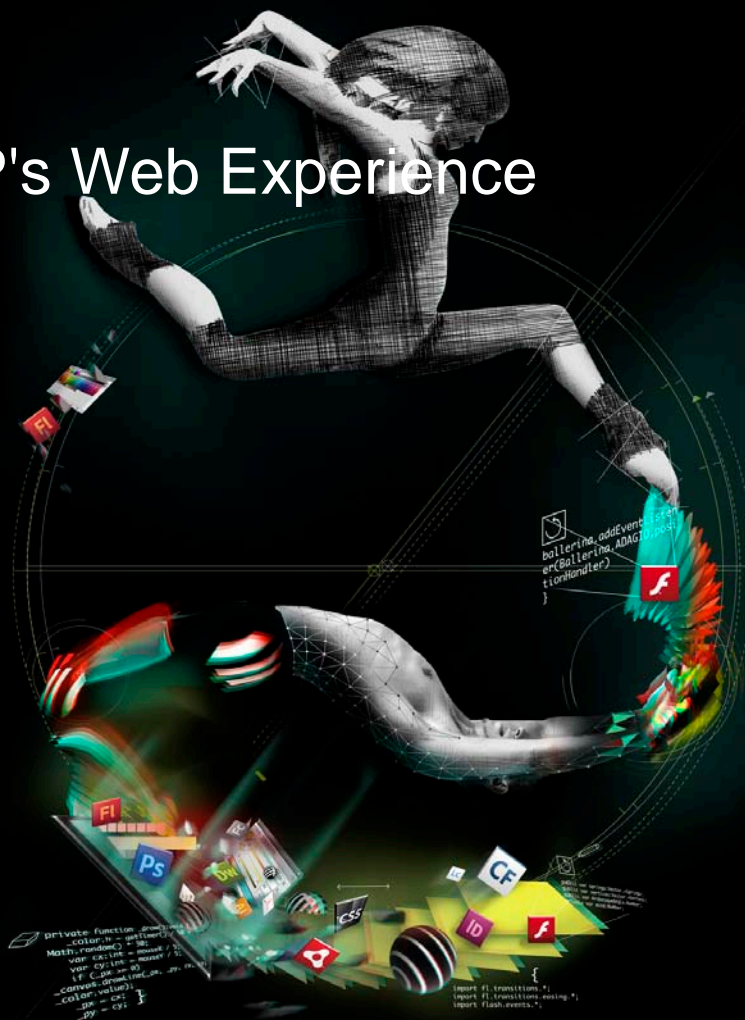




MAX

Creating Components for ADEP's Web Experience Management (WEM) Solutions

Ray Blaak, CTO, ensemble.com



- Review key aspects of making a CQ application
- Use a simple example: Adobe Products Table Component
- Also uses custom login module to authorize Adobe ID users
- Focus is on end-to-end considerations, not Web UI building as such

- Web Authoring Components
 - How to make available for drag and drop
- UI Components via JSPs
- Exposing Web Services via JSPs
- Back end Services
- Persistence Approaches
- THEME: Everything uses the repository!

- A reusable web component that presents a table of Adobe products

Sample Products Table

Product Name	Description	Version	Keys	Downloads
Flash Builder	Flash Builder 4.x Download	4.1.23	View	Download (460.38 MB)
Current Key: Not Allocated				Request Key
LiveCycle Suite	LiveCycle 5.x Download	5.2.33	View	Download (3.07 GB)
Current Key: 1111-2222-abcd-zzzz				
Photoshop	Photoshop 9.x Download	9.5.0	View	Download (460.38 MB)

