

Web Quick Start

a technical tour

Brian Remmington
Alfresco Product Architect

Web Quick Start – A Technical Tour

What we'll cover

- A demonstration
- The Web Quick Start model
- Overview of the features in the repository tier
- Overview of the sample web application

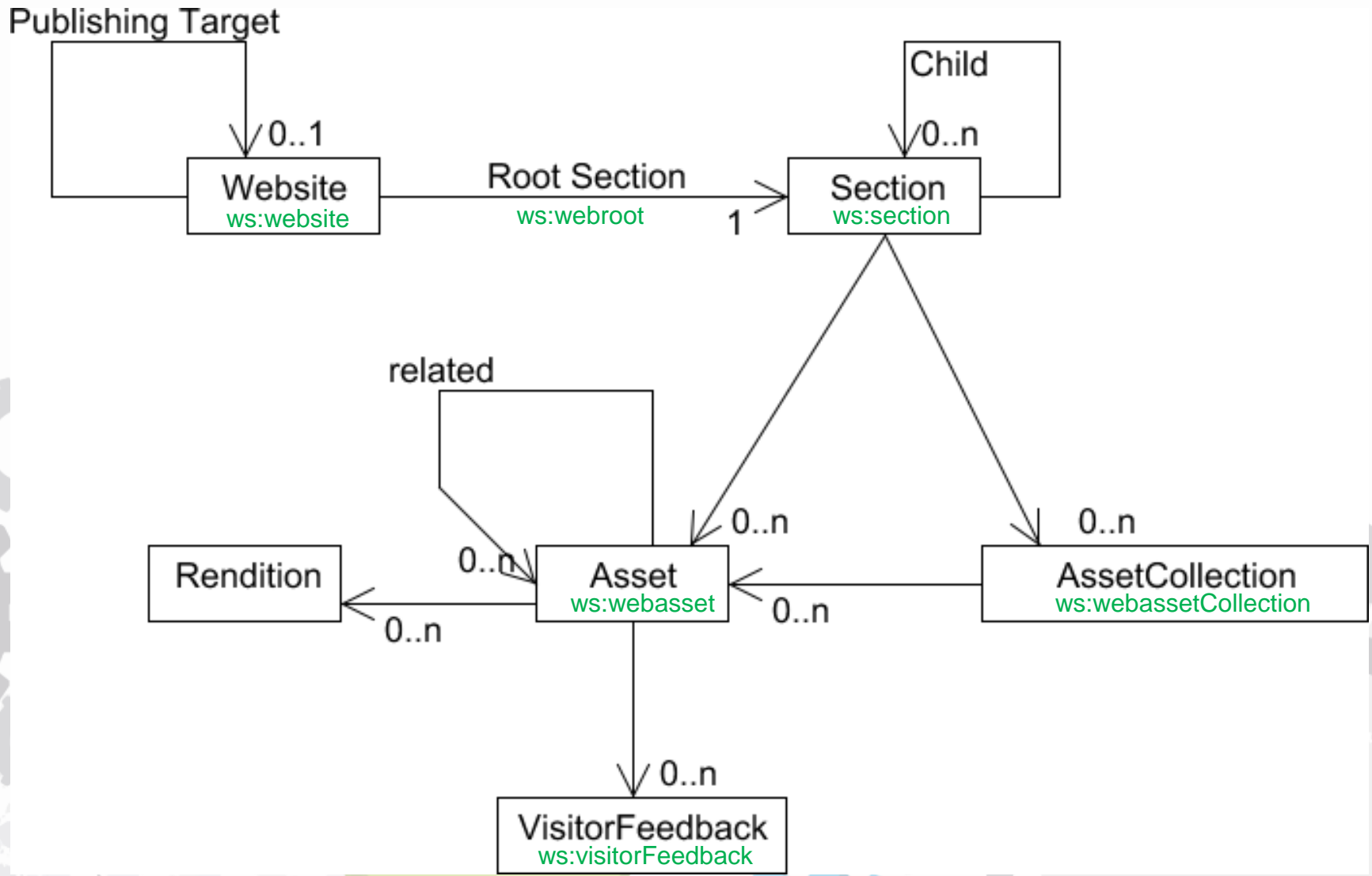
Primary goal:

- To show you how Web Quick Start works so you can start to use it and extend it

WQS in action



The Web Quick Start model



Asset collections

- Groups of assets that are uniquely named within the scope of a particular section
- Static asset collections
 - Editor control over which assets are included
- Dynamic asset collections
 - Configured with a search query, maximum number of results, and a refresh period in minutes
 - The query is executed periodically (as configured), and the results are placed in the asset collection
 - CMIS and Lucene supported

Featured

This is a collection of featured content items.

- [Ethical funds](#)
- [Fresh flight to Swiss franc as Europe's bond strains return](#)
- [Alfresco Datasheet - Social Computing](#)

Latest Blog Articles

Ethical Funds

Nonumy postea vivendum his at, dicit ornatus eos ut. Mei id tantas eligendi, his vidit tritani ut. ...

29 July 2010 05:24 PM

Company organises workshop

Nonumy postea vivendum his at, dicit ornatus eos ut. Mei id tantas eligendi, his vidit tritani ...

