
  jcr:title="Rookie-Session: JCR & Sling" sling:resourceType="/apps/rookiedemo/components/talk" speaker="Andres Pegam, Stefan Seifert"
  startDate="20130923T081500Z">
   + <discussion jcr:primaryType="sling:OrderedFolder" jcr:created="2013-09-21T23:23:16.837+02:00" jcr:createdBy="admin">
     </discussion>
  </rookie-session>
```

Resource Types

- Resources are typed.
  - sling:resourceType
- Used in script resolution.
- Inheritance.
  - sling:resourceSuperType
- Similar to a path.
  - Because ResourceTypes are Resources themselves

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
```

What does adaptTo() do?

- adaptTo() allows to “get a view of the same object in terms of another class”
  - Content is kept encapsulated.
  - Functionality is abstracted from content.
  - Implementing classes are not constrained by inheritance hierarchy.
  - Yet another form of polymorphism.
- Example: Resource to ValueMap to access the properties of a resource

JSP Example: Iterate over resources

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

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sling" uri="http://sling.apache.org/taglibs/sling"%>

<sling:defineObjects/>

<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 includes

Sling supports different types of includes to support script modularization

<sling:call script="…"/>

- Comparable to jsp:include, but supports resource type inheritance for script resolution.

<sling:include path="…"/>

- Directly include a path with resource type defined in content.

<sling:include resourceType="…"/>

- Render resource using a different resource type.

Optionally use add/replaceSelectors, replaceSuffix

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>
```

Servlet Example: Previous/Next links

Servlet in OSGi Bundle: **org.adaptto.demo.rookie.components.ResourceSiblingNavigator**

```
@SlingServlet(resourceTypes="/apps/rookiedemo/components/resourceSiblingNavigator")
public class ResourceSiblingNavigator extends SlingSafeMethodsServlet {

  protected void doGet(SlingHttpServletRequest pRequest, SlingHttpServletResponse pResponse)
      throws ServletException, IOException {
    Writer out = pResponse.getWriter();

    // get previous/next sibling
    Resource previousResource = null;
    Resource currentResource = null;
    Resource nextResource = null;
    for (Resource sibling : pRequest.getResource().getParent().getChildren()) {
      if (currentResource!=null) {
        nextResource = sibling;
        break;
      }
      else if (StringUtils.equals(sibling.getPath(), pRequest.getResource().getPath())) {
        currentResource = pRequest.getResource();
      }
      else {
        previousResource = sibling;
      }
    }

    // anchor for previous/next sibling
    if (previousResource!=null) {
      out.write(" | <a href=\"" + previousResource.getPath() + ".html\">Previous</a>");
    }
    if (nextResource!=null) {
      out.write(" | <a href=\"" + nextResource.getPath() + ".html\">Next</a>");
    }
  }

}
```

JSP Example: JCR queries in Sling

JSP Script in JCR: **/apps/rookiedemo/components/index/tagsearch.jsp**

Selector resolution mapping

```
<!doctype html>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<%@taglib prefix="sling" uri="http://sling.apache.org/taglibs/sling"%>
<sling:defineObjects/>

<%-- Extract tag name from suffix (remove leading slash), and build XPath query --%>
<c:set var="tagName" value="${fn:substring(slingRequest.requestPathInfo.suffix,1,100)}"/>
<c:set var="xpathQuery" value="/jcr:root${resource.path}//*[tags='${tagName}']"/>

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

    <%-- Execute XPath query and display results --%>
    <sling:findResources var="searchResult" query="${xpathQuery}" language="xpath"/>
    <ul>
      <c:forEach var="child" items="${searchResult}">
        <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>

  </body>
</html>
```

Writing data to repository with Sling

- SlingPostServlet
  - Supports writing back data to repository (resources) without custom code
  - Maps POST parameters to property names
  - Supports additional logic that can be triggered by special parameter names
  - Make sure POST is only possible with proper authentication (delegates to repository authentication by default)

JSP Example: Post new discussion entry

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

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sling" uri="http://sling.apache.org/taglibs/sling"%>
<sling:defineObjects/>

<%-- 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/rookiedemo/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>
```