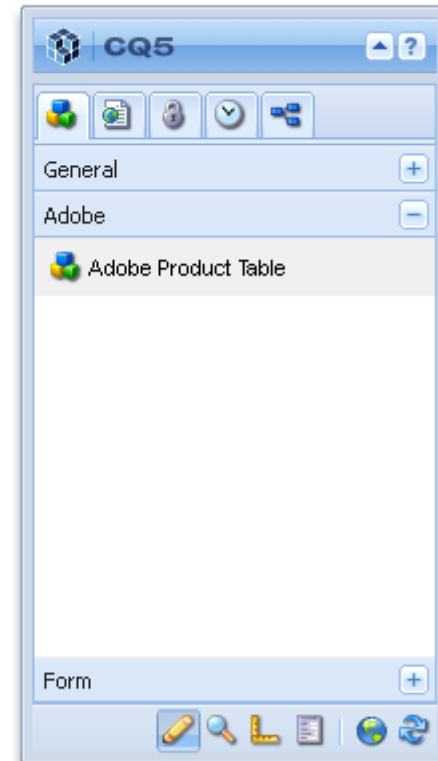
Start of Form | Edit Delete New... Paste

Sample Products Table

Product Name	Description	Version	Keys	Downloads
Flash Builder	Flash Builder 4.x Download	4.1.23	View	Download (460.38 MB)
LiveCycle Suite	LiveCycle 5.x Download	5.2.33	View	Download (3.07 GB)

End of Form | Edit New... Paste

Drag components or assets here



Adobe Products Table, Applying Products Filter

Start of Form | Edit Delete New... Paste

Sample Products Table

Product Filter

Sku Filter

Sku Regular Expression Filter

Help OK Cancel

End of Form | Edit New... Paste

Drag components or assets here

CQ5

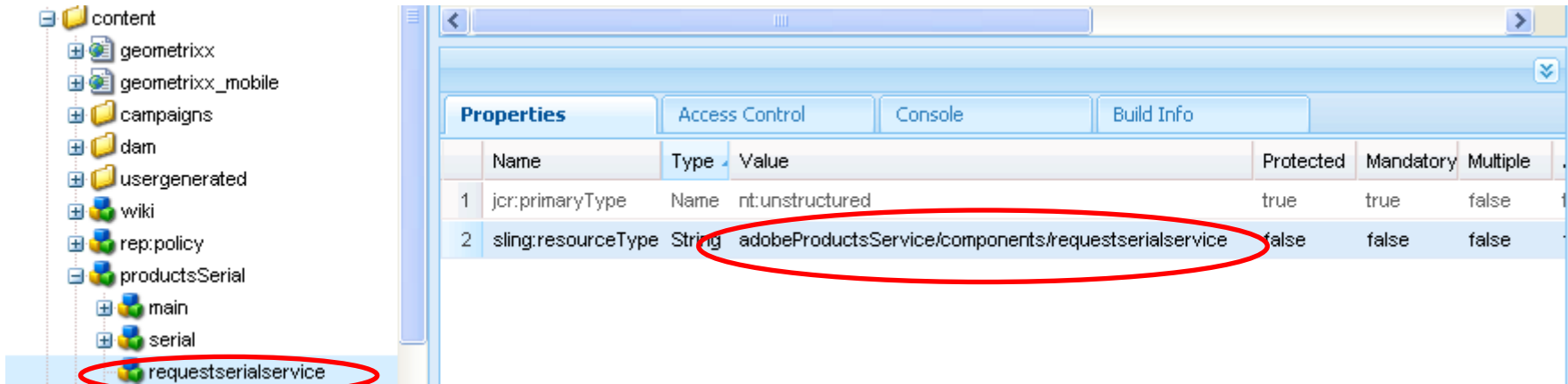
Page Properties...
Create Child Page
Copy Page
Move Page
Delete Page
Activate Page
Lock Page
Show References...
Rollout Page

Adobe Products Table, application definition

The screenshot shows the Adobe CRXDE Lite interface. The left sidebar displays a file tree for the path `/apps/adobeProductsService/components/producttable`. The `adobeProductsService` folder and its sub-items (`producttable.jsp`, `GET.jsp`) are circled in red. The main content area shows the 'Properties' tab for the selected component. The 'Properties' table lists various attributes, with the `componentGroup` value 'Adobe' and the `jcq:primaryType` value 'cq:Component' circled in red.

