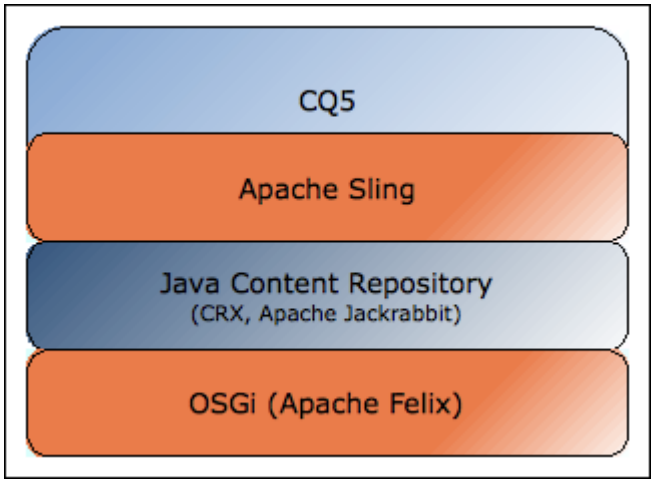
Contents

Qus: Technology Stack of CQ5 and describe each one?

CQ5 Technology Stack



Apache Sling

Apache Sling is a web application framework for content-centric applications, using a Java Content Repository, such as Apache Jackrabbit or CRX, to store and manage content.

Sling:

- is based on **REST** principles to provide easy development of content-oriented applications.
- is embedded within CQ5.
- is used to process HTTP rendering and data-storage requests which assemble, render and send the content to a client (i.e. the *new delivery*).
- maps Content objects to Components (which render them and process incoming data).
- comes with both server-side and AJAX scripting support.
- can be used with a range of scripting languages, including JSP, ESP and Ruby.
- started as an internal project of Day Management AG .
- has been contributed to the Apache Software Foundation.

Note

See http://incubator.apache.org/projects/sling.html for more information.

OSGi (Apache Felix)

CQ5 is built within an application framework which is based on the OSGi Service Platform Release 4.OSGi technology

- "is the dynamic module system for Java™."
- comes under the classification Universal Middleware.
- "provides the standardized primitives that allow applications to be constructed from small, reusable and collaborative components. These components can be composed into an application and deployed."
- OSGi bundles can contain compiled Java code, scripts, content that is to be loaded in the repository, and configuration or additional files, as needed.
- allows the bundles to be loaded, and installed, during normal operations. In the case of CQ5, this is managed by the Sling Management Console.

Apache Felix has been used to implement this framework.

- "Apache Felix is a open-source project to implement the OSGi R4 Service Platform, which includes the OSGi framework and standard services, as well as providing and supporting other interesting OSGi-related technologies."

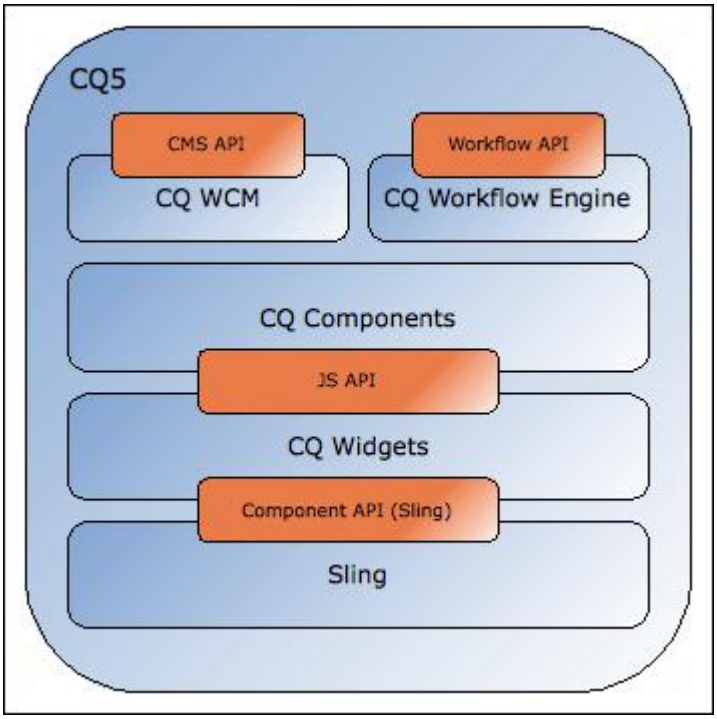
Java Content Repository (JSR-170 API)

A JCR uses the JSR-170 API to access the content repository using Java, independent of the physical implementation.

Inside CQ5

CQ5 forms a stable platform for content-centric applications such as CQ WCM:

CQ5 Internal Layers



CQ WCM

Web Content Management within the CQ5 platform allows you to generate and publish pages to your website..

CQ Workflow Engine

The CQ Workflow Engine is a powerful and easy to use process engine that can be used by all applications running on the CQ5 platform. A Java API and RESTful HTTP interface is also provided for access by applications outside CQ5.Within CQ WCM workflows can be used to control the process of generating and publishing content, which are often subject to organizational processes, including steps such as approval and sign-off by various participants.

CQ Components

Components provide the logic (code) to render content. They include both templates and specific components such as Text with Image, Column Control and Subtitle amongst others. Components are based on a combination of widgets, replacing the CFC from Communiqué 4.

CQ Widgets

Widgets are the basic elements used to implement a specific user function, often the editing of a piece of content; they include buttons, radio-boxes, dialogs, etc.

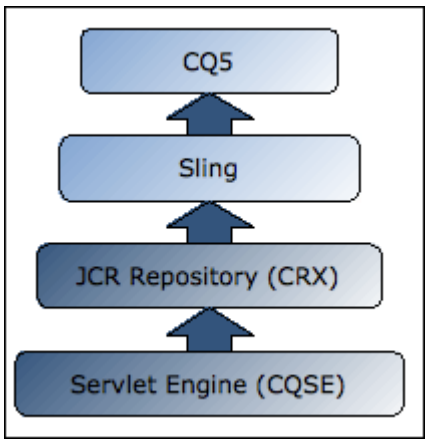
Apache Sling

The Component Framework (Sling) provides the underlying mechanisms for rendering content.

Server Startup Sequence

CQ5 has drastically reduced startup time as compared to Communiqué 4. For CQ WCM (straight out-of-the-box) the startup time is now in the order of 5 seconds. The elements required for CQ WCM are started in the following sequence (bottom-up):

Server Startup Sequence



Repository Basics – JCR / JSR and CRX?

JCR API is defined in JSR-170 and JSR-283. Apache Jackrabbit is the reference implementation of these JSRs. Adobe CRX is the Adobe commercial extension of Jackrabbit.

What are Namespaes?

Adobe CQ uses namespaces to maintain modularity between the properties and also tags. Namespaces help in distinguishing different modules of the project. We create thousands of properties in our project and many times it consists of just words. Hence to make the properties names meaningful we can add a namespace to project related namespaces.

Qus: Content Modeling: David's Model

1. Data First. Structure Later. Maybe.
2. Drive the content hierarchy, don't let it happen. – Properly Design the content Hierarchy
3. Workspaces are for clone(), merge() and update()
4. Beware of Same Name Siblings
5. References considered harmful
6. Files are Files are Files
7. ID's are evil

Qus: Boosting

Any particular property or search term can be "boosted" either at indexing time or at runtime. Boosting allows you to control the relevance of a document by boosting its term. For example,

boosting can specify that a match on the “Title” property of a node is more important than the match on the “Text” property.

```
<index-rule nodeType=”nt:unstructured” boost=”2.0” condition=”@priority = high”>
       <property>Text</property>
</index-rule>
```

Qus: What is the search engine used in JackRabbit?

Jackrabbit includes Apache Lucene as its Search and Indexing engine. Lucene synchronously indexes content in the repository.

Qus: Development objects

The following are of interest at the development level:

Node (and their properties)

Nodes and their properties store content, CQ object definitions, rendering scripts and other data.

Nodes define the content structure, and their properties store the actual content and metadata.

Content nodes drive the rendering. Sling gets the content node from the incoming request. The property sling:resourceType of this node points to the Sling rendering component to be used.

A node, which is a JCR name, is also called a resource in the Sling environment.

Note

In CQ, everything is stored in the repository.

Widget

In CQ all user input is managed by widgets. These are often used to control the editing of a piece of content.

Dialogs are built by combining Widgets.

Dialog

A dialog is a special type of widget.

To edit content, CQ uses dialogs defined by the application developer. These combine a series of widgets to present the user with all fields and actions necessary to edit the related content.

Dialogs are also used for editing metadata, and by various administrative tools.

Component

A software component is a system element offering a predefined service or event, and able to communicate with other components.

Within CQ a component is often used to render the content of a resource. When the resource is a page, the component rendering it is called a Top-Level Component or a Pagecomponent. However, a component does not have to render content, nor be linked to a specific resource; for example, a navigation component will display information about multiple resources.

The definition of a component includes:,

- the code used to render the content
- a dialog for the user input and the configuration of the resulting content.

Template

A template is the blueprint for a specific type of page. When creating a page in the siteadmin the user has to select a template. The new page is then created by copying this template.

A template is a hierarchy of nodes that has the same structure as the page to be created, but without any actual content.

It defines the page component used to render the page and the default content (primary top-level content). The content defines how it is rendered as CQ is content-centric.

Page Component (Top-Level Component)

The component to be used to render the page.

Page

A page is an 'instance' of a template.

A page has a hierarchy node of type cq:Page and a content node of type cq:PageContent. The property sling:resourceType of the content node points to the Page Component used for rendering the page.

Qus: CQ Widgets

Widgets are the basic elements used to implement a specific user function, often the editing of a piece of content; they include buttons, radio-boxes, dialogs, etc

Qus: PRIVILEGES in cq5?

The following privileges are available for selection when adding an access control entry (see the Security API for full details):

Privilege Name	Which controls the privilege to...
`jcr:read`	Retrieve a node and read its properties and their values.
`rep:write`	This is a jackrabbit specific aggregate privilege of jcr:write and jcr:nodeTypeManagement.

jcr:all	This is an aggregate privilege that contains all other predefined privileges.
Advanced	
crx:replicate	Perform replication of a node.
jcr:addChildNodes	Create child nodes of a node.
jcr:lifecycleManagement	Perform lifecycle operations on a node.
jcr:lockManagement	Lock and unlock a node; refresh a lock.
jcr:modifyAccessControl	Modify the access control policies of a node.
jcr:modifyProperties	Create, modify and remove the properties of a node.
jcr:namespaceManagement	Register, unregister and modify namespace definitions.
jcr:nodeTypeDefinitionManagement	Import node type definitions to the repository.
jcr:nodeTypeManagement	Add and remove mixin node types and change the primary node type of a node. This also includes any calls to Node.addNode and XML importing methods where the mixin or primary type of new node is explicitly specified.
jcr:readAccessControl	Read the access control policy of a node.
jcr:removeChildNodes	Remove child nodes of a node.
jcr:removeNode	Remove a node.
jcr:retentionManagement	Perform retention management operations on a node.
jcr:versionManagement	Perform versioning operations on a node.
jcr:workspaceManagement	The creation and deletion of workspaces through the JCR API.
jcr:write	This is an aggregate privilege that contains: - jcr:modifyProperties - jcr:addChildNodes - jcr:removeNode - jcr:removeChildNodes
rep:privilegeManagement	Register new privilege.

Qus: ORDERING LOCAL ACCESS CONTROL POLICIES

The order in the list indicates the order in which the policies are applied.

1. In the table of **Local Access Control Policies** select the required entry and drag it to the new position in the table.

2. The changes will be shown in both the tables for the **Local** and the **Effective Access Control Policies**.

REMOVING AN ACCESS CONTROL POLICY

1. In the table of **Local Access Control Policies** click the red icon (-) at the right of the entry.
2. The entry will be removed from both the tables for the **Local** and the **Effective Access Control Policies**.

Qus: Overlay / Override

Use of the Overlay component is , when you want to use the out-of-box component and you want to add some extra feature in it , so instead of changing the code from "libs/foundation/components/component-name.jsp" just follow the below process and use it. By using this we can use the out-of-box feature with our extra feature without changing behaviour of out-of-box component. So that other projects which are using it can't be affected.

Qus: Changing a CQ Component's Design Path

Usually, the default location used by the CQ Designs configuration works, however there are cases where you may want to set the designs to exist at a different location. For example, you may want multiple components to share a configuration or have all of the instances of a component on a site share the same configuration.

CQ provides the **EditConfig** object which can be used to configure how the edit dialog is loaded in CQ.

To configure your design path in Adobe CQ, first set the edit context path in the Edit Config object when in design mode:

```
String myComponentDesignPath = currentDesign.getPath() + "/jcr:content/mycomponent";
if (WCMMode.fromRequest(request) == WCMMode.DESIGN) {
 log.debug("Setting content path to {}", myComponentDesignPath);
 editContext.setContentPath(myComponentDesignPath);
}
```

Next, you will need to retrieve the resource you at the path you configured as a ValueMap, to be able to retrieve the properties you set.

```
Resource myComponentDesignRsrc =
resource.getResourceResolver().getResource(myComponentDesignPath);
```

```
if (myComponentDesignRsrc != null) {
 pageContext.setAttribute("myComponentDesign",
myComponentDesignRsrc.adaptTo(ValueMap.class));
}
```

Finally, once you have the design properties set as a page context attribute, you can retrieve the values in your component JSP.

```
${myComponentDesign.attribute}
```

The only unfortunate thing about this approach is that the *currentStyles* variable will no longer point to the correct design path.

Qus: How to use multi language translation in JSP in CQ / WEM

http://blogs.adobe.com/dekesmith/2012/10/21/internationalization-within-sling-and-cq/

Use Case: You want to display certain text in page based on page language

Prerequisite: http://sling.apache.org/site/internationalization-support-i18n.html

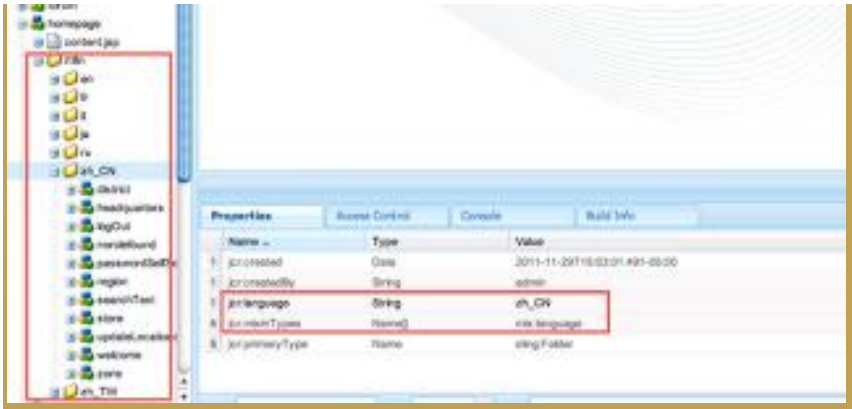
Solution: You can use i18n bundle approach for this

Suppose you want to display certain text for a component in different language based on page language or URL. Take an example of /apps/geometrixx/components/homepage component.

1) Create a i18n node under /apps/geometrixx/components/homepage of type sling:Folder
2) create a node of type sling:Folder under i18n of mixin type mix:language
3) Add a property called jcr:language and assign value of lang code or lang_COUNTRY

Structure would be like

```
/apps/myApp
+-- English (nt:folder, mix:language)
| +-- jcr:language = en
| +-- mx (sling:messageEntry)
| +-- sling:key = "msgXXX"
| +-- slign:message = "An Application Text"
+-- Deutsch (nt:folder, mix:language)
+-- jcr:language = de
+-- mx (sling:messageEntry)
+-- sling:key = "msgXXX"
+-- slign:message = "Ein Anwendungstext"
```

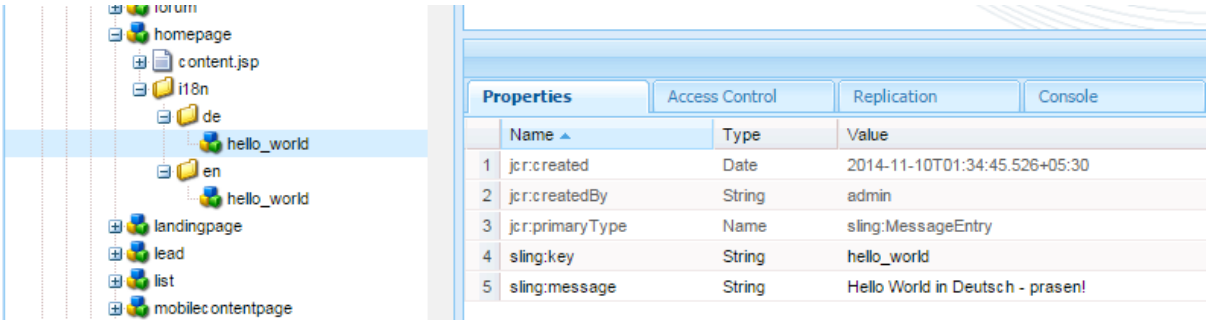


4) Then on jsp page use following code

<cq:setContentBundle/>
<%
Locale pageLocale = currentPage.getLanguage(false);
//If above bool is set to true. CQ looks in to page path rather than jcr:language property.
ResourceBundle resourceBundle = slingRequest.getResourceBundle(pageLocale);
I18n i18n = new I18n(resourceBundle);
%>
I am printing using I18n <%=i18n.get("updateLocations") %>
I am printing using fmt:message now <fmt:message key="updateLocations"/>
I found page locale is <%=pageLocale.toString() %>

You can also test language using grid under http://localhost:4502/libs/cq/i18n/translator.html
More Info: http://labs.sixdimensions.com/blog/dklco/2012-05-25/adobe-cq5-translator

To change what languages are shown in the translator's grid, create a multi-value string property /etc/languages/languages (node structure must be created), containing the iso language codes (e.g. ["de", "fr", …]). The default is hard coded in the translator and maps the ootb CQ languages: ['de', 'fr', 'it', 'es', 'ja', 'zh-cn']



Now in the content page (jsp) of the rendering component add the following code.
<%@ page import="org.apache.sling.api.resource.Resource"%>
<%@ taglib prefix="sling" uri="http://sling.apache.org/taglibs/sling/1.0" %>
<sling:defineObjects />
<%= slingRequest.getResourceBundle(slingRequest.getLocale()).getString("hello_world") %>

Qus: How to connect external DB from CQ5?