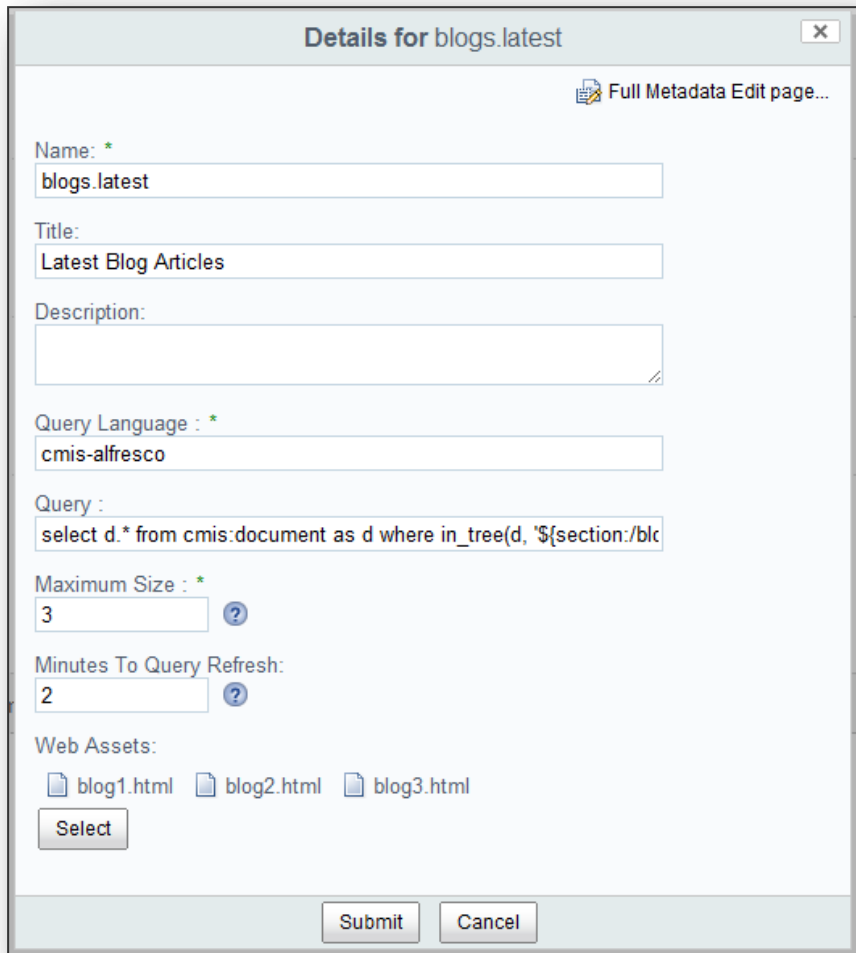
29 July 2010 05:24 PM

Our top analyst's latest thoughts

Nonumy postea vivendum his at, dicit ornatus eos ut. Mei id tantas eligendi, his vidit tritani ...

29 July 2010 05:24 PM

Dynamic asset collections



The screenshot shows a web-based configuration window titled "Details for blogs.latest". It contains several input fields and buttons:

- Name:** * blogs.latest
- Title:** Latest Blog Articles
- Description:** (empty text area)
- Query Language :** * cmis-alfresco
- Query :** select d.* from cmis:document as d where in_tree(d, '\${section}/blc
- Maximum Size :** * 3 (with a help icon)
- Minutes To Query Refresh:** 2 (with a help icon)
- Web Assets:** Includes icons for blog1.html, blog2.html, and blog3.html, and a "Select" button.
- Buttons:** "Submit" and "Cancel" at the bottom.

- Queries can contain references to sections using the “section” placeholder:
- `${section:.}` is the section that owns the asset collection
- `${section:/}` is the root section of the website
- `${section:..}` is the parent of this collection’s section
- `${section:/news/global}` is the section at the path “news/global” from the root
- You can plug in your own placeholders too

Dynamic asset collections

Details for blogs.latest

Full Metadata Edit page...

Name: *
blogs.latest

Title:
Latest Blog Articles

Description:

Query Language : *
cmis-alfresco

```
SELECT d.* FROM cmis:document as d WHERE in_tree(d, '${section:/blog}')  
AND d.cmis:objectId='D:ws:article'  
ORDER BY d.cmis:creationDate DESC
```

Minutes To Query Refresh:
2

Web Assets:
blog1.html blog2.html blog3.html

Select

Submit Cancel

- Queries can contain references to sections using the “section” placeholder:
- `${section:.}` is the section that owns the asset collection
- `${section:/}` is the root section of the website
- `${section:/news/global}` is the section at the path “news/global” from the root
- You can plug in your own placeholders too

Generating renditions

- Each section may define what renditions should be created when an asset is added or updated
- The configuration allows rendition definitions to be related to asset types (both by content type and MIME type)

Rendition Config:

```
ws:image=ws:smallThumbnail,ws:image=ws:imagePreview,application/pdf=ws:swfPreview
```

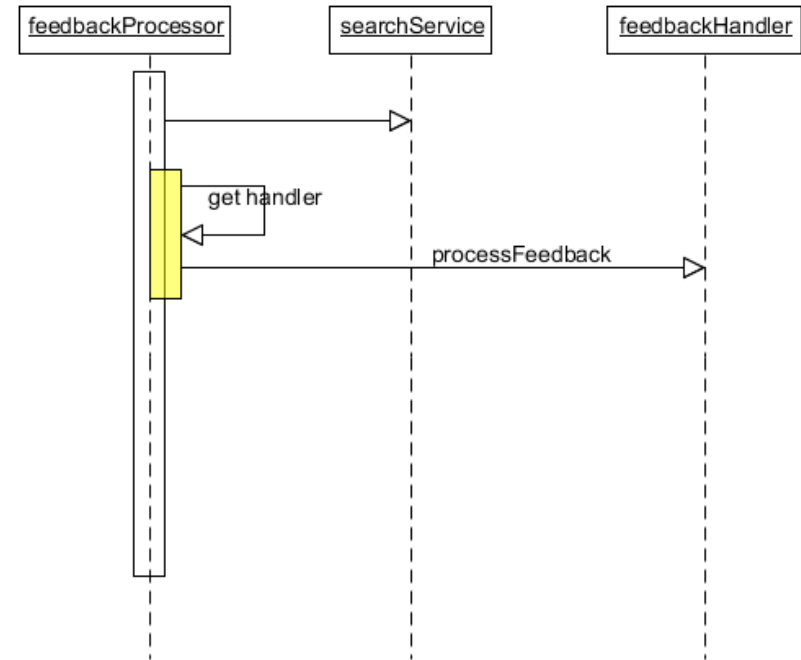
- Rendition definitions are defined in a Spring bean named “*wqsmodule_renditionDefinitions*” (defaults are in the file “rendition-context.xml”)
- Each section may either inherit the rendition configuration from its parent section or not