	Name	Type	Value	Protected	Mandatory	Multiple
1	allowedParents	String[]	*/parsys	false	false	true
2	componentGroup	String	Adobe	false	false	false
3	jcq:created	Date	2011-09-20T16:15:22.358-07:00	true	false	false
1	jcq:createdBy	String	admin	true	false	false
5	jcq:primaryType	Name	cq:Component	true	true	false
1	jcq:title	String	Adobe Product Table	false	false	false

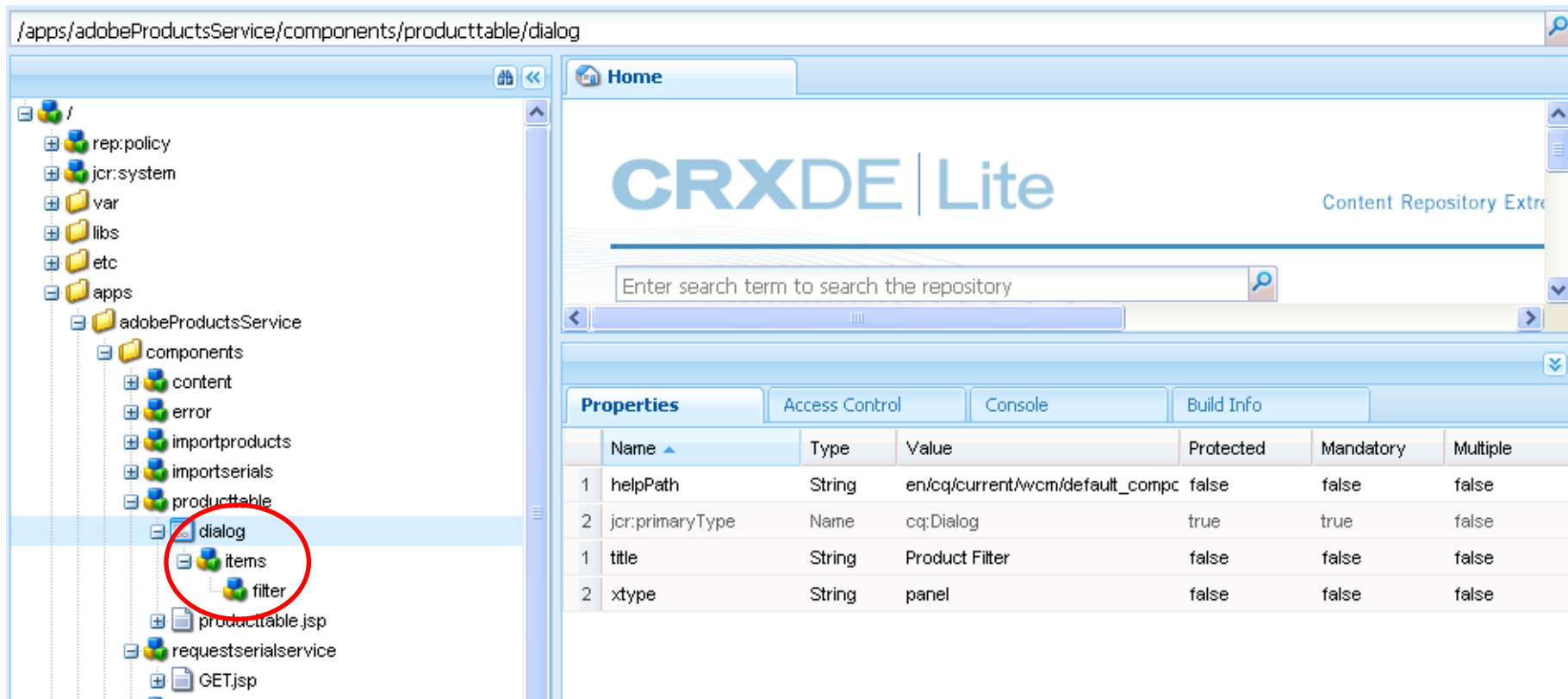
Adobe Product Table, exposing the request serial service



The screenshot displays the Adobe Experience Manager interface. On the left, a content tree shows a folder named 'requestserialservice' circled in red. On the right, the 'Properties' tab is active, showing a table with two rows. The second row is circled in red, showing the property 'sling:resourceType' with the value 'adobeProductsService/components/requestserialservice'.