Access an external SQL database to so that your CQ applications can interact with the data:

Create or obtain an OSGi bundle that that exports the JDBC driver package.

Configure a JDBC data source pool provider.

Obtain a data source object and create the connection in your code.

BUNDLING THE JDBC DATABASE DRIVER

Some database vendors provide JDBC drivers in an OSGi bundle, for example MySQL. If the JDBC driver for your database is not available as an OSGi bundle, obtain the driver JAR and wrap it in an OSGi bundle. The bundle must export the packages that are required for interacting with the database server. The bundle must also import the packages that it references.

The following example uses the Bundle plugin for Maven to wrap the HSQLDB driver in an OSGi bundle. The POM instructs the plugin to embed the hsqldb.jar file that is identified as a dependency. All org.hsqldb packages are exported.

The plugin automatically determines which packages to import and lists them in the MANIFEST.MF file of the bundle. If any of the packages are not available on the CQ server, the bundle will not start upon installing. Two possible solutions are as follows:

Indicate in the POM that the packages are optional. Use this solution when the JDBC connection does not actually require the package members. Use the Import-Package element to indicate optional packages as in the following example:

```
<Import-Package>org.jboss.*;resolution:=optional,*</Import-Package>
```

Wrap the JAR files that contain the packages in an OSGi bundle that exports the packages, and deploy the bundle. Use this solution when the package members are required during code execution.

Knowledge of the source code enables you to decide which solution to use. You can also try either solution and perform testing to validate the solution.

POM that bundles hsqldb.jar

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.adobe.example.myapp</groupId>
 <artifactId>hsqldb-jdbc-driver-bundle</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <name>wrapper-bundle-hsqldb-driver</name>
 <url>www.adobe.com</url>
 <description>Exports the HSQL JDBC driver</description>
 <packaging>bundle</packaging>
 <properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 </properties>
 <build>
  <plugins>
```

```
    <plugin>
     <groupId>org.apache.felix</groupId>
     <artifactId>maven-bundle-plugin</artifactId>
     <version>1.4.3</version>
     <extensions>true</extensions>
     <configuration>
      <instructions>
        <Embed-Dependency>*</Embed-Dependency>
        <_exportcontents>org.hsqldb.*</_exportcontents>
      </instructions>
     </configuration>
    </plugin>
   </plugins>
  </build>
  <dependencies>
   <dependency>
    <groupId>hsqldb</groupId>
    <artifactId>hsqldb</artifactId>
    <version>2.2.9</version>
   </dependency>
  </dependencies>
</project>
```

The following links open the download pages for some popular database products:

Microsoft SQL Server

Oracle

IBM DB2

CONFIGURING THE JDBC CONNECTION POOL SERVICE

Add a configuration for the JDBC Connections Pool service that uses the JDBC driver to create data source objects. Your application code uses this service to obtain the object and connect to the database.

JDBC Connections Pool (`com.day.commons.datasource.jdbcpool.JdbcPoolService`) is a factory service. If you require connections that use different properties, for example read-only or read/write access, create multiple configurations.

To create a connection, use the Configuration tab of the Web Console or create a `sling:OsgiConfig` node in the repository. (See Adding a New Configuration to the Repository.)

The following properties are available to configure a pooled connection service. The property names are listed as they appear in the Web Console. The corresponding name for a `sling:OsgiConfig` node appears in parentheses. Example values are shown for an HSQLDB server and a database that has an alias of `mydb`:

JDBC Driver Class (`jdbc.driver.class`): The Java class to use that implements the java.sql.Driver interface, for example `org.hsqldb.jdbc.JDBCDriver`. The data type is `String`.

JDBC Connection URI (`jdbc.connection.uri`): The URL of the database to use to create the connection, for example `jdbc:hsqldb:hsql//10.36.79.223:9001/mydb`. The format of the URL must be valid for use with the getConnection method of the java.sql.DriverManager class. The data type is `String`.

Username (`jdbc.username`): The user name to use to authenticate with the database server. The data type is`String`.

Password (`jdbc.password`): The password to use for authentication of the user. The data type is `String`.

Validation Query (`jdbc.validation.query`): The SQL statement to use to verify that the connection is successful, for example `select 1 from INFORMATION_SCHEMA.SYSTEM_USERS`. The data type is `String`.

Readonly By Default (default.readonly): Select this option when you want the connection to provide read-only access. The data type is `Boolean`.

Autocommit By Default (`default.autocommit`): Select this option to create separate transactions for each SQL command that is sent to the database, and each transaction is automatically committed. Do not select this option when you are committing transactions explicitly in your code. The data type is `Boolean`.

Pool Size (`pool.size`): The number of simultaneous connections to be made available to the database. The data type is `Long`.

Pool wait (`pool.max.wait.msec`): The amount of time before a connection request times out. The data type is`Long`.

Datasource Name (`datasource.name`): The name of this data source. The data type is `String`.

Additional Service Properties (`datasource.svc.properties`): A set of name/value pairs that you want to append to the connection URL. The data type is `String[]`.

The JDBC Connections Pool service is a factory. Therefore, if you use a `sling:OsgiConfig` node to configure the connection service, the name of the node must include the factory service PID followed by `-alias`. The alias that you use must be unique for all configuration nodess for that PID. An example node name is `com.day.commons.datasource.jdbcpool.JdbcPoolService-myhsqldbpool`.

Properties | Access Control | Replication | Console

	Name ▲	Type	Value
1	datasource.name	String	hsqldbds
2	default.autocommit	Boolean	true
3	default.readonly	Boolean	false
4	jcr:created	Date	2012-08-09T14:14:26.436-04:00
5	jcr:createdBy	String	admin
6	jcr:primaryType	Name	sling:OsgiConfig
7	jdbc.connection.uri	String	jdbc:hsqldb:hsql://10.36.79.223:9001/mydb
8	jdbc.driver.class	String	org.hsqldb.jdbc.JDBCDriver
9	jdbc.password	String	
10	jdbc.username	String	SA
11	pool.max.wait.msec	Long	1000
12	pool.size	Long	21

CONNECTING TO THE DATABASE

In your Java code, use the DataSourcePool service to obtain a `javax.sql.DataSource` object for the configuration that you created. The DataSourcePool service provides the `getDataSource` method that returns a `DataSource` object for a given data source name. As the method argument, use the value of the Datasource Name (or`datasource.name`) property that you specified for the JDBC Connections Pool configuration.

The following example JSP code obtains an instance of the hsqldbds data source, executes a simple SQL query, and displays the number of results that are returned.

JSP that performs a database lookup