Generating renditions

```
<bean id="vqsmodule_renditionDefinitions" class="java.util.ArrayList">
  <constructor-arg>
    <list>
      <bean class="org.alfresco.module.org_alfresco_module_vcmquickstart.rendition.BootstrapCompositeRenditionDefinition">
        <property name="name" value="smallThumbnail"/>
        <property name="renderingEngineName" value="compositeRenderingEngine"/>
        <property name="definitions">
          <list>
            <bean class="org.alfresco.module.org_alfresco_module_vcmquickstart.rendition.BootstrapRenditionDefinition">
              <property name="renderingEngineName" value="reformat"/>
              <property name="parameters">
                <map>
                  <entry key="mime-type" value="image/jpeg"/>
                  <entry key="runAs" value="System"/>
                </map>
              </property>
            </bean>
            <bean class="org.alfresco.module.org_alfresco_module_vcmquickstart.rendition.BootstrapRenditionDefinition">
              <property name="renderingEngineName" value="imageRenderingEngine"/>
              <property name="parameters">
                <map>
                  <entry key="xsize" value="72"/>
                  <entry key="ysize" value="72"/>
                  <entry key="maintainAspectRatio" value="false"/>
                  <entry key="runAs" value="System"/>
                </map>
              </property>
            </bean>
          </list>
        </property>
      </bean>
    </list>
  </constructor-arg>
</bean>
```

Receiving and processing feedback

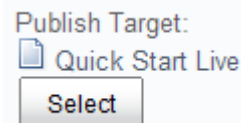
- Each website has a corresponding data list for feedback (auto-created)
- New entries are periodically processed – each type of feedback can have its own handler defined
 - *CommentFeedbackProcessorHandler*
 - *feedbackType* = "Comment"
 - *ContactFeedbackProcessorHandler*
 - *feedbackType* = "Contact Request"



```
<bean id="commentFeedbackProcessorHandler" parent="feedbackProcessorHandler"
      class="org.alfresco.module.org_alfresco_module_wcmquickstart.jobs.feedback.CommentFeedbackProcessorHandler">
  <property name="feedbackType" value="Comment" />
</bean>
```

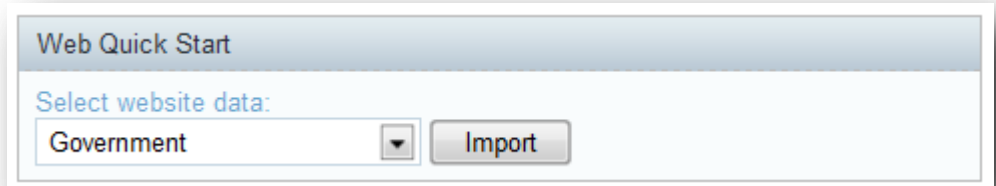
The publishing mechanism

- Each website has a publishing queue
- WQS has a PublishService that processes the queues
- Two actions defined: “*webqs_publishTree*” and “*webqs_publish*”
- Two workflows defined that use the actions
- Queues are published periodically – every minute by default (override with property `wcmqs.publishQueueProcessor.schedule`)
- Each website node may specify another website node that it publishes to



The WQS Share Dashlet

- Override map of available ACP files using a bean named *"wqsmodule_siteImportFileLocations"*



Web Quick Start

Select website data:

Government

```
<bean id="wqsmodule_siteImportFileLocations" class="java.util.TreeMap">
  <constructor-arg>
    <map>
      <entry key="financial" value="${wcmqs.importFileLocation.finance}"/>
      <entry key="media" value="${wcmqs.importFileLocation.media}"/>
      <entry key="government" value="${wcmqs.importFileLocation.government}"/>
    </map>
  </constructor-arg>
</bean>
```

```
# Import file location
wcmqs.importFileLocation.finance=alfresco/module/org_alfresco_module_wcmquickstart/bootstrap/webquickstart-finance.acp
wcmqs.importFileLocation.government=alfresco/module/org_alfresco_module_wcmquickstart/bootstrap/webquickstart-government.acp
wcmqs.importFileLocation.media=alfresco/module/org_alfresco_module_wcmquickstart/bootstrap/webquickstart-media.acp
```