	Name	Type	Value	Protected	Mandatory	Multiple
1	jcr:primaryType	Name	nt:unstructured	true	true	false
2	sling:resourceType	String	adobeProductsService/components/requestserialservice	false	false	false

Adobe Product Table, enable metadata dialogs



The screenshot shows the CRXDE Lite interface. The left sidebar displays a file tree for the path `/apps/adobeProductsService/components/producttable/dialog`. The `dialog` folder is highlighted with a red circle. The main area shows the `CRXDE | Lite` header and a search bar. Below the search bar, the `Properties` tab is active, displaying a table of metadata for the selected `dialog` folder.

	Name ▲	Type	Value	Protected	Mandatory	Multiple
1	helpPath	String	en/cq/current/wcm/default_compc	false	false	false
2	jcr:primaryType	Name	cq:Dialog	true	true	false
1	title	String	Product Filter	false	false	false
2	xtype	String	panel	false	false	false

Adobe Product Table, producttable.jsp

```
<%  
    AdobeProductsService svc = new AdobeProductsService(currentNode.getSession());  
    String filter= properties.get("jcr:filter", "");  
    List<ProductData> allProducts = filter.isEmpty() ? svc.getAllProducts() : svc.getAllProducts(filter);  
%>  
<table>...  
<%  
    for(ProductData aProduct : allProducts) {  
%>  
    <tr><td class="row<%= i % 2 %>"><%=aProduct.name%></td>...</tr>  
    <tr id="hiddenRow<%=i%>" class="hiddenRow">...  
    <td class="hiddenRightCell<%=i%2%>" align="middle" style="border-left: 0px">  
        <%if(serial==null) { %>  
            <a href="javascript:ajaxGetSerial('<%=aProduct.sku%>')" id="reqKey<%=aProduct.sku%>">Request Key</a>  
        <%} %>  
    </td>  
    </tr>  
    <% } %>  
</table>
```

Adobe Product Table, producttable AJAX call

```
function ajaxGetSerial(sku) {  
  
    var url = "/content/productsSerial/requestserialservice.json?productSku=" + sku;  
    // Assume Firefox, Opera, Safari for this example  
    var req = new XMLHttpRequest();  
    req.onreadystatechange = processResult;  
    req.open("GET",url,true);  
    req.send(null);  
  
    function processResult() {  
        if ((req.readyState == 4 || req.readyState == "complete") && (req.status==200)) {  
            var jsonResponse = eval("(" + req.responseText + ")");  
            var serialDiv = document.getElementById("serial" + jsonResponse.productSku);  
            serialDiv.innerHTML = jsonResponse.productSerial;  
            var reqKey = document.getElementById("reqKey" + jsonResponse.productSku);  
            reqKey.style.display = "none";  
        }  
    }  
}
```

Adobe Product Table, requestserialservice/GET.jsp

```
<%@ page contentType="application/json" %>
<%@ page import="javax.scr.Session,com.adobe.crx.adobeproducts.*"%>
<%@include file="/libs/foundation/global.jsp"%>
<%/ "serialNum": "=getNewProductSerialNumberForCurrentUser(request.getParameter("productSku"))"
AdobeProductsService svc = new AdobeProductsService(currentNode.getSession());

String newSerial;
try
{
newSerial=svc.getNewProductSerialNumberForCurrentUser((String)request.getParameter("productSku"));
}
catch(Exception e)
{
newSerial = "None Available";
}%>
{
"productSku": "<%= (String)request.getParameter("productSku") %>",
"productSerial": "<%= newSerial %>"
}
```

We desire a service to abstract away the details of persistency in the repository

```
public class AdobeProductsService {  
    public AdobeProductsService(Session session)  
    public List<ProductData> getAllProducts(String skuRegExp)  
    public String getNewProductSerialNumber(final String sku, final String forUser) // use session.getUserID() for current user  
    public void importProducts(final List<ProductData> products)  
    public void importSerialNumbers(final String sku, final List<String> newSerialNumbers)  
}
```