```
<%@include file="/libs/foundation/global.jsp"%><%
%><%@page session="false"%><%
%><%@ page import="com.day.commons.datasource.poolservice.DataSourcePool" %><%
%><%@ page import="javax.sql.DataSource" %><%
%><%@ page import="java.sql.Connection" %><%
%><%@ page import="java.sql.SQLException" %><%
%><%@ page import="java.sql.Statement" %><%
%><%@ page import="java.sql.ResultSet"%><%
%><html>
<cq:include script="head.jsp"/>
<body>
<%DataSourcePool dspService = sling.getService(DataSourcePool.class);
  try {
     DataSource ds = (DataSource) dspService.getDataSource("hsqldbds");
     if(ds != null) {
         %><p>Obtained the datasource!</p><%
         %><%final Connection connection = ds.getConnection();
          final Statement statement = connection.createStatement();
          final ResultSet resultSet = statement.executeQuery("SELECT * from
INFORMATION_SCHEMA.SYSTEM_USERS");
          int r=0;
          while(resultSet.next()){
             r=r+1;
          }
          resultSet.close();
          %><p>Number of results: <%=r%></p><%
      }
   }catch (Exception e) {
        %><p>error! <%=e.getMessage()%></p><%
    }
%></body>
</html>
```

Qus: What is Segmentation?

Segmentation

Segmentation is a key consideration when creating a campaign. In most cases, you will need to have segments already defined before starting your campaign.

Site visitors have different interests and objectives when they come to a site. Understanding these goals and fulfilling the expectations is an important success factor for online marketing.

Segmentation helps to achieve this by analyzing and characterizing a visitor's:

- activity on the website
- profile
- activity on other websites

Content can then be specifically targeted to the visitor's needs and interests, depending on the segment(s) they match.

When discussing segmentation, the following terminology is used:

Visitor

A visitor is a person visiting a website. That person's visit typically starts from a referring page, then moves on to one or more page views on your own website. A behavioral profile can be created from the details of that person's visit.

User

A user is a visitor who registers with the website to receive an account profile. To generate their profile they provide additional identification, such as an email address and gender, amongst others. Additional information can also be collected, including community activity and purchase patterns, again amongst others. Based on the information provided in the profile, a demographic profile can be created.

Trait

A trait is a characteristic or property of a visitor that can be used to determine membership in a specific segment.

Segment

A segment is a collection of visitors that share certain traits. Segments should be distinctive, with a minimum of overlap with other segments.

Behavioral Traits

Behavioral traits are those that relate to a visitor's behavior on the website. These include:

- Interest within your website; including pages visited and products bought.
- Interest on the referring website; including search terms used, or adverts clicked on.
- Interest on other sites; determined using tools such as Spyjax.
- Visitor loyality; duration of the visit, frequency of visits.

Demographic Traits

These are selected population characteristics including:

- Age
- Income
- Family size
- Marital status
- Gender
- Location

Derived Traits

Some demographic traits are hard to determine without registration, but can be derived by combining behavioral and demographic traits.

For example, combining the referring URL (as a behavioral trait) with demographic data (acquired from tools such asGoogle Ad Planner) allows site owners to derive demographics traits of their visitors.

Subsegment

A segment can be subdivided into several subsegments. This is done by defining additional traits.

Teaser Page

A teaser page is directed at a specific audience. It contains re-usable content that can be used in the teaser paragraph.

Campaign

A campaign is a collection of teaser pages and e-mail marketing pages, such as newsletters or invitations. Typically a campaign runs for a limited period and is superceded by another campaign.

Teaser Paragraph

This is a paragraph that pulls content from another page dependent on a selection strategy. This selection strategy can take segments and campaigns into consideration.

List

A list is extracted from a segment of registered users. For example, the location used to steer the contents of the teaser paragraph.

> **NOTE**
>
> Please see Segmentation for further information on segments in CQ.

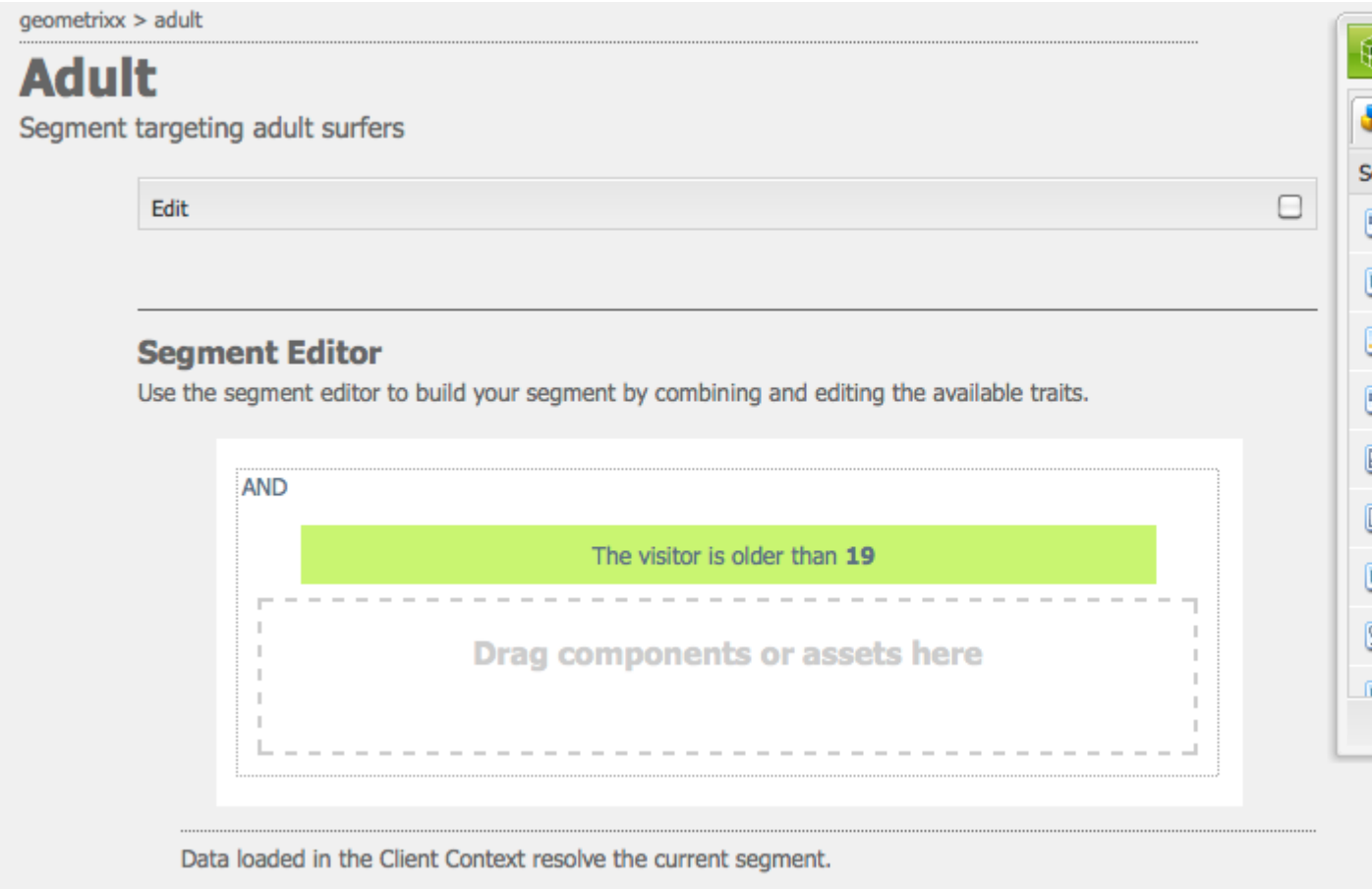
SEGMENTATION IN CQ

Depending on the information you have already collected about your site visitors and the goals you want to achieve, you will need to define the segments and strategies needed for your targeted content.

These segments are then used to provide a visitor with specifically targeted content. This content is maintained in theCampaigns section of the website. Teaser pages defined here can be included as teaser paragraphs on any page and define which visitor segment the specialized content is applicable for.

CQ allows you to easily create and update segments, teasers, and campaigns. It also allows you to verify the results of your definitions.

The **Segment Editor** allows you to easily define a segment:



You can **Edit** each segment to specify a **Title**, **Description** and **Boost** factor. Using the sidekick you can add **AND** and**OR** containers to define the **Segment Logic**, then add the required **Segment Traits** to define the selection criteria.

Boost Factor

Each segment has a **Boost** parameter that is used as a weighting factor; a higher number indicates that the segment will be selected in preference to a segment with a lower number.

Segment Logic

The following logic containers are available out-of-the-box and allow you to construct the logic of your segment selection. They can be dragged from the sidekick to the editor:

AND Container	The boolean AND operator.
OR Container	The boolean OR operator.

Segment Traits

The following segment traits are available out-of-the-box; they can be dragged from the sidekick to the editor:

IP Range	Defines a range of IP addresses that the visitor can have.
Page Hits	How often the page has been requested.
Page Property	Any property of the visited page.
Referral Keywords	Keywords to match with information from the referring website.
Script	Javascript expression to be evaluated.
Segment Reference	Reference to another segment definition.
Tag Cloud	Tags to be matched with those from the pages visited.
User Age	As taken from the user profile.
User Property	Any other information that is available in the user profile.

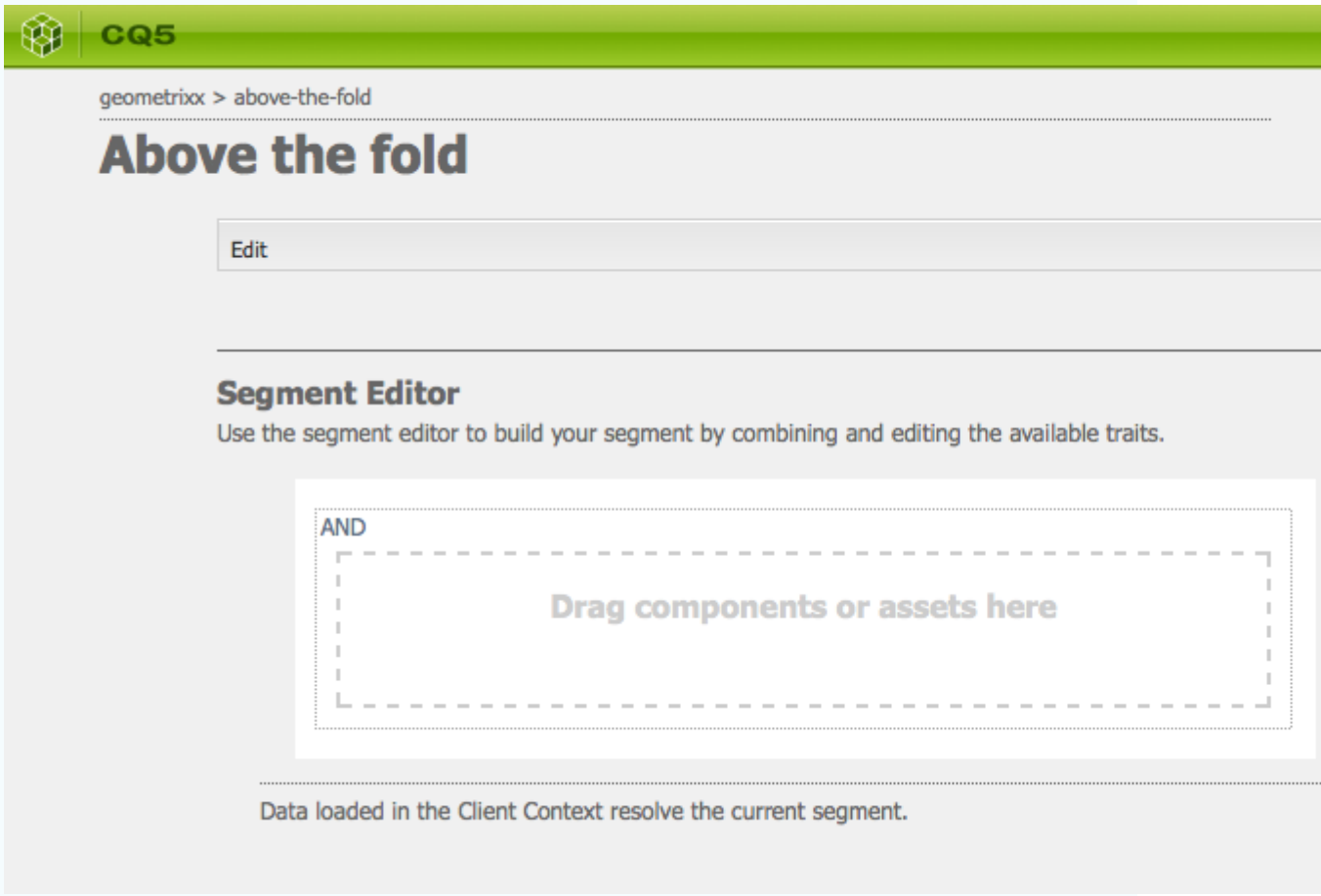
You can combine these traits using the boolean operators OR and AND (see Defining a New Segment) to define the exact scenario for selecting this segment.

When the entire statement evaluates to true then this segment has resolved. In the event of multiple segments being applicable, then the **Boost** factor is also used.

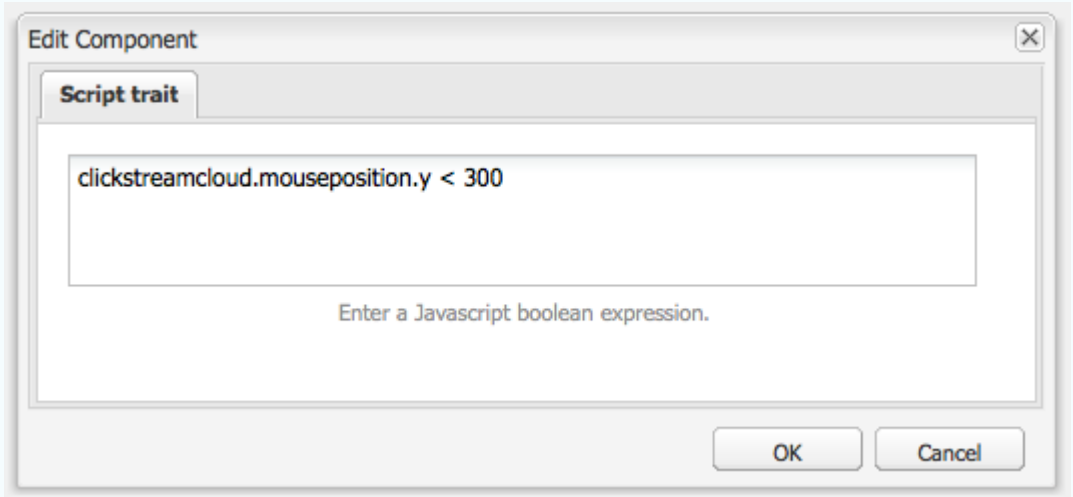
DEFINING A NEW SEGMENT

To define your new segment:

1. Open the **Tools** console.
2. Click on the **Segmentation** page in the left pane, and navigate to the required location.
3. Create a new page using the **Segment** template.
4. Open the new page to see the segment editor:

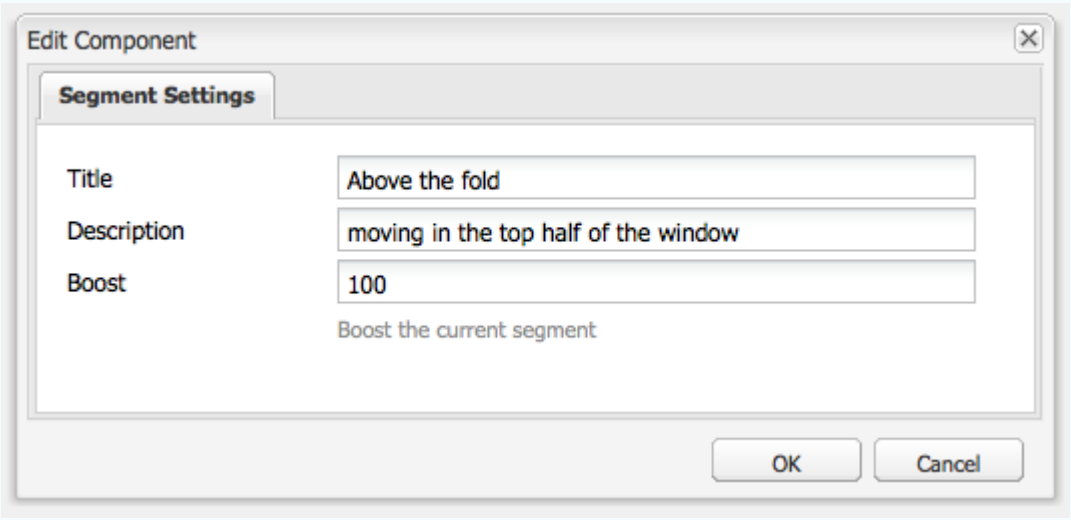


5. Use either the sidekick or the context menu (usually right mouse button click, then select **New...** to open the Insert New Component window) to find the segment trait you need. Then drag it to the **Segment Editor** it will appear in the default **AND** container.
6. Double-click on the new trait to edit the specific parameters; for example the mouse position:



7. Click **OK** to save your definition:

8. You can **Edit** the segment definition to give it a **Title**, **Description** and **Boost** factor:



9. Add more traits if required. You can formulate boolean expressions using the **AND Container** and **OR Container** components found under **Segment Logic**. With the segment editor you can delete traits or containers not needed anymore, or drag them to new positions within the statement.

USING AND AND OR CONTAINERS

You can construct complex segments in CQ. It helps to be aware of a few basic points:

- The top level of the definition is always the AND container that is initially created; this cannot be changed, but does not have an effect on the rest of your segment definition.
- Ensure that the nesting of your container makes sense. The containers can be viewed as the brackets of your boolean expression.

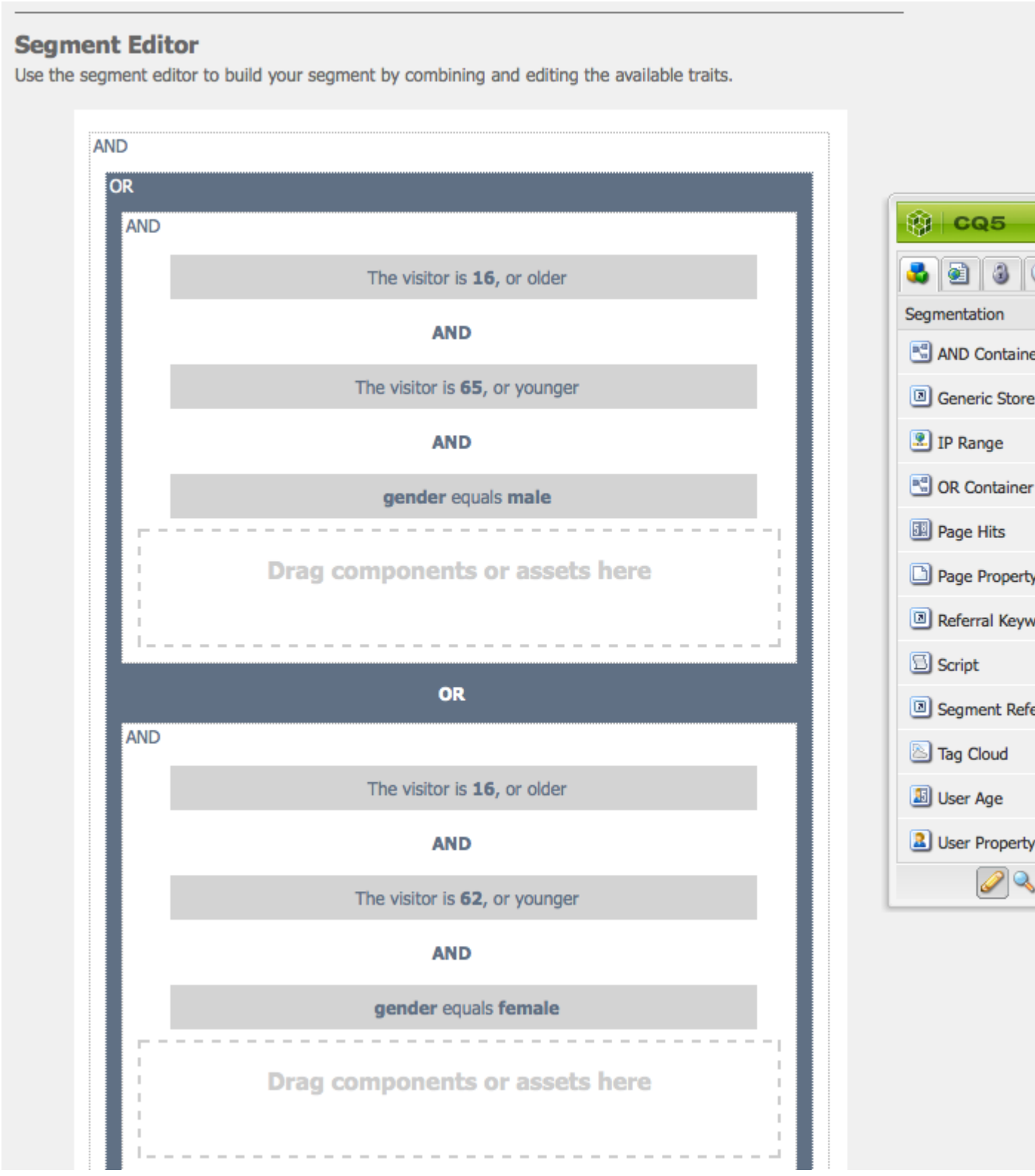
The following example is used to select visitors who are either:

Male and between the ages of 16 and 65

OR

Female and between the ages of 16 and 62

As the main operator is OR you need to start with an **OR Container**. Within this you have 2 AND statements, for each of these you need an **AND Container**, into which you can add the individual traits.



TESTING THE APPLICATION OF A SEGMENT

Once the segment has been defined, potential results can be tested with the help of the **Client Context**:

1. Select the segment to be tested.
2. Press **Ctrl-Alt-C** to open the **Client Context**, which shows the data that has been collected. For testing purposes you can **Edit** certain values, or **Load** another profile to see the impact there.
3. Depending on the traits defined, the data available for the current page may or may not match the segment definition. The status of the match is shown underneath the definition.

For example, a simple segment definition can be based on the age and gender of the user. Loading a specific profile shows that the segment is successfully resolved:

Or not:

> **NOTE**
>
> All traits are resolved immediately, though most only change on page reload. Changes to mouse position are visible immediately, so useful for testing purposes.

Such tests can also be performed on content pages and in combination with **Teaser** components.

Mouseover on a teaser paragraph will show the segments applied, whether they currently resolve and therefore, why the current teaser instance has been selected:

Using Your Segment

Segments are currently used within Campaigns. They are used to steer the actual content seen by specific target audiences.

Qus: Campaign Management

Campaign Management

You are reading the **Adobe Experience Manager 5.6.1** version of *Campaign Management*.
This documentation is also available for the following versions: Adobe Experience Manager 6.0 AEM 5.6 CQ 5.5 CQ 5.4 CQ 5.3

Campaign management provides digital marketers the opportunity to deliver personalized content and so create dedicated experiences for visitors.

It allows you to orchestrate your marketing campaigns across the web, email and mobile services and so engage your visitors. You can create content, segment visitors, push and promote targeted content for specific user profiles and manage campaigns across multiple channels.

Campaign management is made up of various elements:

- **Brands**
  In CQ, brands are the top level unit and form a collection of **Campaigns**.
- **Campaigns**
  A campaign is a collection of individual **Experiences**.
- **Experiences**
  The focused content forms the various experiences, presented to the visitor at **Touchpoints**. There are several types of experience available:
  - **Teasers**
    Teaser Pages / Paragraphs are used to steer specific visitor **Segments** to content that is focused on their interests.
    Teaser pages can:
    - present a range of options for the visitor to choose from