Servlet Example: Render discussion entry

Servlet in OSGi Bundle: **org.adaptto.demo.rookie.components.DiscussionComment**

```
@SlingServlet(resourceTypes="/apps/rookiedemo/components/social/comment")
public class DiscussionComment extends SlingSafeMethodsServlet {

  protected void doGet(SlingHttpServletRequest pRequest, SlingHttpServletResponse pResponse)
      throws ServletException, IOException {
    Writer out = pResponse.getWriter();

    // read properties via Sling API
    ValueMap props = ResourceUtil.getValueMap(pRequest.getResource());
    String author = props.get("author", "Anonymous");
    Date created = props.get("jcr:created", Date.class);
    String text = props.get("text", "");

    // output comment as HTML
    out.write("<p>");
    out.write("<em>" + StringEscapeUtils.escapeHtml4(author)
        + " (" + DateFormat.getDateTimeInstance().format(created) + ")</em></br>");
    out.write(StringEscapeUtils.escapeHtml4(text));
    out.write("</p>");

  }

}
```

The example shows the usage of Sling API for reading content from resources.
Less verbose and more convenient than directly using JCR API.

Servlet Example: Custom POST, Sling CRUD

Servlet in OSGi Bundle: **org.adaptto.demo.rookie.components.LikeMe**

```
@SlingServlet(resourceTypes="/apps/rookiedemo/components/talk", selectors="likeme", methods="POST")
public class LikeMe extends SlingAllMethodsServlet {

  protected void doPost(SlingHttpServletRequest pRequest, SlingHttpServletResponse pResponse)
      throws ServletException, IOException {
    updateLikeCounter(pRequest);
    // return to main view
    pResponse.sendRedirect(pRequest.getResource().getPath() + ".html");
  }

  private void updateLikeCounter(SlingHttpServletRequest pRequest) throws PersistenceException {
    ValueMap props = ResourceUtil.getValueMap(pRequest.getResource());

    // check if a user with this ip address has already liked this
    String ipAddress = pRequest.getRemoteAddr();
    String[] likedAddresses = props.get("likedAddresses", new String[0]);
    if (ArrayUtils.contains(likedAddresses, ipAddress)) {
      return;
    }

    // increment like counter and store ip address
    ValueMap writeProps = pRequest.getResource().adaptTo(ModifiableValueMap.class);
    writeProps.put("likes", writeProps.get("likes", 0L) + 1);

    List<String> updatedLikedAddresses = new ArrayList<>(Arrays.asList(likedAddresses));
    updatedLikedAddresses.add(ipAddress);
    writeProps.put("likedAddresses", updatedLikedAddresses.toArray());

    // save to repository
    pRequest.getResourceResolver().commit();
  }

}
```

OSGi example: Scheduled background job

Component in OSGi Bundle: **org.adaptto.demo.rookie.services.CommentCleanUp**

```
@Component(immediate = true, metatype = true, label = "adaptTo() Rookie Demo Comment Cleanup Service")
@Service(value = Runnable.class)
public class CommentCleanUp implements Runnable {
  @Property(value = "0 0/15 * * * ?", label = "Scheduler Expression")
  private static final String PROPERTY_CRON_EXPRESSION = "scheduler.expression";
  @Reference
  ResourceResolverFactory mResourceResolverFactory;

  public void run() {
    ResourceResolver adminResolver = mResourceResolverFactory.getAdministrativeResourceResolver(null);

    // fire query to get all comment nodes
    Iterator<Resource> comments = adminResolver.findResources("SELECT * "
        + "FROM [sling:OrderedFolder] "
        + "WHERE ISDESCENDANTNODE([/content/adaptto]) "
        + "AND [sling:resourceType]='/apps/rookiedemo/components/social/comment'", Query.JCR_SQL2);

    // iterate over all comments and remove those that have empty text
    while (comments.hasNext()) {
      Resource comment = comments.next();
      ValueMap props = ResourceUtil.getValueMap(comment);
      if (StringUtils.isEmpty(props.get("text", String.class))) {
        adminResolver.delete(comment);
      }
    }

    // save changes to repository
    if (adminResolver.hasChanges()) {
      adminResolver.commit();
    }
    adminResolver.close();
  }
}
```

Is that all Sling is?

- This was just a glimpse at some parts.
- Sling is much bigger than this.
- And it is still evolving...

${EndOfSessionTitle}

Questions?

Thank you for listening
and enjoy adaptTo() 2013.