The WQS sample web app

<FreeMarker>

Surf

springsource

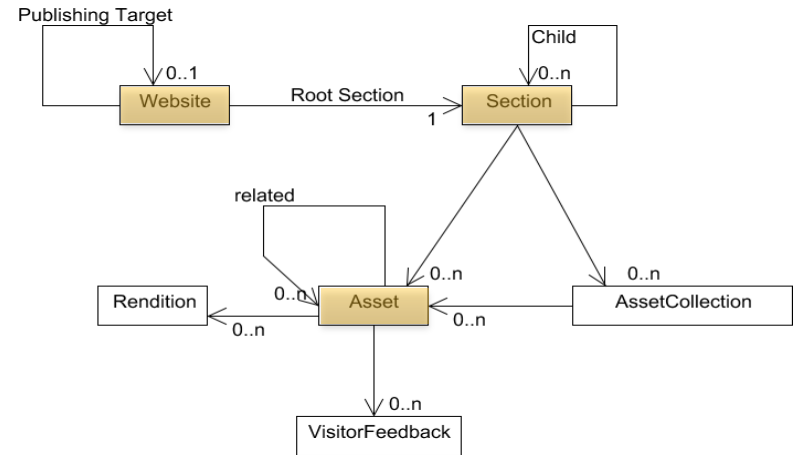
WQS API



Alfresco

The WQS API – websites, sections, assets

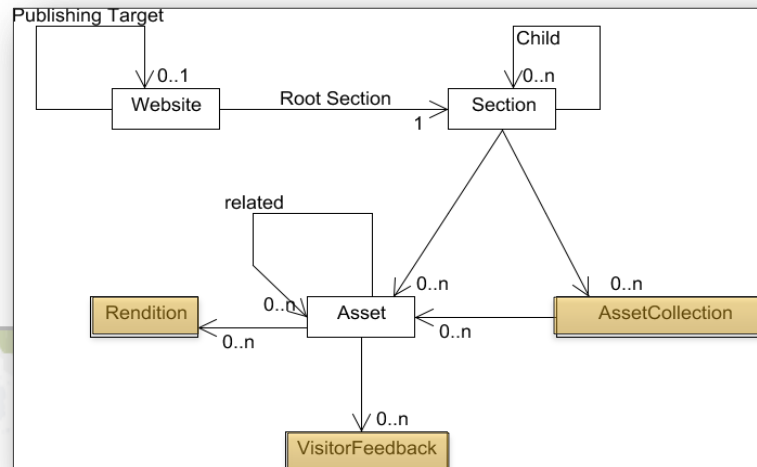
Bean “webSiteService” is a good starting point:



```
public class DevConExample1
{
    public void example1()
    {
        ApplicationContext appContext = new ClassPathXmlApplicationContext("alfresco/wcmqs-api-context.xml");
        WebSiteService webSiteService = (WebSiteService)appContext.getBean("webSiteService");
        WebSite website = webSiteService.getWebSite("localhost", 8080);
        Section globalNewsSection = website.getSectionByPath("/news/global/");
        Asset article1 = globalNewsSection.getAsset("article1.html");
        System.out.println(article1.getProperties());
    }
}
```

The WQS API – asset collections, renditions, feedback

```
UgcService ugcService = website.getUgcService();
AssetCollection latestArticles = globalNewsSection.getAssetCollection("latest.articles");
for (Asset asset : latestArticles.getAssets())
{
    System.out.println(asset.getTitle());
    ContentStream content = asset.getContentAsStream();
    if (content != null && content.getMimeType().startsWith("image"))
    {
        Rendition thumbnail = asset.getRenditions().get("smallThumbnail");
        System.out.println(thumbnail.getLength());
    }
    VisitorFeedbackPage feedbackPage = ugcService.getFeedbackPage(asset.getId(), 10, 0);
    for (VisitorFeedback feedback : feedbackPage.getFeedback())
    {
        System.out.println(feedback.getVisitorName());
        System.out.println(feedback.getComment());
    }
    ugcService.postFeedback(asset.getId(), "brian", "brian@example.com", null, "This is my comment");
}
```



URLs in WQS

- “Friendly” URLs
- The requested URL is parsed to find the website, section, and asset that are being addressed

website

www.example.com/news/global/financial_outlook_good

section

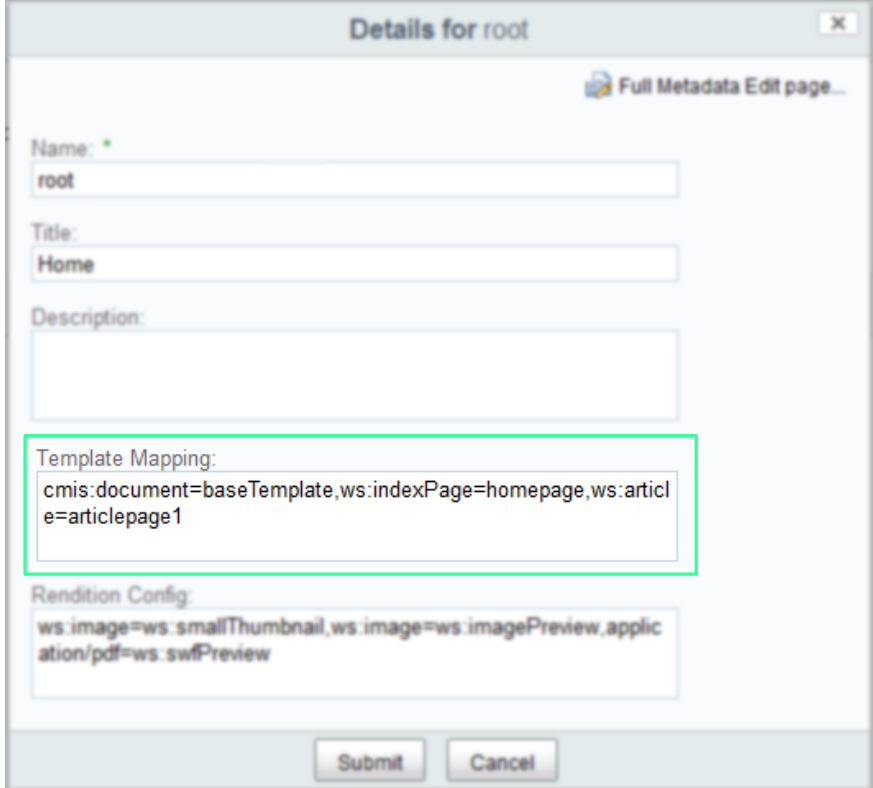
asset

financial_outlook_good

- The resolved API objects are stored on the request context with the names “website”, “section”, and “asset” for use by page components.