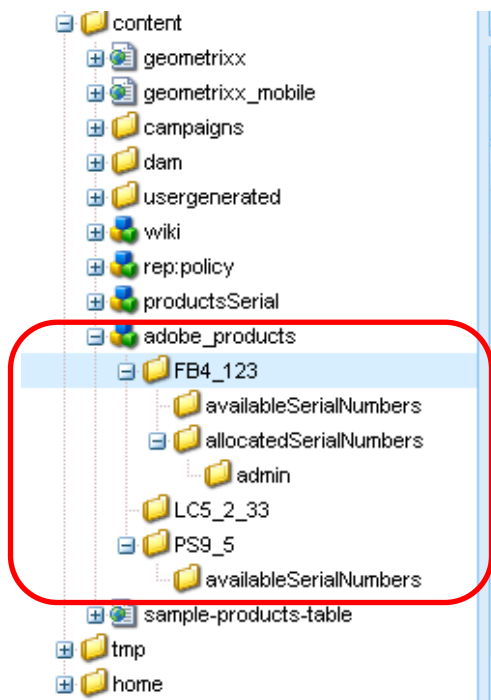
```
public class ProductData {  
    public String sku; // must be unique  
    public String name;  
    public String description;  
    public String version;  
    public String downloadUrl;  
    public long downloadSize;  
}
```

- Persistence is based on updating nodes and properties in the repository
- We place our data in a known location under the content root, e.g.