    - show only one teaser paragraph that is based on the specific visitor segment; for example, the teaser paragraph shown may be dependent on the age of the visitor.

    Typically a teaser page is a temporary action that will last for a specific period of time, until it is replaced by the next teaser page.
  - **Newsletters**
    E-mail Communications are used to engage users and encourage them to visit your web site. These usually take the form of a newsletter, sent to your **Leads** (which are usually grouped into **Lists**).
  - **Test&Target**
    This allows integration with Adobe's Test&Target which gives marketers a conversion website optimization

tool with the necessary capabilities to continually make their online content and offers more relevant to their customers—yielding greater conversion. Test&Target provides an intuitive interface for designing and executing tests, creating audience segments and targeting content—all from a single application.

- **Touchpoints**
  These are the points of contact between the visitor and your campaign. The touchpoints are connected to the experiences that you have created.
  For example, for teasers it is the content page where the teaser paragraph is located, for a newsletter it is the mailing list.
- **Leads**
  The information that you have collected about your visitors and how to contact them forms the basis for your leads.
- **Lists**
  Leads are usually grouped into lists so that you can take collective action on them.
- **Segments**
  Site visitors have different interests and objectives when they come to a site. Analyzing this according to factors such as activity on the website, profile information registered and activity on other websites, helps you to define segments. Content can then be specifically targeted to the visitor's needs and interests according to the segment(s) they match.
- **MCM**
  The Marketing Campaign Manager (MCM) is a console that allows you to access all the functionality you need to create and control your campaigns, brands, experiences, touchpoints, leads, lists, segments and reports.
  It can be accessed from various locations (labelled as **Campaigns**), or with, for example, the URL:
  `http://localhost:4502/libs/mcm/content/admin.html`

> **NOTE**
>
> Brands were introduced with CQ 5.5, so campaigns created before then are not connected to a brand; such campaigns are fully supported though.

Marketing Campaign Manager

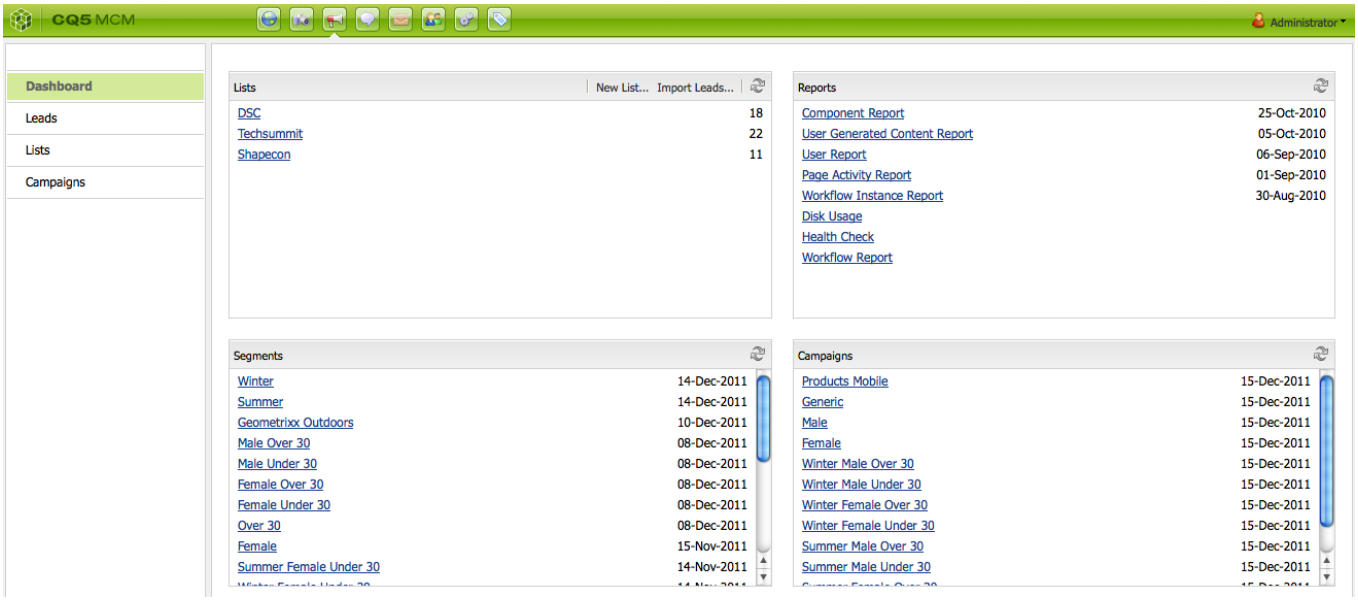
In CQ, the Marketing Campaign Manager (MCM) is a console that helps you manage multi-channel campaigns. With this marketing automation software you can manage all your brands, campaigns and experiences together with the related segments, lists, leads, and reports.

MCM can be accessed from various locations in CQ; for example, the Welcome screen, using the Campaigns icon or with the URL:

`http://<hostname>:<port-nr>/libs/mcm/content/admin.html`

For example:

`http://localhost:4502/libs/mcm/content/admin.html`

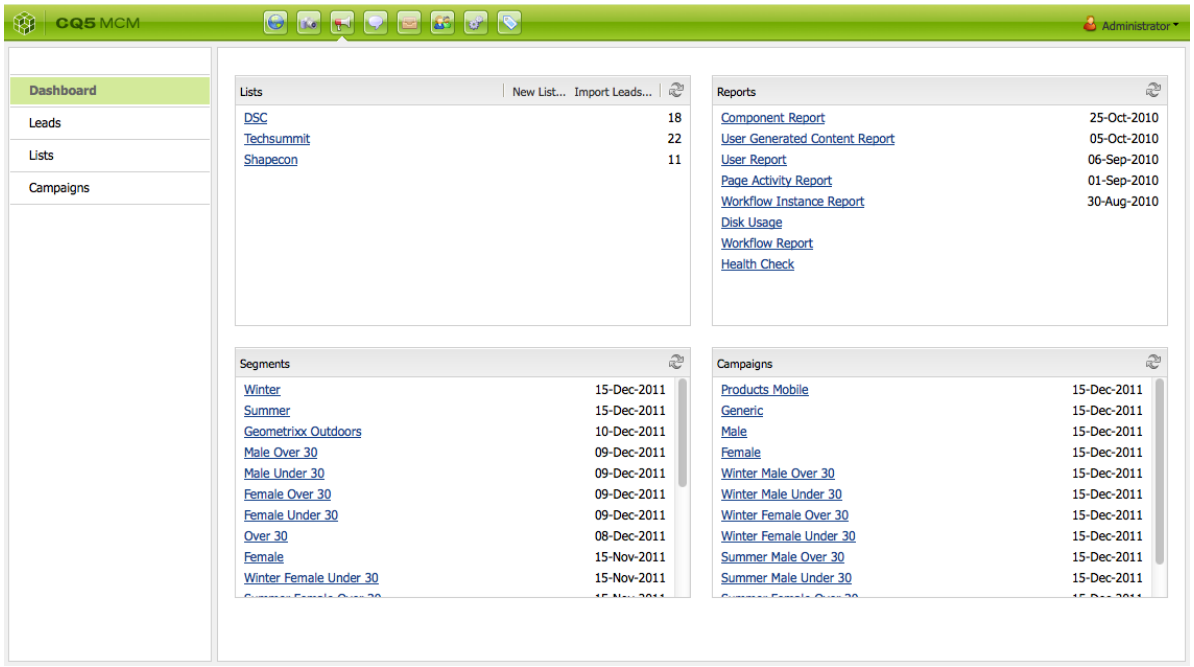


From the MCM you can access:

- **Dashboard**
  This is divided into four panes:
  - Lists
    This pane shows the lists you have already created, together with the number of leads in that list. From this pane you can create a new list directly or import leads to create a new list.
    Selecting a specific list will take you to the Lists section showing details for your list.
  - Segments
    This pane show the segments that you have defined. Segments let you characterize a collection of visitors that share certain traits.
    Selecting a specific segment will open the segment definition page.
  - Reports
    CQ provides different reports to help you analyze and monitor the state of your instance. This MCM pane lists the reports.
    Selecting a report will open the report page.
  - Campaigns
    This pane lists your campaign experiences such as newsletters and teasers.
- **Leads**
  Here you can manage your leads. You can create or import leads, edit specific details for individual leads or delete when no longer needed. You can also put leads in different groups, called Lists.
- **Lists**
  Here you can manage your lists (of leads).
- **Campaigns**
  Here you can manage your Brands, Campaigns and Experiences.

DASHBOARD

The dashboard shows four panes that provide you with an overview of your lists (of leads), segments, reports and campaigns. Access to basic functionality for these is also available here.

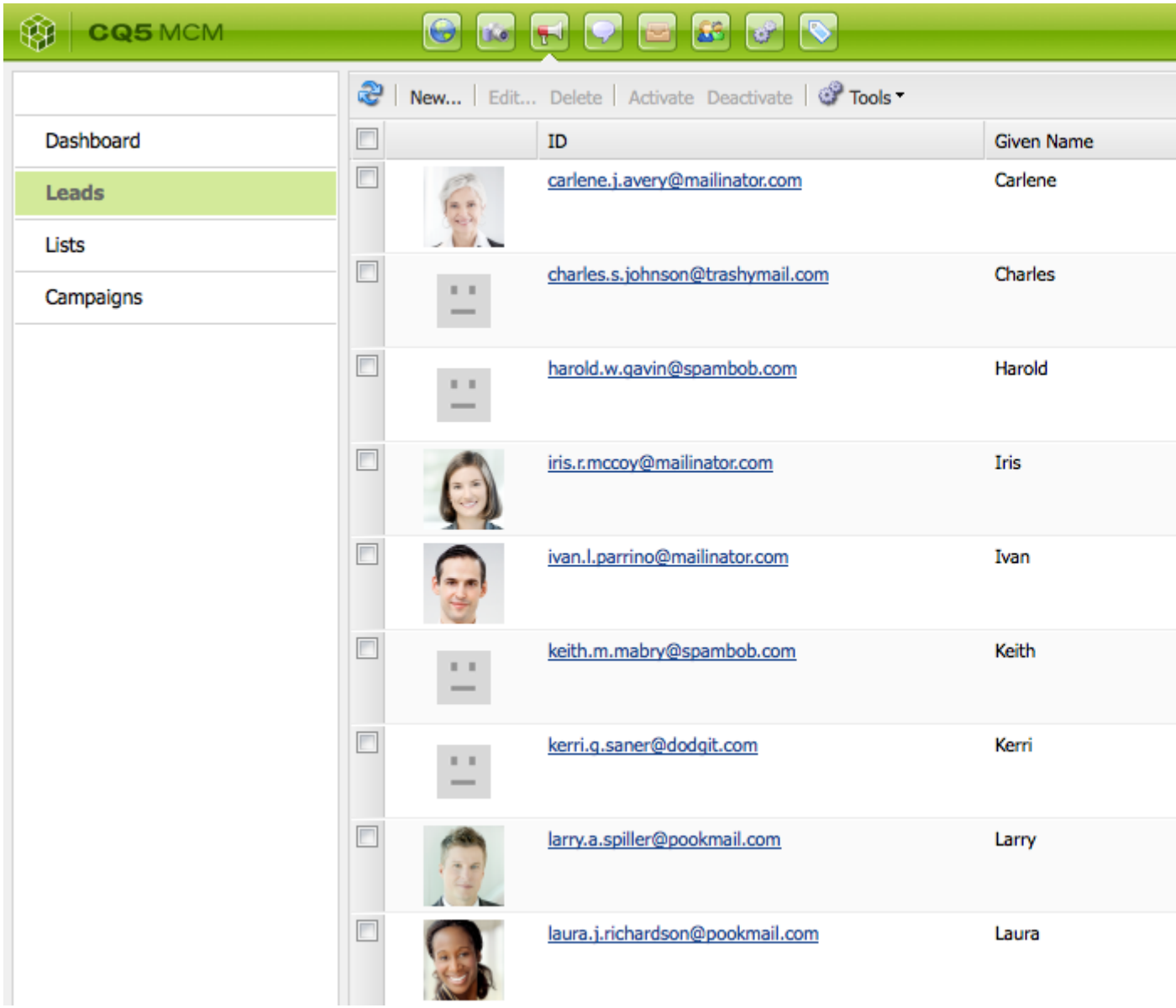


LEADS

> **NOTE**
>
> See Working with Leads for detailed information about specific tasks.

In CQ MCM, you can organize and add leads by either entering them manually or importing a comma-separated list; for example, a mailing list. Additional ways to generate leads are from newsletter sign-ups or community sign-ups (if configured, these can trigger a workflow that populates leads). Leads are usually categorized and put into a list so that later you can perform actions on the whole list; for example, sending out a custom email to a certain list.

Under **Leads** in the left pane you can create, import, edit and delete your leads, then activate or deactivate as required. You can add a lead to a list, or see which lists it already belongs to.

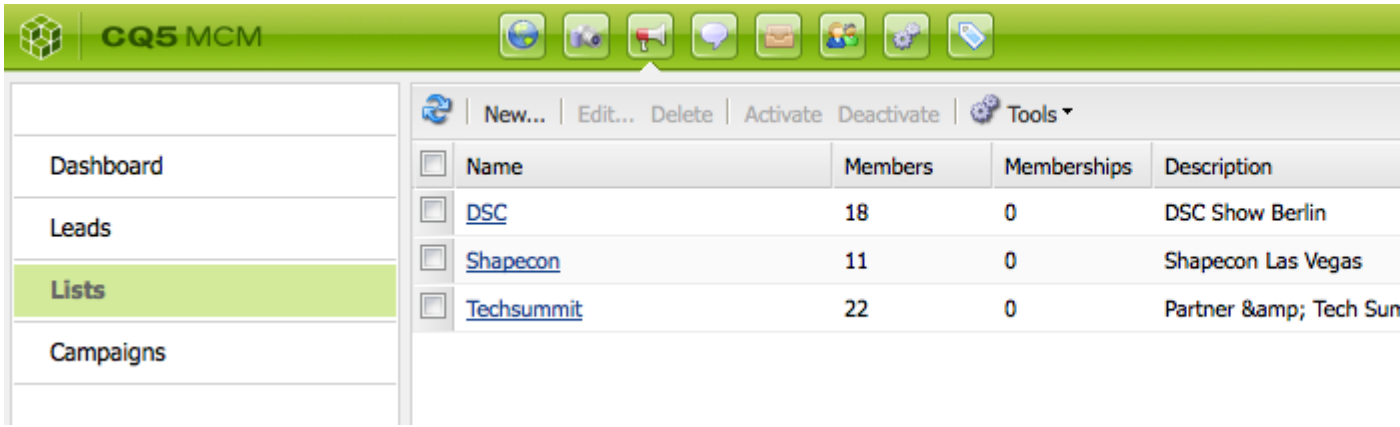


LISTS

> **NOTE**
>
> See Working with Lists for detailed information about specific tasks.

Lists let you organize your leads into groups. With lists, you can target your marketing campaigns to a select group of people; for example, you can send a targeted newsletter to a list.

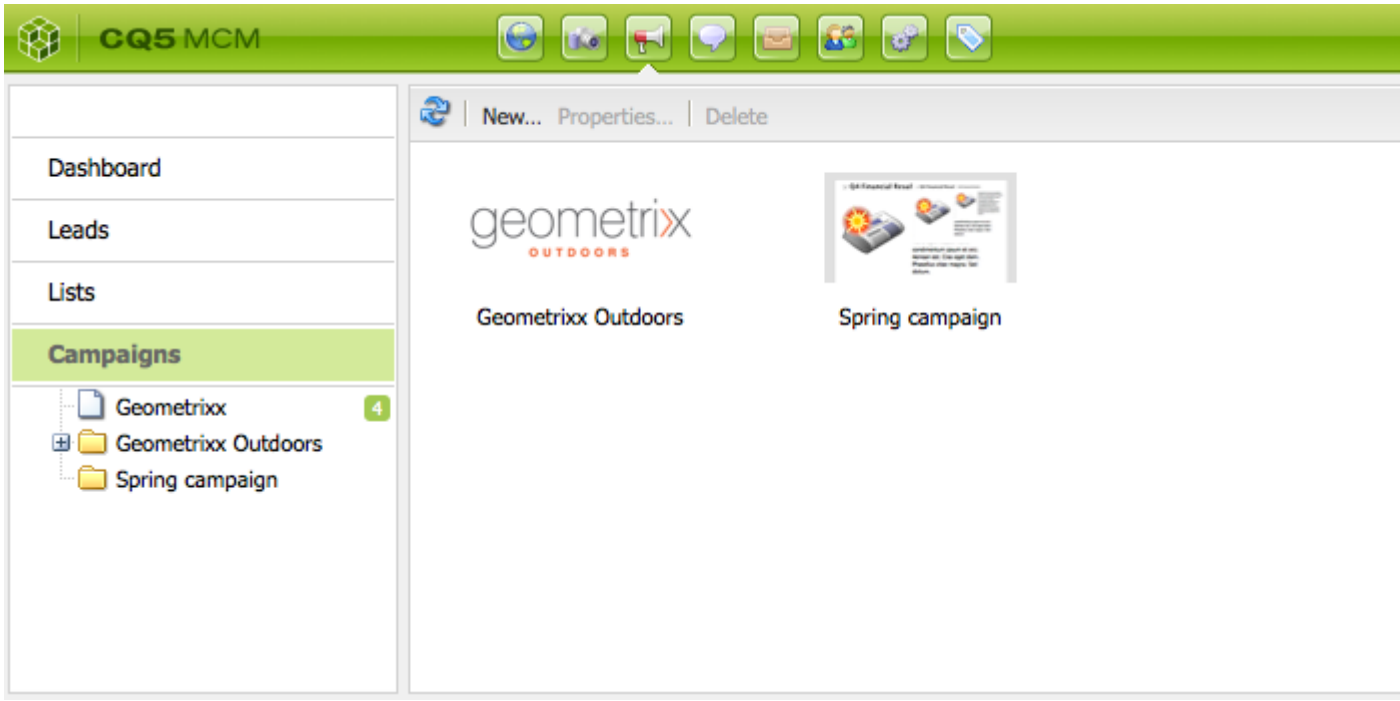
Under **Lists**, you can manage your lists by creating, importing, editing, merging and deleting lists which you can then activate or deactivate as required. You can also view the leads within that list, see if the list is a member of another list or view the description.



CAMPAIGNS

> **NOTE**
>
> See Teasers and Strategies, Setting up your Campaign and Newsletters for detailed information about specific tasks.

To access existing campaigns, in the MCM click **Campaigns**.

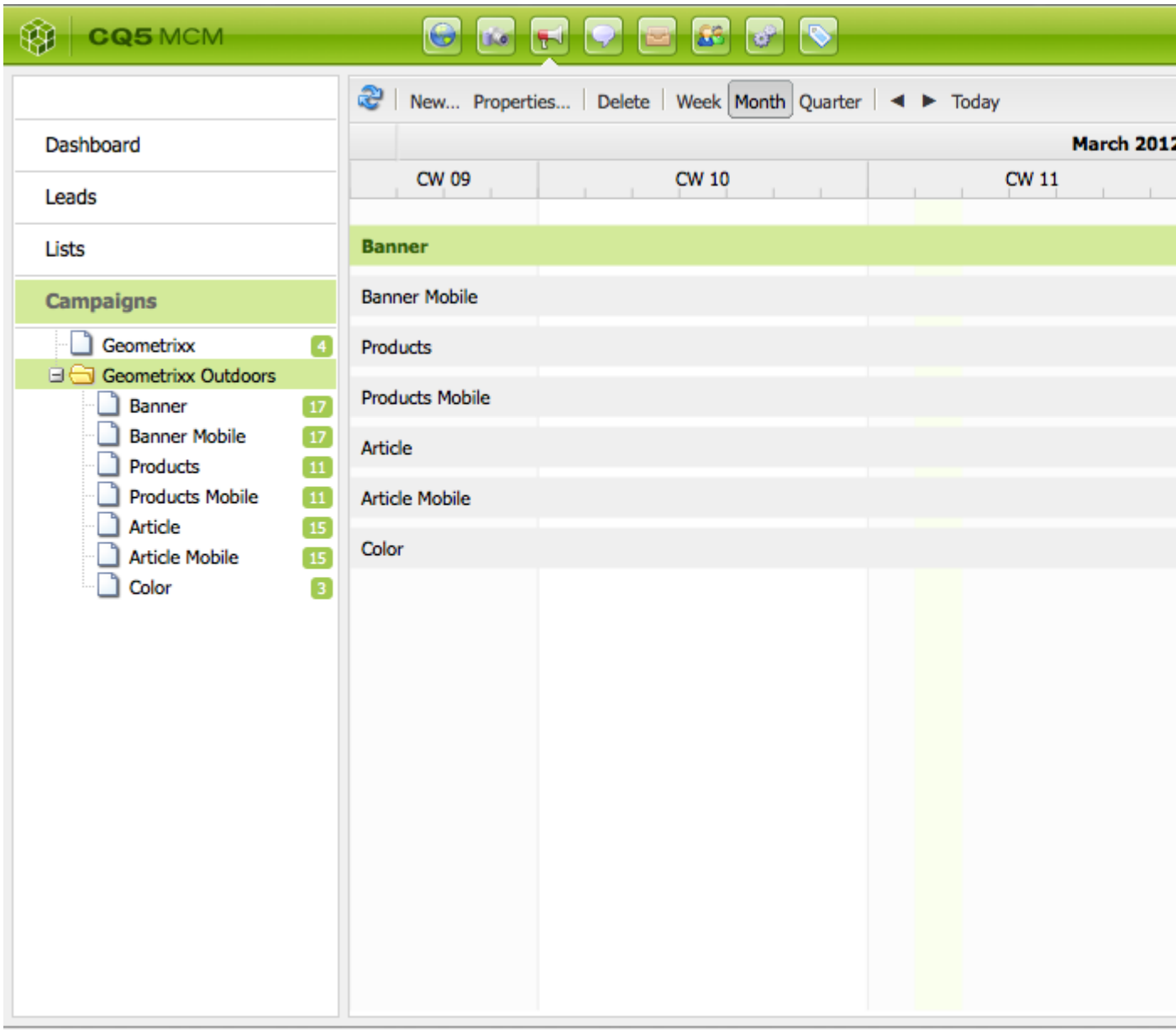


- **In the left pane**:
  There is a list of all brands and campaigns.
  A single click on a brand will both:
  - expand the list to show all related campaigns in the left pane; this list also shows the number of experiences that exist for each campaign.
  - open the brand overview in the right pane.

> **NOTE**
>
> Brands were introduced with CQ 5.5, so campaigns created before then will also be listed - without a brand.
>
> A single click on such entries (e.g. Geometrixx) will open the campaign overview in the right pane.

- **In the right pane**:
  Icons are shown for each brand (historical campaigns will not be shown).
  You can double-click on these to open the brand overview.

Brand Overview



From here you can:

- See the number of campaigns and experiences (number shown in the left pane) that exist for this brand.
- Create a **New...** campaign for this brand.
- Change the timespan being viewed; select **Week**, **Month** or **Quarter**, use the arrows to select specific periods or return to **Today**.
- Select a campaign (in the right pane) to:
  - Edit the **Properties...**
  - **Delete** the campaign.
- Open the campaign overview (double-click a campaign in the right pane, or single click in the left pane).

Campaign Overview

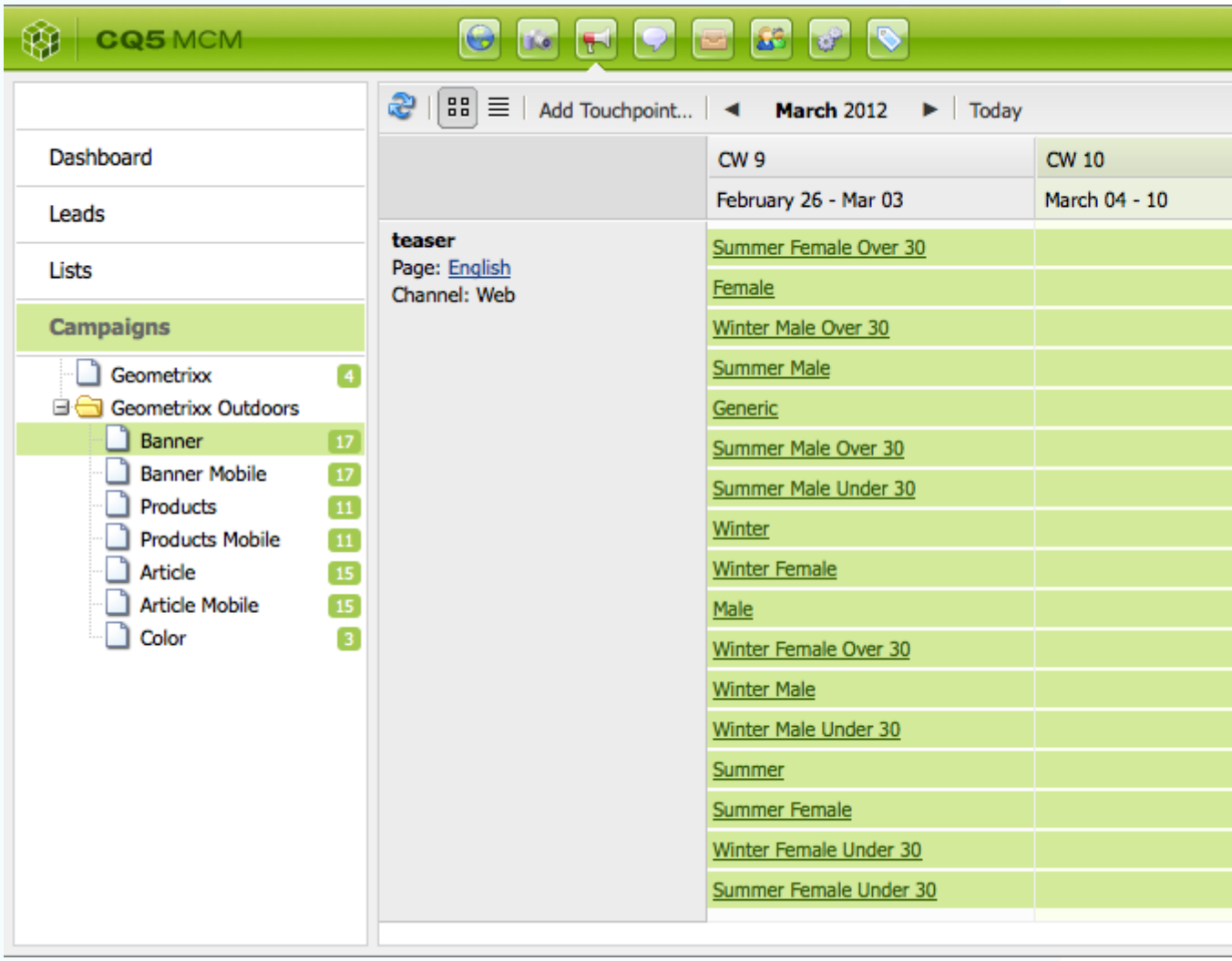
For the individual campaigns there are two views available:

1. **Calendar View**

   Use the icon:

   This presents a list of all touchpoints (grey) with a horizontal timeframe of the experiences (green) connected to that touchpoint:



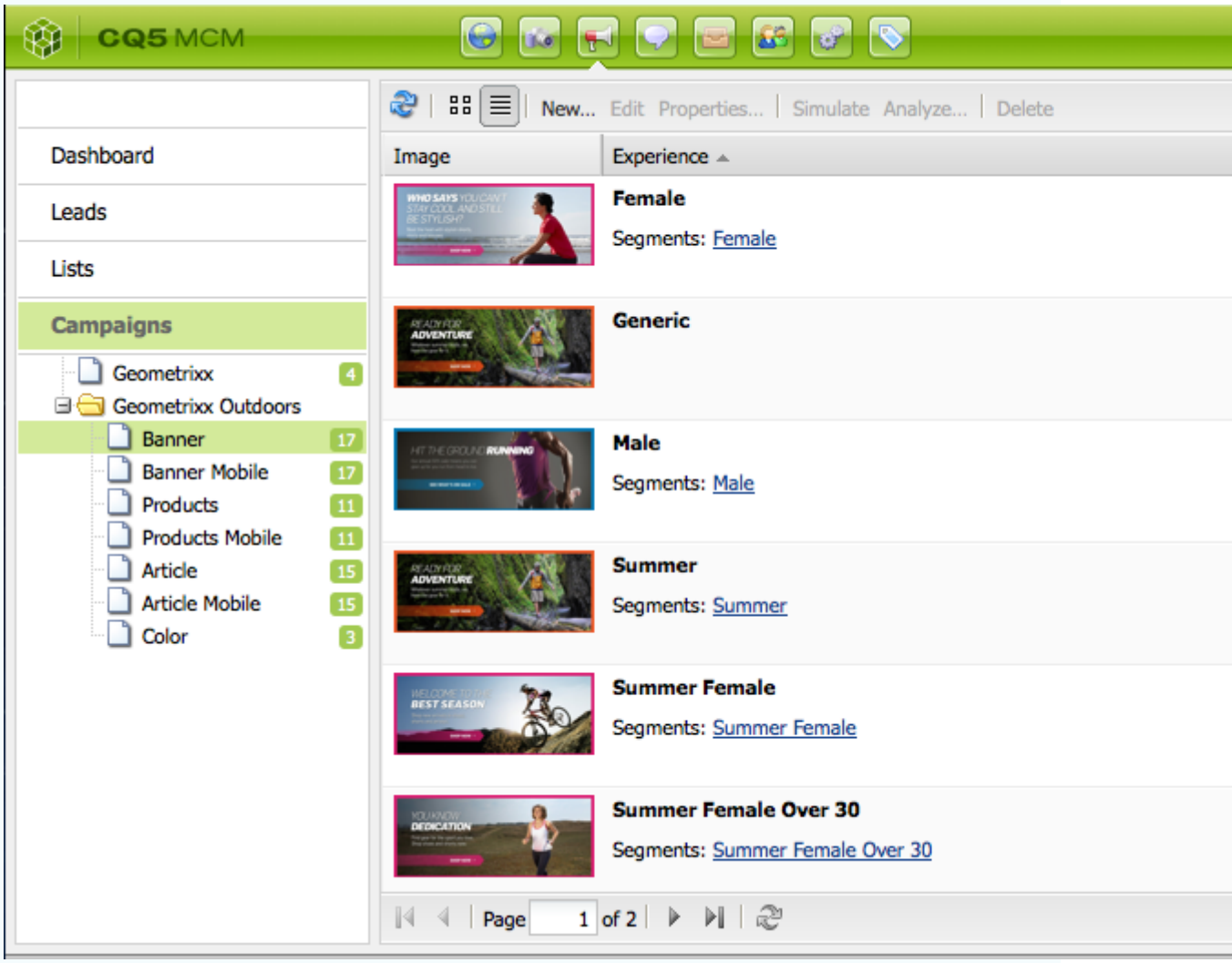
From here you can:

- Change the timespan you are viewing by using the arrows, or return to **Today**.
- Use **Add Touchpoint...** to add a new touchpoint for an existing experience.
- Click on a teaser (in the right pane) to set the **On Time** and **Off Time**.

2. **List View**

Use the icon:

This lists all experiences (e.g. teasers and newsletters) for the selected campaign:

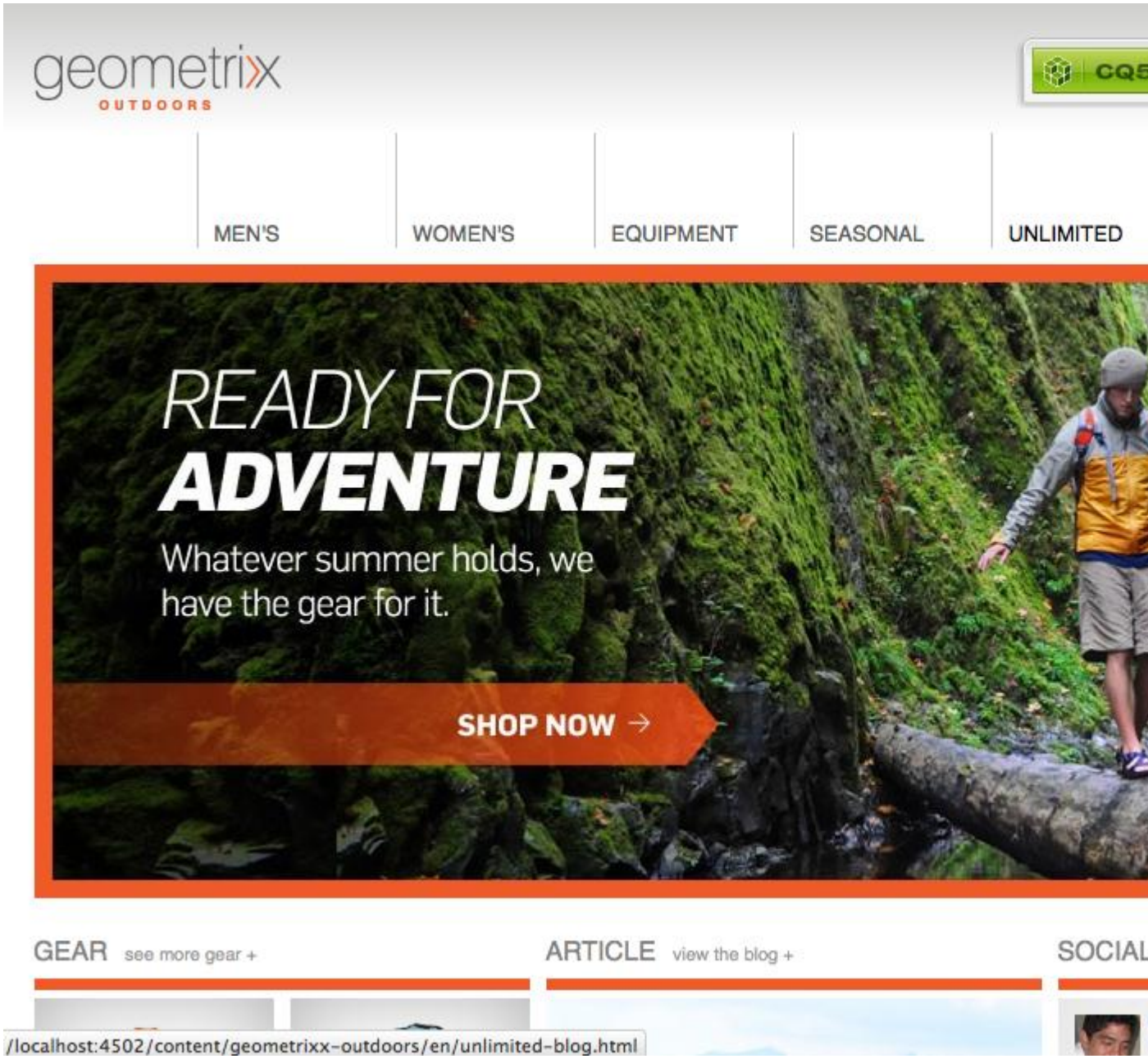


From here you can:

- Create a **New...** experience; for example, Test&Target offers, teasers and newsletters.
- **Edit** the details of a specific teaser page or newsletter (a double-click can also be used).
- Define the **Properties...** for a specific teaser page or newsletter.
- **Simulate** the look and feel of an experience (teaser page or newsletter).
  When the simulated page is open you can then open the sidekick to switch into edit mode for that page.
- **Analyze...** the impressions generated for a page.
- **Delete** items when they are no longer needed.
- **Search** for your text (the Title field of the experience will be searched).
- Use **Advanced** search to apply filters to the search.

SIMULATING YOUR CAMPAIGN EXPERIENCES

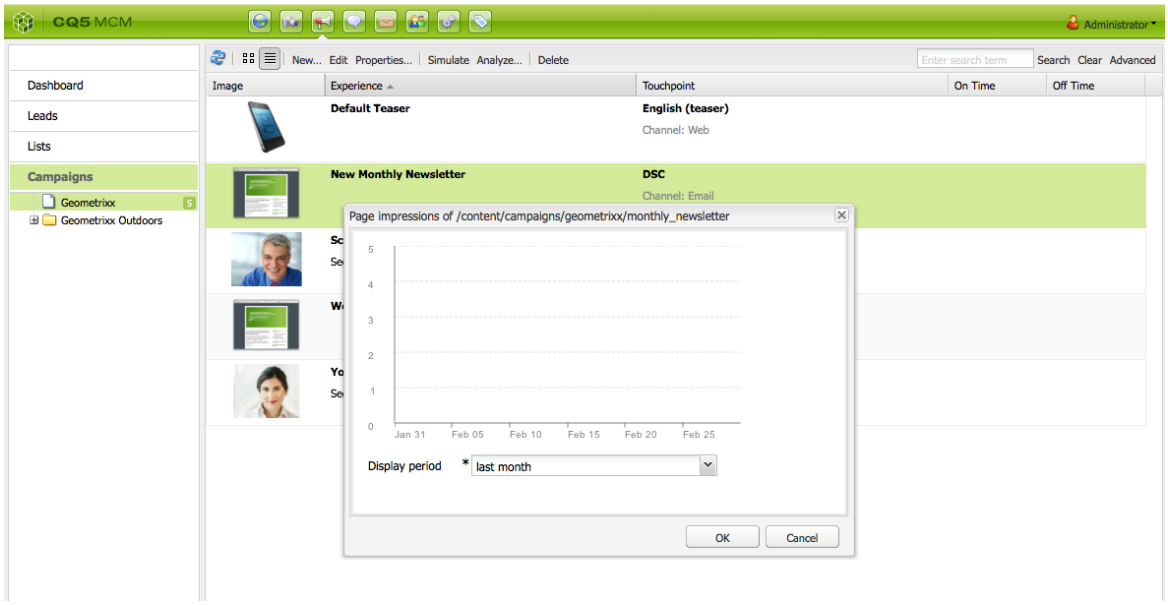
In the MCM, click **Campaigns**. Ensure that the list view is active, then select the required campaign experience and click **Simulate**. The touchpoint (teaser or newsletter page) will be opened to show the experience that you have selected - as the visitor will see it.



From here you can also open the sidekick (click the small down arrow) to change to edit mode for updating the page.

ANALYZING YOUR CAMPAIGN EXPERIENCES

In the MCM, click **Campaigns**. Ensure that the list view is active, then select the required campaign experience and select **Analyze...**. A chart of the page impressions over time will be shown.



Qus: How you can inherit properties of one dialog to another dialog?

For inheriting properties we have to create two components with unique names in the base component dialog. For eg. If your plan is to have two rich text two rich text areas in the dialog of components that inherit from the base, then you must include two rich text areas with unique names in the base component dialog. In any case every input field of a dialog must have a unique name, else they will point to the same property path relative to the jcr:content node of the component when used on a page.

Any issue if name is not unique?
Each input field of a dialog must have a unique name else both will point to the same property path relative to the jcr:content node of the component when used on a page.

Qus: Can we restrict for certain users not to display some digital assets ?

You can always limit who can access certain folders in CQ Digital Assets by making the folder part of a CUG(closed user group).
Steps to make a folder part of a CUG:

>In CQ DAM, right-click the folder you want to add closed user group properties for and select Properties.
>Click the CUG tab.
>Select the Enabled check box to make the folder and its assets available only to a closed user group.

>Browse to the login page, if there is one, to add that information. Add admitted groups by clicking Add item. If necessary, add the realm. Click OK to save your changes.

Qus: How to connect external DB from CQ5 ?

This is possible. Follow link below:

http://helpx.adobe.com/experience-manager/kb/HowToConfigureSlingDatasource.html

Qus: What is the role of servlet engine in CQ5?

In CQ5, servlet Engine acts as the server within which each CQ (and CRX if used) instance runs as a web application.
Any Servlet Engine supporting the Servlet API version 2.4 (or higher) can be used.
We can always run CQ WCM without an application server, a Servlet Engine is needed. CRX + CQ WCM, shipped with CQSE can be used freely and is fully supported.

Qus: Mention 7 rules of David Model?

1. Data First, Structure Later. Maybe.	- Data first
2. Drive the content hierarchy, do not let it happen.	- Content Hierarchy
3. Workspaces are for clone(), merge() and update().	- Workspaces
4. Beware of Same Name Siblings.	- SNS