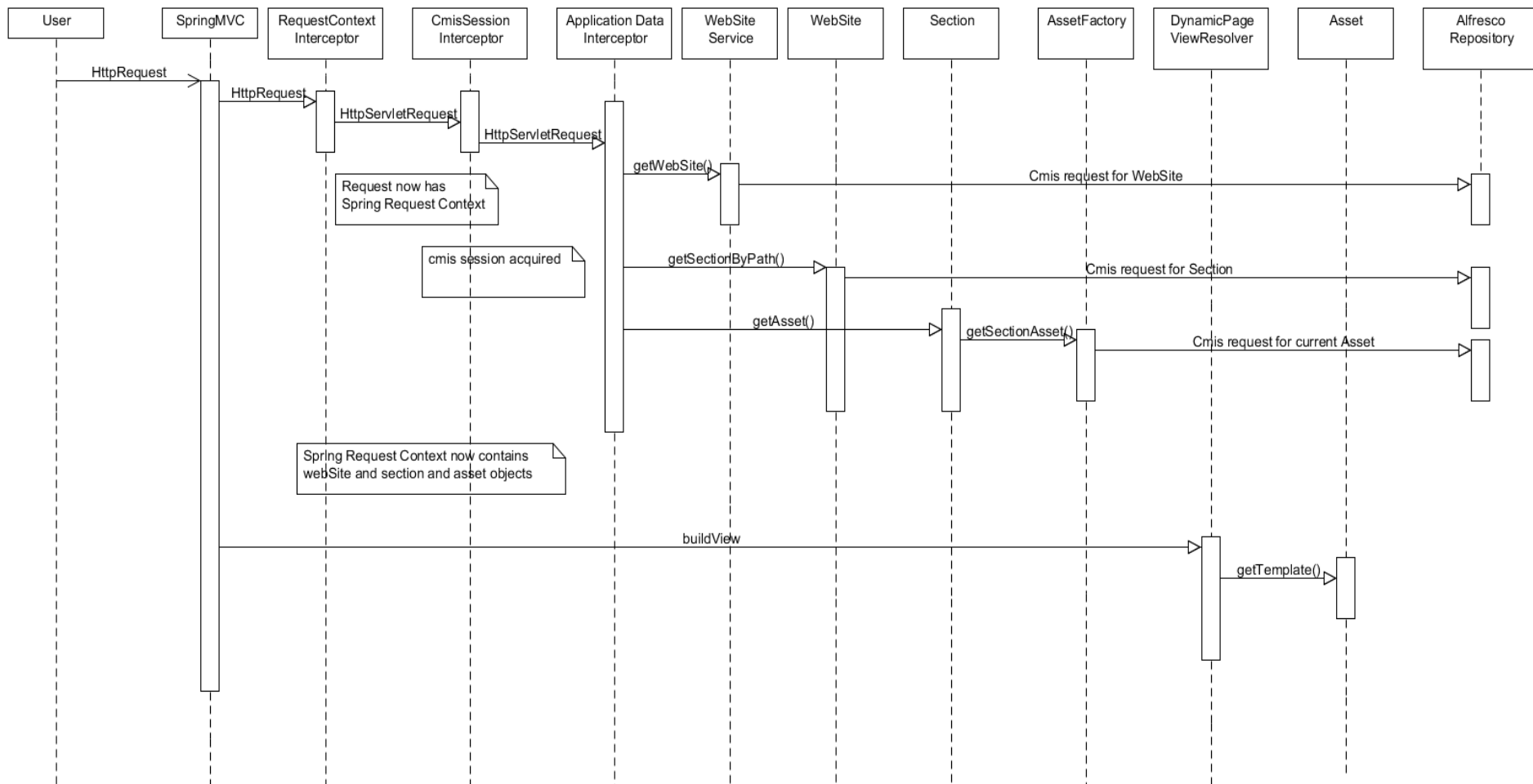
The dynamic page view resolver

- Wired in to respond whenever an asset has been resolved
- Discovers the page name to be used to render the asset
- Each section may define template mappings from asset type to view name
- Searches up the type hierarchy first, then the section hierarchy

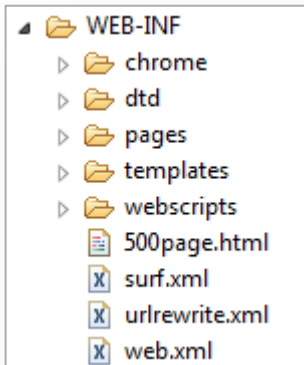


The screenshot shows a dialog box titled "Details for root" with a close button in the top right corner. Below the title bar is a link labeled "Full Metadata Edit page...". The form contains several input fields: "Name" with the value "root", "Title" with the value "Home", and an empty "Description" field. Below these is a "Template Mapping:" section, which is highlighted with a green border and contains the text: `cmis:document=baseTemplate,ws:indexPage=homepage,ws:article=articlepage1`. Below that is a "Rendition Config:" section containing the text: `ws:image=ws:smallThumbnail,ws:image=ws:imagePreview,application/pdf=ws:swfPreview`. At the bottom of the dialog are "Submit" and "Cancel" buttons.