```
Node productsRoot = session.getRootNode().getNode("content/adobe_products");
```

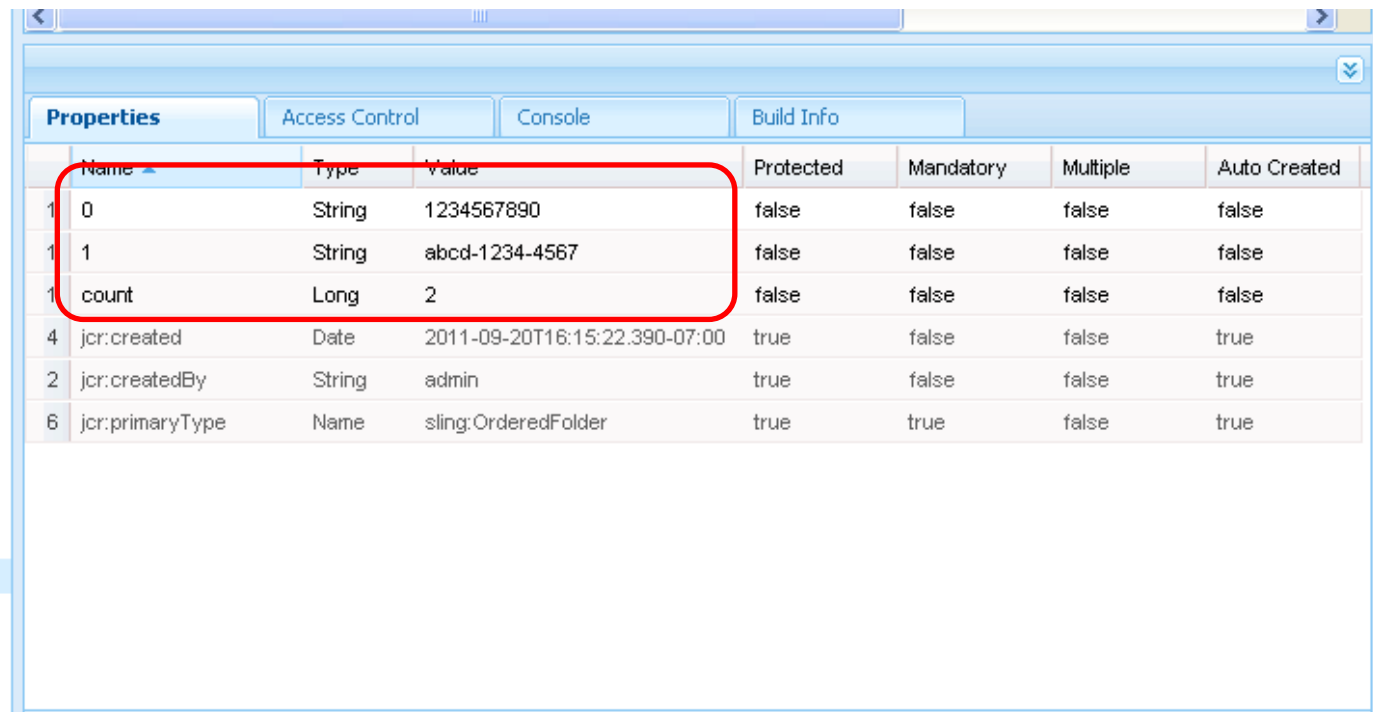
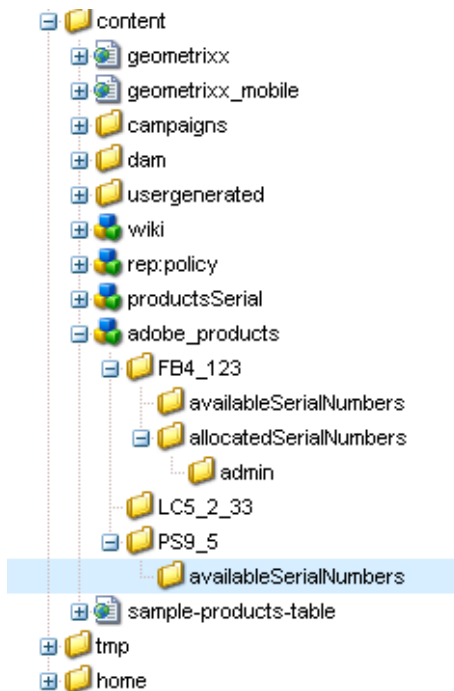
- From there we map ProductData details, allocated and unallocated serial numbers to the node subtree.
- Take care to invoke session.save() appropriately
- Take care to synchronize access to repository updates