5. References considered harmful.	- References
6. Files are Files are Files.	- Files
7. IDs are evil.	- ID

Qus: Explain the process of Content Moved from author to publish instance in CQ5?

Replication agent is the guy who does movement of data from author instance to publish instance.
It agent has following functions:

1. Move content from author to publish instance.
2. If any content is updated, then it flushes the older content from dispatcher cache.

Qus: Can we configure many replication agents?

Usually a development environment can contain multiple CQ – author and multiple CQ – publish instances. So an author instance can be configured to have many replication agents. Each replicate agent will replicate the content in 1 or more publish instances from then.

Qus: Any idea on is persistence manager in CQ5?

With the help of persistence manager CQ5 does save the content to a persistent storage like file system or a database.

CQ-crx content is stored using Tar(standard linux archive file format) Persistence manager by default.

When you need to store the repository content in a database, you can configure CQ5 to use a database persistence manager.

Qus: What is a repository structure?

- **Persistence Manager**: The persistence manager handles storage of the node/property tree that makes up the structure of the repository. This includes the name and position of each node and property in the hierarchy as well as the actual values of the *smaller* properties (the data store takes care of storing the larger values, see below). The default persistence manager is `TarPersistenceManager`. By default, it stores its data at `crx-quickstart/repository/workspaces/crx.default/`.
- **Data Store**: The data store handles storage of large content objects. When the value of property exceeds a given threshold, that data is not stored directly in the persistence manager storage. Rather, the data is stored in the data store and only a reference to the data is stored in the actual property record within the persistence manager storage. The default data store is `ClusterDataStore`. By default, it stores its data in `crx-quickstart/repository/workspaces/crx.default/`.
- **Journal**: The journal helps maintains data consistency and helps the system to recover quickly from crashes. In a clustered environment the journal plays the critical role of synchronizing content across cluster instances. The default journal is `TarJournal`, which stores its state in `crx-quickstart/repository/tarJournal/`.
- **Version Storage**: The version storage holds all the versioning information for the repository. From *within the repository* this information is exposed at the repository path `/jcr:system/jcr:versionStorage`. By default, the actual storage location of the data on-disk is `crx-quickstart/repository/version/`.
- **Indexes**: The indexes are used for fast lookup of data for search and other functions. On disk, the index files can be found in `crx-quickstart/repository/workspaces/crx.default/index/` and `crx-quickstart/repository/repository/index/`.
- **Other File-Based Storage**: In addition to the above elements, CRX also stores some repository state as plain text and XML files directly in the following file system directories under `crx-quickstart/repository/repository/`:
  - `namespaces/`: Namespace registry.
  - `nodetypes/`: Node type registry.
  - `privileges/`: Privileges information.
  - `meta/`: Identifier of repository root node.

Qus: Explain OSGi[Open Systems Gateway initiative] in CQ5 ?

• Dynamic module system for Java.

• Universal Middleware Category.

• Helps applications to be constructed from small, reusable and collaborative components.

• OSGi bundles can contain compiled Java code, scripts, or any contents to be loaded in the repository.

• Helps the bundles to be loaded, installed.

Qus: Annotations

Service components can be annotated using the annotations provided by the org.apache.felix.dependencymanager.annotation bundle.
The following annotations are supported:

• **Component** - To register your components without the Annotations, you would have to implement Activators, which extend the DependencyActivatorBase class.

- **Activate**
- **Deactivate**
- **Modified**

• **Service** - The @Service annotation defines whether and which service interfaces are provided by the component. This is a class annotation.

• **Property** - The @Property annotation defines properties which are made available to the component through the ComponentContext.getProperties() method

• **Reference** - The @Reference annotation defines references to other services. These other services (consumed services) are made available to the component by the Service Component Runtime.

Qus: How clustering is done in CQ5?

CQ5 CRX is pre-loaded to run within a cluster, even when running a single instance. Hence the configuration of multi-node clusters with little effort happens in CQ5.

Qus: What is the contribution of Servlet Engine in CQ5?

Servlet Engine pretends as a server within which each CQ (and CRX if used) instance runs. Eventhough you can run CQ WCM without an application server, always a Servlet Engine is needed.

Qus: Explain the role of Dispatcher in CQ5?

In CQ5 Dispatcher helps to cache and load-balance. The main responsibilities are,

i) Caching – Cache as much content as possible[It helps to reduce the continuous functioning of layout engine frequently for generating content when in dynamic.

ii) Load-balancing – To increase the performance by load-balancing.

Qus: State various strategies used by Dispatcher?

i) Cache as much content as possible as static pages.
ii) Accessing layout engine as little as possible.

Qus: Where does the cache directory exists for CQ5?
The cached documents are created in the root of a web-server which is preconfigured.

3) Explain the methods of Caching adopted by Dispatcher?

i) Dispatcher invalidates those pages whose content has been updated and replaces it with new content.
ii) Auto-Inavidation automatically removes the contents which are not relevant.

Qus: Explain the process steps involved in content moved from publish instance to author instance?

The latest/recent content is moved from publish instance to author instance using reverse replication and the job is done by reverse replication agent.

The agent places any content updates in an out-box configured in publish instance. The author environment replication listeners always keep listening to the publish out-box and whenever any content is placed in publish out-box, the listeners takes it and update the content.

Qus: Explain methods through which dispatcher performs Load-balancing?

i) Performance Statistics
ii) Sticky Connections

Qus: How to Use Dispatcher with Mapped content

Use Case: You are using /etc/map or resource resolution to map your content and dispatcher flush is not working any more.
More information about Sling mapping can be found here http://sling.apache.org/site/mappings-for-resource-resolution.html

I have seen a lot of customer using sling mapping or resource resolver setting to map /content/<There site> to / to shorten URL. But as soon as they do that, They claim that dispatcher flush is not working. And here is reason why,

1) Dispatcher flush does not take mapping or resource resolver rule in to account to flush cache or invalidate static file.

To understand it better, Here is use case,

1) From example your site URL is somedomain.com/content/mysite/en/us/survey.html and then you shorten it to somedomain.com/en/us/survey.html by mapping rules or resource resolver rule.

2) Suppose your document root is /docroot/htdocs

3) Now when some one try to access page somedomain.com/en/us/survey.html your page will get cached under /docroot/htdocs/en/us/survey.html

4) But when you will activate this page from author page under somedomain.com/content/mysite/en/us/survey.html will get flush, As dispatcher flush agent has no idea about mapping rules.

Resolution:

Now in order to avoid this problem, You should have mod_rewrite rule along with resource resolver and mapping rules.

Here is basic example of rules you want in your mod_rewrite (These rules will differ from site to site)

RewriteRule ^/$ <Your home page>

RewriteCond %{REQUEST_URI} !^/apps/(.*) [NC]
RewriteCond %{REQUEST_URI} !^/etc(.*) [NC]
RewriteCond %{REQUEST_URI} !^/libs(.*) [NC]
RewriteCond %{REQUEST_URI} !^/content(.*) [NC]
RewriteCond %{REQUEST_URI} !^/system(.*) [NC]
RewriteCond %{REQUEST_URI} !^/dam(.*) [NC]
RewriteRule ^/(.*) /content/mysite/$1 [PT]
RewriteRule ^/(.*)\?(.*) /content/mysite/$1 [PT]

Above rule make sure that content at right path is getting flushed.

Some more trick (But will not work 100% see http://forums.adobe.com/thread/1082213 for detail):

You can also do something like

LoadModule headers_module modules/mod_headers.so
RequestHeader edit CQ-Handle /content/mysite(/.*) $1 early

Another solution at replication agent level By David to solve this issue,

http://adobe-consulting-services.github.io/acs-aem-commons/features/dispatcher-flush-rules.html

code for above is found here https://github.com/Adobe-Consulting-Services/acs-aem-commons/tree/master/bundle/src/main/java/com/adobe/acs/commons/replication

Q: Why I will use etc/map or resource resolution if these rules are enough for mapping ?

A: Mod_rewriter rules will take care of just mapping and not link rewriting. You need /etc/map or resource resolver rules for link rewriting.

Q: Why dispatcher flush do not take mapping rules in to account ?
A: There is already an enhancement request for this.

Q: I don't want to write these rewrite rules, How else I can handle this ?
A: One option is set your stat file level to 0 (Which is default), This might invalidate all resource under invalidation tag of dispatcher.any upon activation. But again if you watch your dispatcher log carefully, wrong file will get evicted on activation. But you still see updated content due to invalidation.

Q: How about vanity URL ?
A: Well thats problematic. You might want to stick with stat file or write mod_rewrite rules for them as well.

Qus: Question / Problem

What is the Tar Persistence Manager and how to configure it?

Tar Persistence Manager

You can store CRX content in a Persistence Manager. By default, when you install CRX 1.4, the persistence manager saves the repository content to tar files. See also: Tar Persistence Manager.

Purpose

The Tar Persistence Manager (Tar PM) is a disk-based persistence manager that uses the tar file format for storing content and is useful in situations where high performance of creating and modifying data is required (as the Tar format is append-only, so writes are very efficient).

Tar PM clustering lets CRX compensate for hardware and software failures by eliminating a single point of failure (one server) while using applications without changes. Unlike clustering storage based on a database server (also available in CRX), Tar PM clustering is based on networked file systems.

Tar PM versus a RDBMS-based PM

Both Tar PM and database-based PMs support transactions, any file system, and optimization at runtime or in batch mode. Although Tar PM is a new technology, it does have the following advantages over using an database based persistence manager:

Tar files are append-only.

Tar files can be backed up easily online.

Tar is a standard file format, accessible via known tools, such as tar, WinZip, and so on.

Tar is a platform-independent format.

Low cost of ownership and license.

Tar PM is specifically designed for JCR repositories.

Tar PM is faster than RDBMS-based persistence managers for the JCR use case.

The Tar PM takes advantage of the very simple key-value pair data structure of CRX.

Configuring the Tar PM

The configuration options are:

```
<PersistenceManager class="com.day.crx.persistence.tar.TarPersistenceManager">
   <param name="maxFileSize" value="256"/>
   <param name="autoOptimizeAt" value="2:00-5:00"/>
   <!-- since 1.4.1 -->
   <param name="bindAddress" value=""/>
   <param name="portList" value=""/>
   <param name="preferredMaster" value="false"/>
   <param name="lockClass" value="com.day.crx.util.NativeFileLock"/>
   <param name="lockTimeout" value="0"/>
   <param name="fileMode" value="rw"/>
   <param name="optimizeSleep" value="1"/>
   <param name="maxIndexBuffer" value="32"/>
</PersistenceManager>
```

maxFileSize	(optional, default is 256) If the current data file grows larger than this number (in megabytes), a new data file is greated (that means, if the last entry in a file is very big, a data file can actually be much bigger, as entries are not split among files). The maximum file size is 1024 (1 GB). Data files are kept open at runtime. Depending on the amount of data is stored in the Tar PM, this value needs to be increased or the limit of open files per process needs to be adjusted. If this value is changed when tar files already exist, new tar files will grow up to this size (existing files are not changed).

autoOptimizeAt	(optional, default is 2:00-5:00) Automatically optimize at the given time. When the optimization should be run. Example: 2:00 to automatically optimize every morning at two. The index files will be merged as well if required. To disable the automatic optimization, set the value to "-0" (which actually means 'stop optimization at midnight').
bindAddress	If the synchronization between cluster nodes should be done over a specific network interface. By default all network interfaces are used. Default: empty (use all interfaces).
portList	The list of ports to use in master mode. By default any free port is used. When using a firewall, open ports must be listed. One port per workspace is required. A list of ports or ranges is supported, for example: 9100-9110 or 9100-9110,9210-9220. Default: 0 (any port).
preferredMaster	Only applicable in a clustering environment. If enabled, this cluster node will try to become the master even if another cluster node was started before. Default: false (not enabled)
lockClass	The name of the class to use for locking. Supported are com.day.crx.util.NativeFileLock and com.day.crx.util.CooperativeFileLock. When using a file system that does not support file locking (for example some older versions of NFS), the cooperative locking class should be used. Default: com.day.crx.util.NativeFileLock
lockTimeout	When clustering is used, the maximum time (in milliseconds) to wait to lock the shared files. Default: 0 for no limit.
fileMode	The file mode how to open the data files. Options are "rw" (read-write), "r" (read-only), "rwd" (read-write, content is written synchronously), and "rws" (read-write, content and metadata changes are written synchronously). Optionally a + can be appended to call fsync after writing (however this will slow down writes a lot). Default: "rw" for read-write.
optimizeSleep	(optional, default is 1) The number of milliseconds to wait after optimizing a transaction. Floating point precision is supported.
maxIndexBuffer	(optional, default is 32) After an abnormal termination, at most this much data (in megabytes) needs to be scanned in order to re-create the tar entry index.

Configuration values are read when the repository is started; that means you may want to re-start the repository after changing the configuration.**Note:** If you change the configuration after a workspace has already been created, you need to change both the repository.xml and workspace.xml files.

Optimizing Tar Files

See TarPM Optimization

Consistency Checking and Fixing

The Tar PM can check repository consistency and fix consistency problems at startup. To enable consistency checking and automatically fix problems, set the following options in the repository.xml and workspace.xml, and re-start CRX:

<param name="consistencyCheck" value="true"/>

<param name="consistencyFix" value="true"/>

In order to fix consistency problems, the consistency check setting must be enabled as well.

After the consistency check finished, disable the relevant settings, otherwisethe consistency check will always run when starting up CRX.

If Tar Files Get Big

If some data*.tar file are very large, the there are large transactions. Large transactions are a problem (in any case - not only for the Tar PM but also for the main memory and other sub-systems). You can analyze what is in a data*.tar file using the jsp file in the attachment.

Migration from Regular to Clustered Environment

The easiest way to migrate to a clustered environment is to export the data, change the configuration, and then import the data.

Migration from Clustered to Regular Environment

The easiest way to migrate to a regular environment is to export the data, change the configuration, and then import the data.

Qus: When and how to optimize the Tar PM?

Answer / Resolution

Optimizing Tar Files

As data is never overwritten in a tar file, the disk usage increases even when only updating existing data. When optimizing, the Tar Persistence Manager copies data that is still used from old tar files into new tar files, and deletes the old tar files that contain only old or redundant data.

If optimization is stopped before it is finished completely, then the next time when it is started it will continue where it left off (it doesn't start from the beginning).

If there is only one file, optimization will have no effect (no new file is created).

The disk space required to run the optimization is at most the size of one data tar file, which is 256 MB by default (for CRX 2.0; this setting can be changed using the parameter maxFileSize). This applies to both the shared directory as well as the local directory, meaning the total amount of temporary disk space used is at most 512 MB by default.

Automatic Scheduled Optimization

CRX automatically runs Tar PM optimization between 2 am and 5 am. If the automatic optimization is not finished at 5 am, then it will stop automatically. It will continue from there the next night (it doesn't start from the beginning).

To change the time when automatic optimization is run, use the Tar PM configuration option "autoOptimizeAt". Setting this value to "02:00" will trigger an optimization every day at two in the morning. In order to change the default time, edit repository/your_workspace/workspace.xml, as an example I set the optimization below to run at 1 a.m every day until 4 a.m the latest:

```
<PersistenceManager class="com.day.crx.persistence.tar.TarPersistenceManager">
    <param name="autoOptimizeAt" value="01:00-04:00" />
</PersistenceManager>
```

Disabling Automatic Scheduled Optimization

To disable the automatic optimization, set the value to "" (an empty string). This will work for CRX 2.1 and newer. For CRX versions up to 2.0, you need to set it to "-0" (which actually means 'stop optimization at midnight').

Manually Optimizing Tar Files using the CRX Explorer

To optimize tar files using the CRX console:

- In the CRX Console, log in as administrator.
- Click **Repository Configuration**.
- Select **Tar Persistence Manager Optimization** and click **Start Optimization**.
- To stop optimization while it is running, click **Stop Optimization**.

Note: In a clustered environment, this only works on the cluster nodes that is currently running as the master. Starting optimization on a slave cluster node has no effect.

Manually Optimizing Tar Files at Runtime

You can start optimizing the tar file manually at runtime by placing a specially named file optimize.tar in the folder where the tar files are. This file can be empty.

When optimization starts, this file is automatically renamed to optimizeNow.tar. If you need to stop optimization, you can do so by deleting this file. The file is automatically deleted when the optimization run ends.

Qus: Parsys Vs iParsys

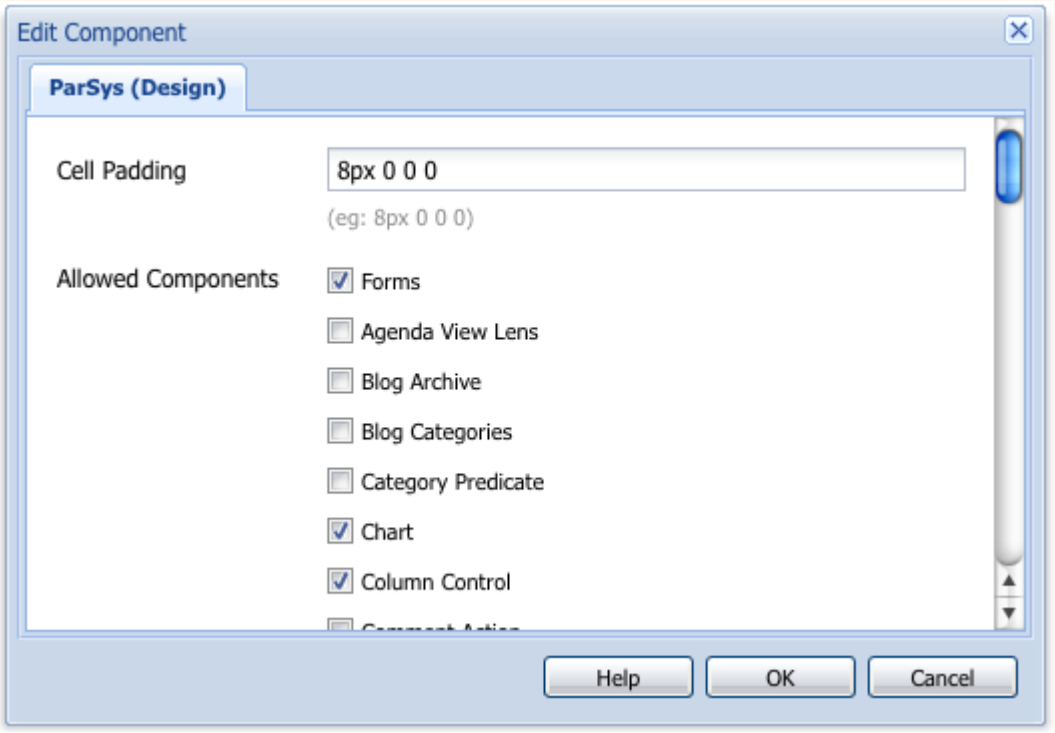
Paragraph System (parsys)

The paragraph system (parsys) is a compound component that allows authors to add components of different types to a page and contains all other paragraph components. Each paragraph type is represented as a component. The paragraph system itself is also a component, which contains the other paragraph components.

You configure which components users see by making them available to the user in Design mode.

For example, the content of a product page may contain the following:

- An image of the product (in the form of an image or textimage paragraph)
- The product description (as a text paragraph)
- A table with technical data (as a table paragraph)
- A form users fill out (as a forms begin, forms element, and forms end paragraph)



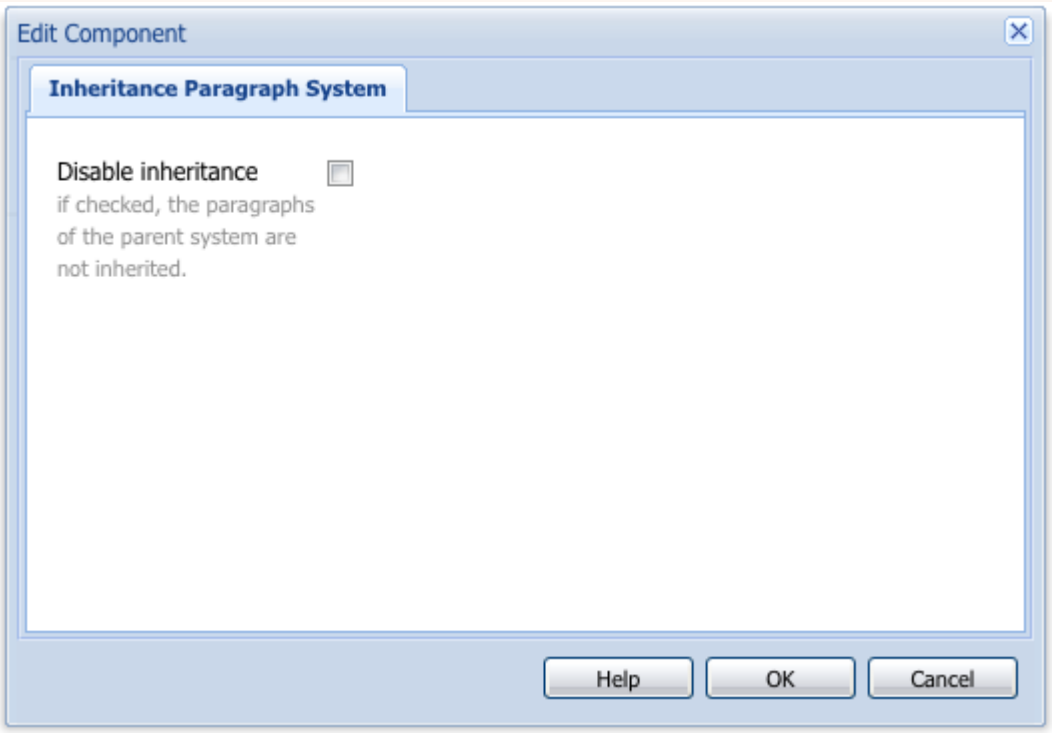
List of components available for use...
See individual components.

Inheritance Paragraph System (iparsys)

The inherited paragraph system is a paragraph system that also allows you to inherit the created paragraphs from the parent. You add paragraphs to iparsys at for example, /content/geometrixx/en/products and as result, all the subpages of products that also have iparsys with the same name inherit the created paragraphs from the parent. On each level, you can add more paragraphs, which are then

inherited by the children pages. You can also cancel paragraph inheritance at a level at any time.

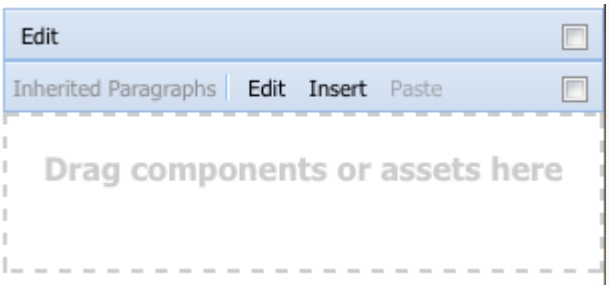
Simply, **iparsys is a parsys that inherits it's content from the ancestor pages.**



Disable Inheritance

If the checkbox is selected, child pages do not inherit the paragraph of this page.

The following example shows the iparsys component in Geometrixx:



Qus: How Sling locates the content and scripts

http://docs.adobe.com/docs/v5_1/html-resources/cq5_guide_developer/ch02s08.html

Sling is **content-centric**. This means that processing is focused on the content as each (HTTP) request is mapped onto content in the form of a JCR resource (a repository node):

the first target is the resource (JCR node) holding the content

secondly, the representation, or script, is located from the resource properties in combination with certain parts of the request (e.g. selectors and/or the extension)

RESTful Sling

Due to the content-centric philosophy, Sling implements a REST-oriented server and thus features a new concept in web application frameworks. The advantages are:

very RESTful; resources and representations are correctly modelled inside the server

removes one or more data models

previously the following were needed: URL structure, business objects, DB schema;

this is now reduced to: URL = resource = JCR structure

URL Decomposition

In Sling, and therefore also CQ5, processing is driven by the URL of the user request. This defines the content to be displayed by the appropriate scripts. To do this, information is extracted from the URL.

If we analyze the following URL:

`http://myhost/tools/spy.printable.a4.html/a/b?x=12`

We can break it down into its composite parts:

Table 1. URL Decomposition

protocol	host	content path	selector(s)	extension		suffix		param(s)
`http://`	`myhost`	`tools/spy`	`.printable.a4.`	`Html`	`/`	`a/b`	`?`	`x=12`

protocol

HTTP.

host

Name of the website.

content path

Path specifying the content to be rendered. Is used in combination with the extension; in this example they translate to `tools/spy.html`.

selector(s)

Used for alternative methods of rendering the content; in this example a printer-friendly version in A4 format.

extension

Content format; also specifies the script to be used for rendering.

suffix

Can be used to specify additional information.

param(s)

Any parameters required for dynamic content.

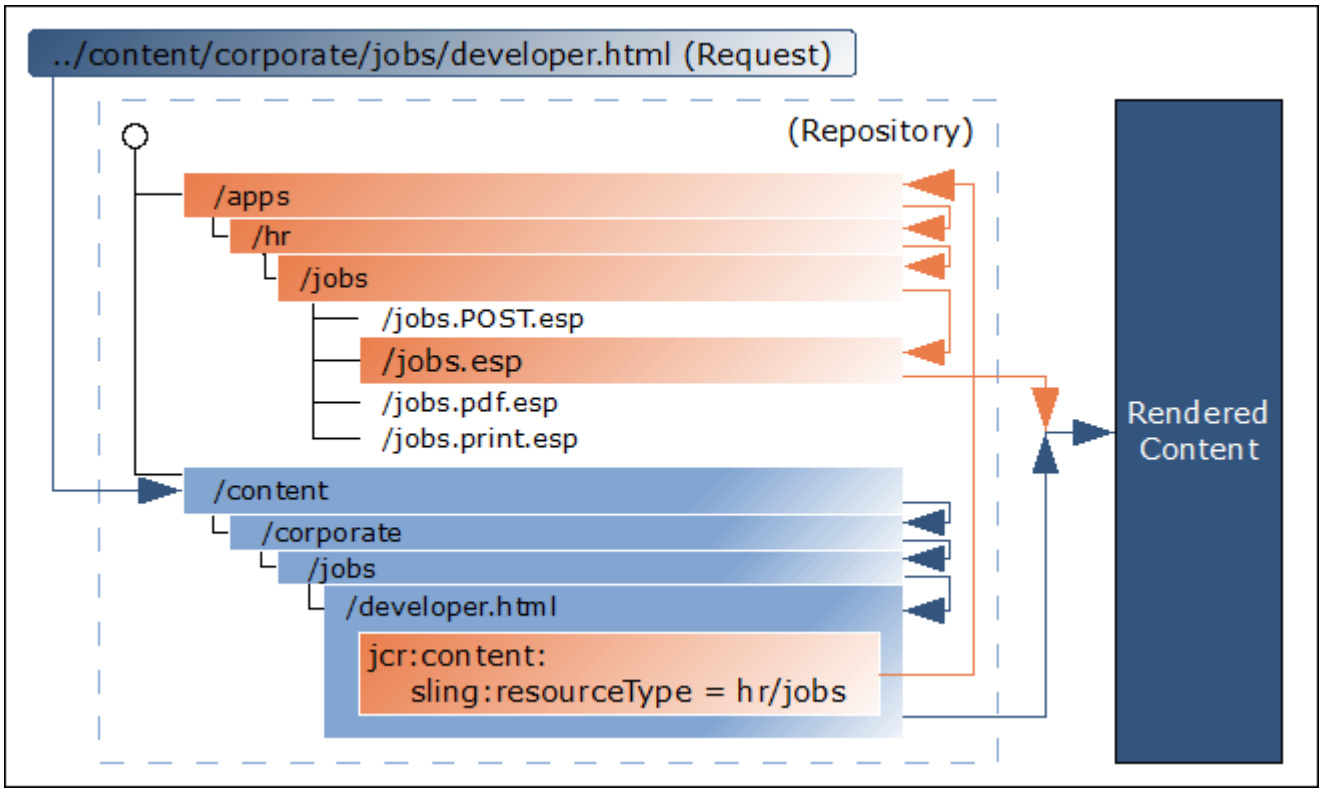
From URL to Content and Scripts

Using these principles:

the mapping uses the *content path* extracted from the request to locate the resource

when the appropriate resource is located, the *sling resource type* is extracted, and used to locate the script to be used for rendering the content

The figure below illustrates the mechanism used, which will be discussed in more detail in the following sections.



Therefore:

DO NOT specify which data entities to access in your scripts (as an SQL statement in a PHP script would do)

DO specify which script renders a certain entity (by setting the sling:resourceType property in the JCR node)

Mapping requests to resources

The request is broken down and the necessary information extracted. The repository is searched for the requested resource (content node):

first Sling checks whether a node exists at the location specified in the request;
e.g. ../content/corporate/jobs/developer.html

if no node is found, the extension is dropped and the search repeated;
e.g. ../content/corporate/jobs/developer

if no node is found then Sling will return the http code 404 (Not Found).

 **Note**

Sling also allows things other than JCR nodes to be resources, but this is an advanced feature.

Locating the script

When the appropriate resource (content node) is located, the *sling resource type* is extracted. This is a path, which locates the script to be used for rendering the content.

The path specified by the *sling:resourceType* can be either:

absolute

relative, to a configuration parameter

 **Note**

Relative paths are recommended by Day as they increase portability.

All Sling scripts are stored in subfolders of either /apps or /libs, which will be searched in this order.

A few other points to note are:

when the Method (GET, POST) is required, it will be specified in uppercase as according to the HTTP specification e.g. jobs.POST.esp (see below)

various script engines are supported:

.esp, .ecma: ECMAScript (JavaScript) Pages (server-side execution)

.jsp: Java Server Pages (server-side execution)

.java: Java Servlet Compiler (server-side execution)

.jst: JavaScript templates (client-side execution)

.js: ECMAScript / JavaScript (client-side execution)

The list of script engines supported by the given instance of CQ are listed on the Felix Management Console (http://localhost:4502/system/console/scriptengines).

Additionally, Apache Sling supports integration with other popular scripting engines (e.g., Groovy, JRuby, Freemarker), and provides a way of integrating new scripting engines.

Using the above example, if the *sling:resourceType* is hr/jobs then for:

GET/HEAD requests, and URLs ending in .html (default request types, default format)

The script will be /apps/hr/jobs/jobs.esp; the last section of the *sling:resourceType* forms the file name.

POST requests (all request types excluding GET/HEAD, the method name must be uppercase)

POST will be used in the script name.

The script will be /apps/hr/jobs/POST.esp.

URLs in other formats, not ending with .html

For example ../content/corporate/jobs/developer.pdf

The script will be /apps/hr/jobs/jobs.pdf.esp; the suffix is added to the script name.

URLs with selectors

Selectors can be used to display the same content in an alternative format. For example a printer friendly version, an rss feed or a summary.

If we look at a printer friendly version where the selector could be **print**; as in ../content/corporate/jobs/developer.print.html

The script will be /apps/hr/jobs/jobs.print.esp; the selector is added to the script name.

If no *sling:resourceType* has been defined then:

the content path will be used to search for an appropriate script (if the path based ResourceTypeProvider is active).

For example, the script for ../content/corporate/jobs/developer.html would generate a search in/apps/content/corporate/jobs/.

the primary node type will be used.

If no script is found at all then the default script will be used.

The default rendition is currently supported as plain text (.txt), HTML (.html) and JSON (.json), all of which will list the node's properties (suitably formatted). The default rendition for the extension .res, or requests without a request extension, is to spool the resource (where possible).

For http error handling (codes 404 or 500) Sling will look for a script at /libs/sling/servlet/errorhandler/404.esp, or 500.esp, respectively.

If multiple scripts apply for a given request, the script with the best match is selected. The more specific a match is, the better it is; in other words, the more selector matches the better, regardless of any request extension or method name match.

For example, consider a request to access the resource /content/corporate/jobs/developer.print.a4.html of type*sling:resourceType="hr/jobs"*. Assuming we have the following list of scripts in the correct location:

1. jobs.esp
2. jobs.GET.esp
3. jobs.GET.html.esp
4. jobs.html.esp
5. jobs.print.esp

6. jobs.print.a4.esp
7. jobs.print.html.esp
8. jobs.print.GET.html.esp
9. jobs.print.a4.html.esp
10. jobs.print.a4.GET.html.esp

Then the order of preference would be (10) - (9) - (6) - (8) - (7) - (5) - (3) - (4) - (2) - (1).

Qus: Describe sling:resourceSuperType ?

In addition to the resource types (primarily defined by the *sling:resourceType* property) there is also the resource super type. This is generally indicated by the *sling:resourceSuperType* property. These super types are also considered when trying to find a script. The advantage of resource super types is that they may form a hierarchy of resources where the default resource type sling/servlet/default (used by the default servlets) is effectively the root.

The resource super type of a resource may be defined in two ways:

by the *sling:resourceSuperType* property of the resource.

by the *sling:resourceSuperType* property of the node to which the *sling:resourceType* points.

For example:

/

a

b

sling:resourceSuperType = a

c

sling:resourceSuperType = b

x

sling:resourceType = c

y

sling:resourceType = c

sling:resourceSuperType = a

The type hierarchy of /x is [c, b, a, <default>] while for /y the hierarchy is [c, a, <default>] because /y has the*slingresourceSuperType* property whereas /x does not and therefore its supertype is taken from its resource type.

Qus: Sling Scripts cannot be called directly

Within Sling, scripts cannot be called directly as this would break the strict concept of a REST server; you would mix resources and representations.

If you call the representation (the script) directly you hide the resource inside your script, so the framework (Sling) no longer knows about it. Thus you lose certain features:

- automatic handling of http methods other than GET, including:
    - POST, PUT, DELETE which are handled with a sling default implementation
    - the POST.js script in your sling:resourceType location
- your code architecture is no longer as clean nor as clearly structured as it should be; of prime importance for large-scale development

Qus: What is the difference between

1. <c:import url="layout-link.jsp" />
2. <sling:include path="layout-link.jsp" />
3. <cq:include script="layout-link.jsp" />

What is the advantage of each tag? When should each be used?

Answer, Resolution

1. `<c:import url="layout-link.jsp" />`
I assume this is the import tag of the Standard Tag Library. This tag is documented at http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/c/import.html and does not know about Sling directly.
But -- asuming -- this tag is using a `RequestDispatcher` to dispatch the request, this tag will also pass Sling and the Sling resource resolver.
2. `<sling:include path="layout-link.jsp" />`
This is the include tag of the Sling JSP Tag library. This tag knows about Sling and also supports `RequestDispatcherOptions`.
3. `<cq:include script="layout-link.jsp" />`
This tag is Communiqué specific extension of the Sling JSP Tag library include tag. IIRC it supports callings scripts in addition to just including renderings of resources.

What is the advantage of each tag? When should each be used?
In a Communiqué application, I would suggest to generally use the Communiqué or Sling include tag since this provides you more Sling support.

You may use the JSTL import tag if you don't have specific requirements for Sling extended features, plan to use the JSP (fragment) outside of Communiqué or if you want to further process the generated (imported) content with a reader or a variable.

In the future, it is conceivable that the Sling and/or Communique tag library will also provide an import tag similar to the JSTL import tag to be able to further process the imported result.

Qus: What is a bundle?

Bundles are the set of java classes that can run as a module in cq5 to provide specific services.

Qus: What is the purpose of Activator.java file?

- The `Activator.java` file. It is the optional listener class to be notified of bundle start and stop events.

Qus: How you can inherit properties of one dialog to another dialog?

No, there is no way to directly inherit dialogs. The best you can do is to include the dialog tabs using path property.

You should create your tab your tabs in a different location and you can include it in your dialog using path property like shown below:

```
<items jcr:primaryType="cq:WidgetCollection">
        <tabs jcr:primaryType="cq:TabPanel">
            <items jcr:primaryType="cq:WidgetCollection">
                <tab1
                        jcr:primaryType="cq:Widget"
                        path="/apps/myproject/tab1.infinity.json"
                        xtype="cqinclude"/>
                <tab2
                        jcr:primaryType="cq:Widget"
                        path="/apps/myproject/tab2.infinity.json"
                        xtype="cqinclude"/>

            </items>
        </tabs>
</items>
```

Where tab1 and tab2 are tab panels.

So, in your case it will be something like this :

```
base_page_dialog_tab
      - dialog
      - title
      - description

inherited page-dialog-tab
      - custom field


base-page-template
    - include base page dialog tab here.

inerited-from-base-page
    - include Tab 1 - inherited page-dialog tab using path property
```

```
    - include Tab 2 - base page dialog tab using path property.
```

Resource Mapping

You are reading the **Adobe Experience Manager 5.6.1** version of *Resource Mapping*.
This documentation is also available for the following versions: Adobe Experience Manager 6.0 AEM 5.6

Resource mapping is used to define redirects, vanity URLs and virtual hosts for AEM.

For example, you can use these mappings to:

- Prefix all requests with `/content` so that the internal structure is hidden from the visitors to your website.
- Define a redirect so that all requests to the `/content/en/gateway` page of your website are redirected to`http://gbiv.com/`.

One possible HTTP mapping prefixes all requests to `localhost:4503` with `/content`. A mapping like this could be used to hide the internal structure from the visitors to the website as it allows:

```
localhost:4503/content/geometrixx/en/products.html
```

to be accessed using:

```
localhost:4503/geometrixx/en/products.html
```

as the mapping will automatically add the prefix `/content` to `/geometrixx/en/products.html`.

> **NOTE**
>
> See the Sling documentation, and Mappings for Resource Resolution and Resources for further information.

VIEWING MAPPING DEFINITIONS

The mappings form two lists that the JCR Resource Resolver evaluates (top-down) to find a match.

These lists can be viewed (together with configuration information) under the **JCR ResourceResolver** option of the Felix console; for example, `http://<host>:<port>/system/console/jcrresolver`:

- Configuration
  Shows the current configuration (as defined for the Apache Sling Resource Resolver).

- Configuration Test
  This allows you to enter a URL or resource path. Click **Resolve** or **Map** to confirm how the system will transform the entry.
- **Resolver Map Entries**
  The list of entries used by the ResourceResolver.resolve methods to map URLs to Resources.
- **Mapping Map Entries**
  The list of entries used by the ResourceResolver.map methods to map Resource Paths to URLs.

The two lists show various entries, including those defined as defaults by the application(s). These often aim to simplify URLs for the user.

The lists pair a **Pattern**, a regular expression matched to the request, with a **Replacement** that defines the redirection to impose.

For example, the:

Pattern `^[^/]+/[^/]+/welcome$`

will trigger the:

Replacement `/libs/cq/core/content/welcome.html`.

to redirect a request:

```
http://localhost:4503/welcome
```

to:

```
http://localhost:4503/libs/cq/core/content/welcome.html
```

New mapping definitions are created within the repository.

> **NOTE**
>
> There are many resources available that help explain how to define regular expressions; for example http://www.regular-expressions.info/.

CREATING MAPPING DEFINITIONS IN AEM

In a standard installation of AEM you can find the folder:

```
/etc/map/http
```

This is the structure used when defining mappings for the HTTP protocol. Other folders (`sling:Folder`) can be created under `/etc/map` for any other protocols that you want to map.

Configuring an Internal Redirect to /content

To create the mapping that prefixes any request to http://localhost:4503/ with `/content`:

1. Using CRXDE navigate to `/etc/map/http`.
2. Create a new node:
   - **Type** `sling:Mapping`
     This node type is intended for such mappings, though its use is not mandatory.
   - **Name** `localhost_any`
3. Click **Save All**.
4. **Add** the following properties to this node:
   - **Name** `sling:match`
     - **Type** `String`
     - **Value** `localhost.4503/`
   - **Name** `sling:internalRedirect`
     - **Type** `String`
     - **Value** `/content/`
5. Click **Save All**.

This will handle a request such as:

```
localhost:4503/geometrixx/en/products.html
```

as if:

```
localhost:4503/content/geometrixx/en/products.html
```

had been requested.

> **NOTE**
>
> See Resources in the Sling Documentation for further information about the sling properties available and how they can be configured.

> **NOTE**
>
> You can use `/etc/map.publish` to hold the configurations for the publish environment. These must then be replicated, and the new location (`/etc/map.publish`) configured for the**Mapping Location** of the Apache Sling Resource Resolver of the publish environment.

Qus: What are the design patterns used in day CQ5?

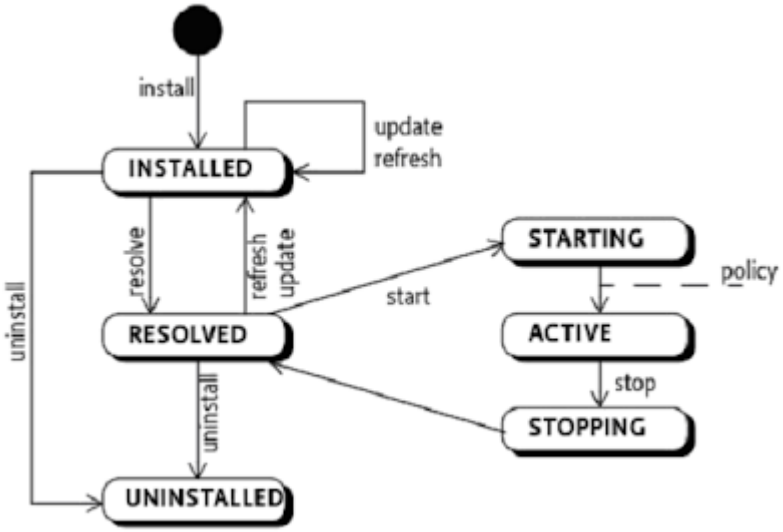
Any patterns that you like. OSGi makes CQ5 very modular, so you should be free to use whatever suits your needs.

CQ can handle what you are looking to do in a straight forward manner. They way to look at it, is that you're products are the content, you'll either have a specific template or data structure that represents your content(product) and the components that you would be building would be views that reference you're content(product)

Qus: What are bundle states?

A bundle can be in one of six states:

- UNINSTALLED
- INSTALLED
- RESOLVED
- STARTING
- STOPPING
- ACTIVE



Qus: Adding custom xtype to page properties

You need to place your script inside a new clientlib, defined with the category cq.wcm.admin. All clientlibs with this category are automatically included on the site admin. If you add the category cq.wcm.edit, your script will be included on the page editor as well.

Qus: AEM New Page vs New Site

"New site" option is used to create a site from an existing **Blueprint**. Blueprint is part of AEM's multisite manager (MSM) functionality. A blueprint is used to create multiple sites with common structure , for example **MySite** in English, French, Spanish and so on.

A Blueprint is created from an existing site. A site is a collection of pages, so you will be using **"New Page"** option for creating the site initially.
So if you don't have **"MySite"** ready as of yet, you'll be using "New page" option. Once "MySite" is ready and you want to create **copies of "MySite"** for different languages and countries to enforce a common structure, you will have to use **"New Site"** option.
You can read more about AEM's MSM capabilities here :

http://dev.day.com/docs/v5_2/html-resources/cq5_guide_power_user/ch13s04.html
http://dev.day.com/docs/en/cq/current/administering/multi_site_manager.html

Qus: Differences between traditional packages com.day.cq.workflow and com.adobe.granite.workflow

Adobe Granite Workflow API, introduced in AEM 5.6, is a replacement for the classic *Day Communique 5 Workflow API*. All new workflows should use the new API, but even in the AEM 6 there are some processes that still relies on the legacy API.
In the recent AEM versions both APIs are supported. For instance, in the *Process step* dialog, where you can list all `WorkflowProcesse`s, you'll see services implementing both the new and the old version of the interface. Nevertheless, to be sure that your code will be compatible with the future versions of AEM, you should use the new API.
Once you decide to use CQ or Granite API, it's important to stick to your choice, as you can't mix types from the first and the second API in one class and you shouldn't do it in one application.

Qus: Is there a way to access SlingRepository in a POJO?

First of all, if you use Sling then using `ResourceResolver` is generally more preferable than `SlingRepository`. It gives you an useful `Resource` abstraction layer and you can still get the underlying JCR `Session` object using `adaptTo()` method.
But back to your question, POJO always lives in a context, there is some entrypoint that runs the whole thing. In Sling there is a few such places: JSP, servlet or an OSGi component. In all of these entrypoints there is at least one way to get access to the repository:

1. JSP
   - use `resourceResolver` binding
   - use `getService(ResourceResolverFactory.class)` to create an administrative resolver,
2. Servlet or filter
   - use `request.getResourceResolver()` to get the request session,
   - or see the next point.
3. Any OSGi service (including servlet or filter)
   - use `@Reference ResourceResolverFactory` to get the factory and create administrative resolver.

After that you can pass the resource resolver to your POJO constructir. I think it's a better option than using hacks with the `FrameworkUtil` for a few reasons:

- if you pass the `ResourceResolver` to the POJO constructor, it's clear that this particular class operates on the repository,
- you don't need to worry about closing sessions (as POJO may not have a defined lifecycle),
- if you create your POJO in servlet or component (cases 1 and 2), it's better to use the request's session to avoid working on the administrative one.