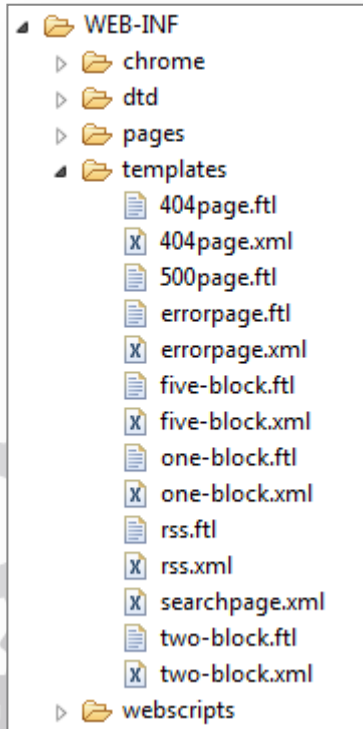
Handling a typical request



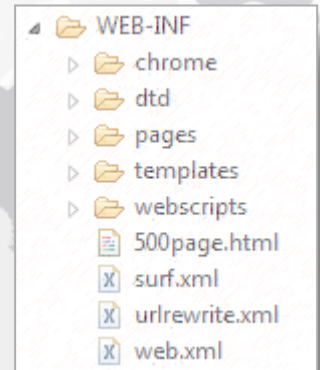
The WQS web app folder structure



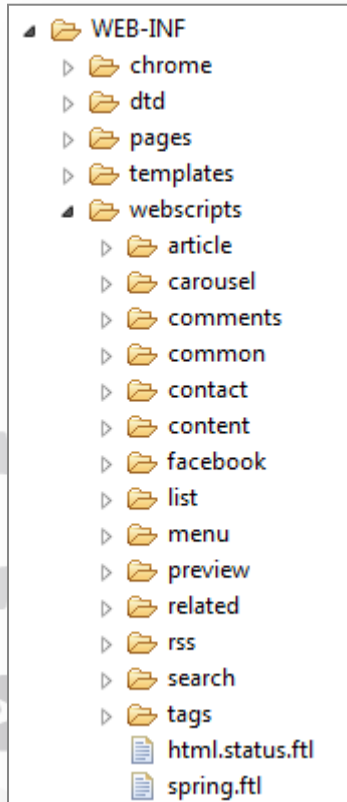
The WQS web app folder structure



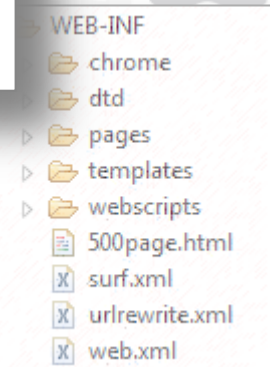
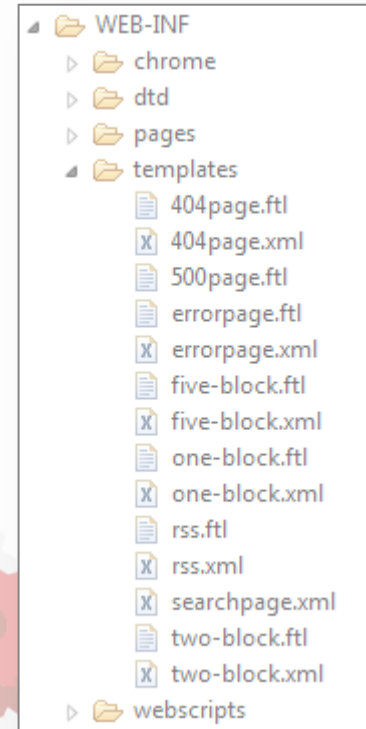
Think of “templates” as layouts:
HTML with holes cut out (“regions”)



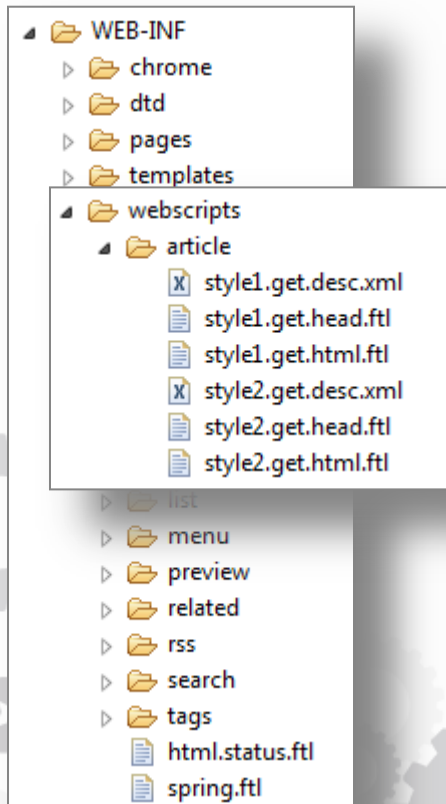
The WQS web app folder structure



The webscripts folder contains components: HTML fragments that can be used to fill in regions