Adobe Product Table, persistence, product nodes



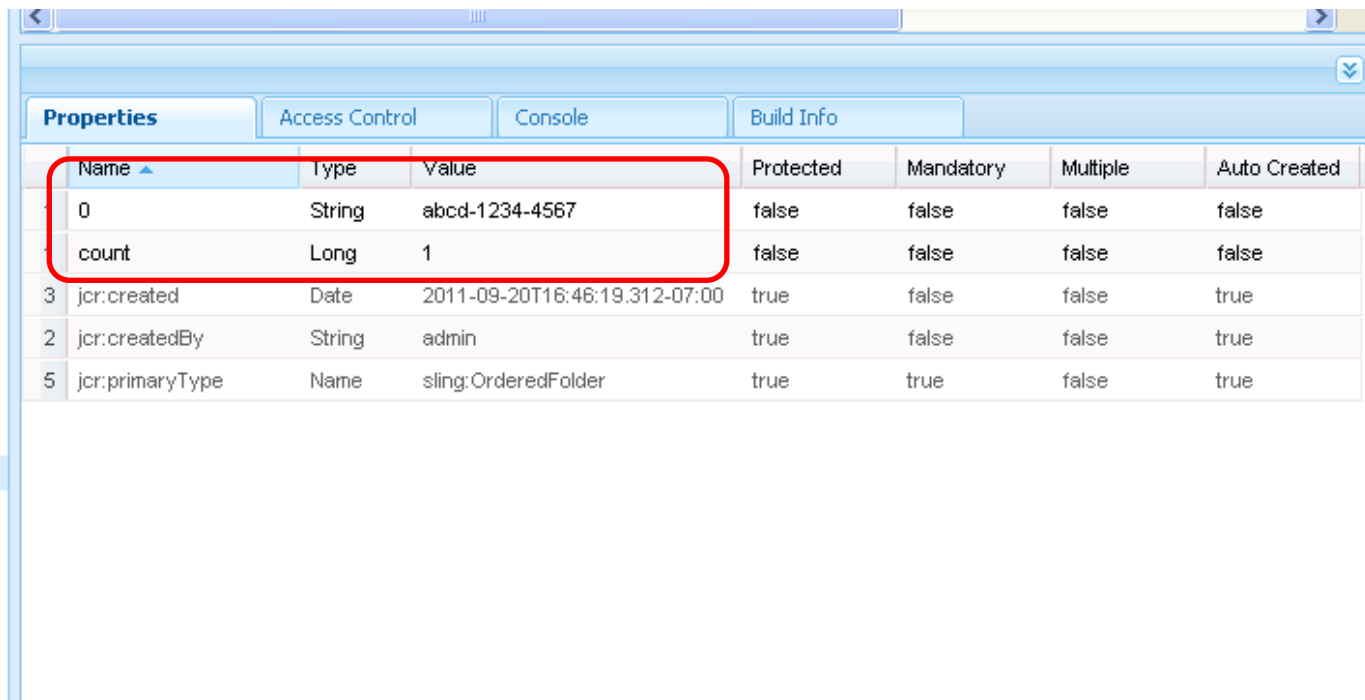
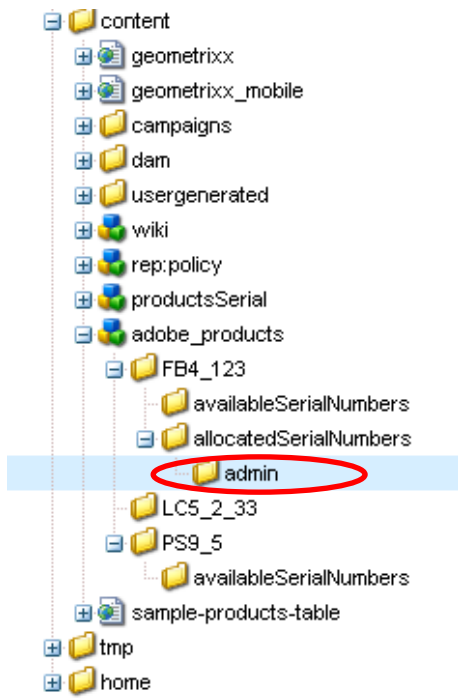
Properties							
	Name	Type	Value	Protected	Mandatory	Multiple	Auto Created
1	bucket	String		false	false	false	false
1	description	String	Flash Builder 4.x Download	false	false	false	false
1	downloadSize	String	482747399	false	false	false	false
4	downloadUrl	String	http://download.mozilla.org/?prod...	false	false	false	false
4	jcr:created	Date	2011-09-20T16:15:22.374-07:00	true	false	false	true
3	jcr:createdBy	String	admin	true	false	false	true
7	jcr:primaryType	Name	slings:OrderedFolder	true	true	false	true
1	key	String		false	false	false	false
4	name	String	Flash Builder	false	false	false	false
5	sku	String	FB4_123	false	false	false	false
7	version	String	4.1.23	false	false	false	false

Adobe Product Table, persistence, available serial numbers



	Name	Type	Value	Protected	Mandatory	Multiple	Auto Created
1	0	String	1234567890	false	false	false	false
1	1	String	abcd-1234-4567	false	false	false	false
1	count	Long	2	false	false	false	false
4	jcr:created	Date	2011-09-20T16:15:22.390-07:00	true	false	false	true
2	jcr:createdBy	String	admin	true	false	false	true
6	jcr:primaryType	Name	slings:OrderedFolder	true	true	false	true

Adobe Product Table, persistence, allocated serial numbers



A screenshot of a 'Properties' window with tabs for 'Properties', 'Access Control', 'Console', and 'Build Info'. The 'Properties' tab is active, showing a table with columns: Name, Type, Value, Protected, Mandatory, Multiple, and Auto Created. A red box highlights the first two rows of the table.