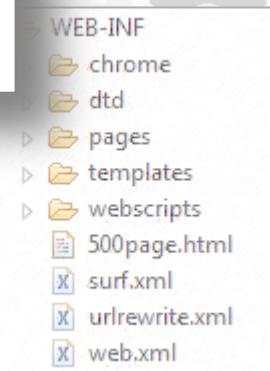
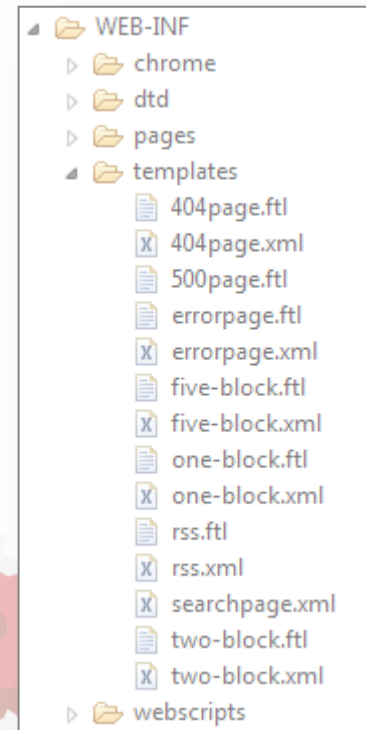


The WQS web app folder structure

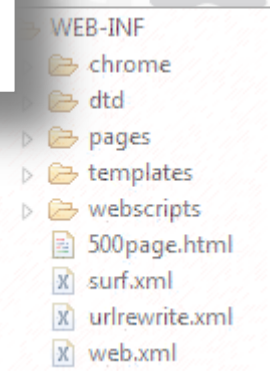
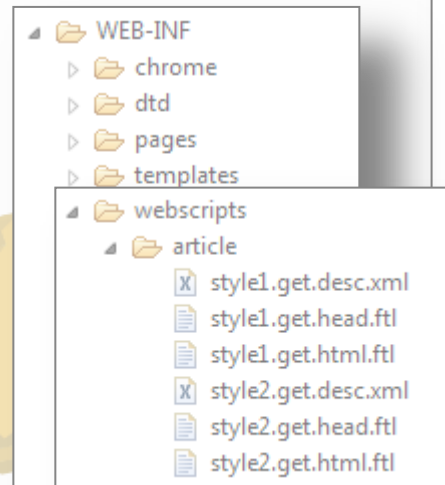
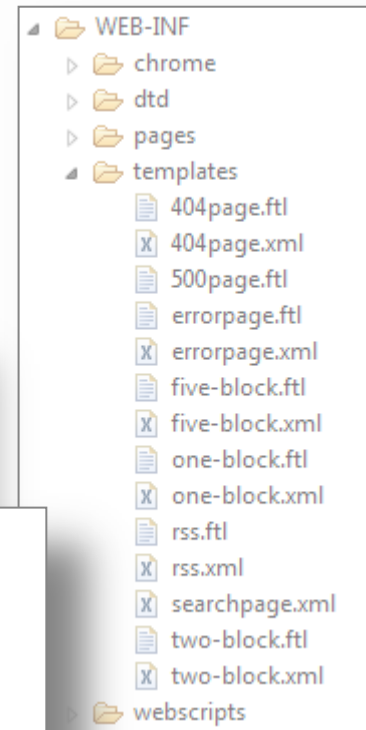
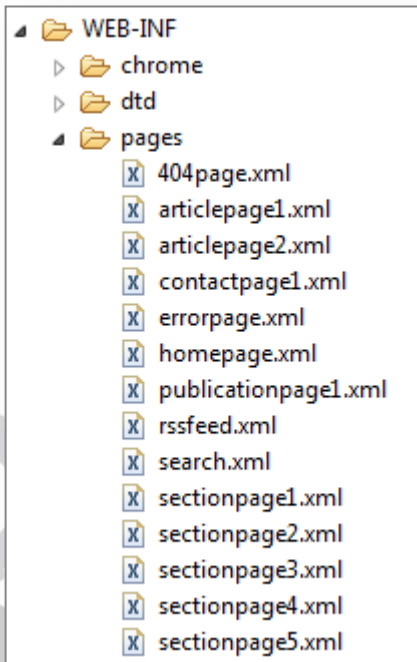


The webscripts folder contains components: HTML fragments that can be used to fill in regions

For example, there are two components that render article details: “article/style1” and “article/style2”



The WQS web app folder structure



Finally, pages weave templates and components together: a page specifies a template and which components are to fill which regions in that template

The anatomy of a page



The anatomy of a page

```
<?xml version="1.0" encoding="UTF-8"?>
<page>
  <id>homepage</id>
  <description>page with 5 main configurable blocks, ideal for site home page</description>
  <template-instance>five-block</template-instance>
  <authentication>none</authentication>
  <components>
    <component>
      <region-id>bellyband</region-id>
      <url>/carousel/slideshow</url>
      <properties>
        <collection>news.featured</collection>
      </properties>
    </component>
    <component>
      <region-id>left1</region-id>
      <url>/list/wide</url>
      <properties>
        <collection>news.top</collection>
      </properties>
    </component>
    <component>
      <region-id>right1</region-id>
      <url>/list/narrow</url>
      <properties>
        <collection>blogs.latest</collection>
      </properties>
    </component>
    <component>
      <region-id>bottom-left</region-id>
      <url>/list/links</url>
      <properties>
        <collection>featured.links</collection>
      </properties>
    </component>
    <component>
      <region-id>bottom-right</region-id>
      <url>/content/named</url>
      <properties>
        <asset>feature.html</asset>
      </properties>
    </component>
  </components>
</page>
```