Name	Type	Value	Protected	Mandatory	Multiple	Auto Created
0	String	abcd-1234-4567	false	false	false	false
count	Long	1	false	false	false	false
3 jcr:created	Date	2011-09-20T16:46:19.312-07:00	true	false	false	true
2 jcr:createdBy	String	admin	true	false	false	true
5 jcr:primaryType	Name	sling:OrderedFolder	true	true	false	true

- Use Maven
 - helps tremendously with the millions of versioned jars you need to refer to
- Source control is an issue
 - CQ wants to manage its content
 - Usually projects need to be in local the corporate SVN repository as well
 - Can use VLT to manage
 - Maps repository to the file system
 - Need to continually checkin/out from CQ to filesystem to SVN and back
 - We found it simpler to edit to do edits outside of CQ
 - Would reimport package as needed to run
 - Use build scripts to help package and deploy

- All (most?) Adobe product users have a registered Adobe ID
- We wish to have any registered Adobe ID user be authorized to access product-related pages
- CQ supports custom login modules
 - Based on JAAS (Java Authentication and Authorization Service)
 - Controlled by a jaas.config file
- Unknown users have authorizations attempt against the Adobe ID servers
- If successful, their user info is cached in the CQ user repository
- They can then access the system as normal users

Adobe ID Login Module, login example



CQ5 

User name

Password



Adobe ID Login Module, cached user data

Path: /home/groups/a/adobeid-everyone

Name	Type	Value
.	rep:Group	
jcr:created	Date	2011-09-21T15:51:18.468-07:00
jcr:createdBy	String	system
rep:members	WeakReference[]	jeremy@ensemble.com
rep:principalName	String	adobeid-everyone
preferences	nt:unstructured	
privileges	sling:Folder	
profile	nt:unstructured	
rep:members	rep:Members	

PERM_PHONE	Boolean	true
PERM_PHONE_RENT	Boolean	false
ShortForm	Boolean	false
Status	Long	0
adobePartnerLastUpdated	Long	1316645489983
email	String	JEREMY@ENSEMBLE.COM
familyName	String	Liao
givenName	String	Jeremy
isAdobeIdUser	Boolean	true
isAdobePartner	Boolean	false
rep:fullname	String	Jeremy Liao

<!-- disable the standard login

```
<LoginModule class="com.day.crx.core.CRXLoginModule">
```

```
<param name="anonymousId" value="anonymous"/>
```

```
<param name="adminId" value="admin"/>
```

```
<param name="tokenExpiration" value="43200000"/>
```

```
</LoginModule>
```

-->

<!-- Make our custom login module available, configure via jaas.config -->

```
<Module class="com.adobe.crx.auth.renga.AdobeIdLoginModule">
```

```
<param name="configWspName" value="crx.default"/>
```

```
<param name="configPath" value="/apps/adobeid/config" />
```

```
</Module>
```

```
com.day.crx {  
    com.day.crx.core.CRXLoginModule sufficient;  
    com.adobe.crx.auth.AdobeIdLoginModule required  
        renga.env="na1r"  
        default.user.groups="adobeid-everyone";  
};
```

- In theory need only implement `javax.security.auth.spi.LoginModule`
- In practice extend from `CRXLoginModule`

```
public class AdobeldLoginModule extends CRXLoginModule {  
  
    protected Principal getPrincipal(Credentials credentials) {  
        // check for if existing user, don't return null for unknown users  
    }  
  
    protected boolean authenticate(Principal principal, Credentials credentials) {  
        // key override, reach out to adobe id servers  
    }  
  
    public boolean commit() {  
        // finally create or update the user data in repository here  
    }  
}
```

- `getPrincipal()` has to return the user credentials even for unknown users
 - Otherwise `authenticate()` will not be invoked
- Cache authorization results of external id server.
 - This is critical.
 - Server page accesses can cause waves of authentication requests
 - The number of expensive id server calls need to be minimized
 - Need to synchronize and serialize updates to result cache
- In `commit()` be sure to synchronize and serialize updates to the user repository
- Authenticated users are implicitly created in the repository with appropriate default user group that defines access permissions.



Follow up: blaak@ensemble.com