The anatomy of a page

five-block.ftl

```
<#include "/common/page.ftl"/>

<@templateBody>
  <@region id="bellyband" scope="page"/>

  <div id="left">
    <@region id="left1" scope="page"/>
    <@region id="left2" scope="page"/>
    <@region id="left3" scope="page"/>

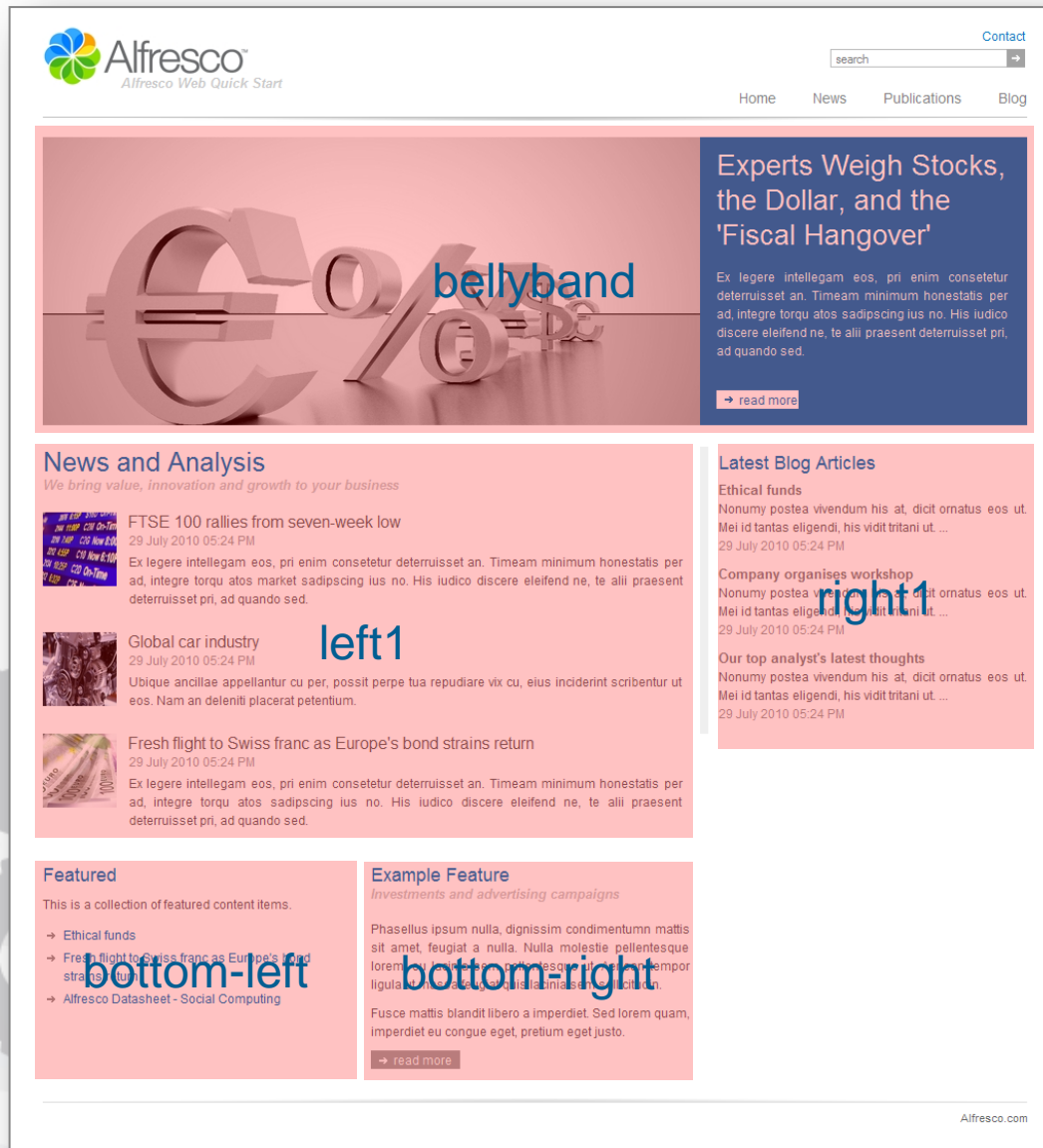
    <div class="h-box-1">
      <@region id="bottom-left" scope="page"/>
    </div>

    <div class="h-box-2">
      <@region id="bottom-right" scope="page"/>
    </div>

  </div>

  <div id="right">
    <@region id="right1" scope="page"/>
    <@region id="right2" scope="page"/>
    <@region id="right3" scope="page"/>
  </div>
</@>
```

The anatomy of a page



The anatomy of a page

```
<?xml version="1.0" encoding="UTF-8"?>
<page>
  <id>homepage</id>
  <description>page with 5 main configurable blocks, ideal for site home page</description>
  <template-instance>five-block</template-instance>
  <authentication>none</authentication>
  <components>
    <component>
      <region-id>bellyband</region-id>
      <url>/carousel/slideshow</url>
      <properties>
        <collection>news.featured</collection>
      </properties>
    </component>
    <component>
      <region-id>left1</region-id>
      <url>/list/wide</url>
      <properties>
        <collection>news.top</collection>
      </properties>
    </component>
    <component>
      <region-id>right1</region-id>
      <url>/list/narrow</url>
      <properties>
        <collection>blogs.latest</collection>
      </properties>
    </component>
    <component>
      <region-id>bottom-left</region-id>
      <url>/list/links</url>
      <properties>
        <collection>featured.links</collection>
      </properties>
    </component>
    <component>
      <region-id>bottom-right</region-id>
      <url>/content/named</url>
      <properties>
        <asset>feature.html</asset>
      </properties>
    </component>
  </components>
</page>
```

For example, the region “left1” is populated by the component “list/wide”. This component accepts a property named “collection” – in this case the value of that property is being set to “news.top”

The anatomy of a page

webscripts/list/wide.get.js

```
model.articles = collectionService.getCollection(context.properties.section.id, args.collection);
```

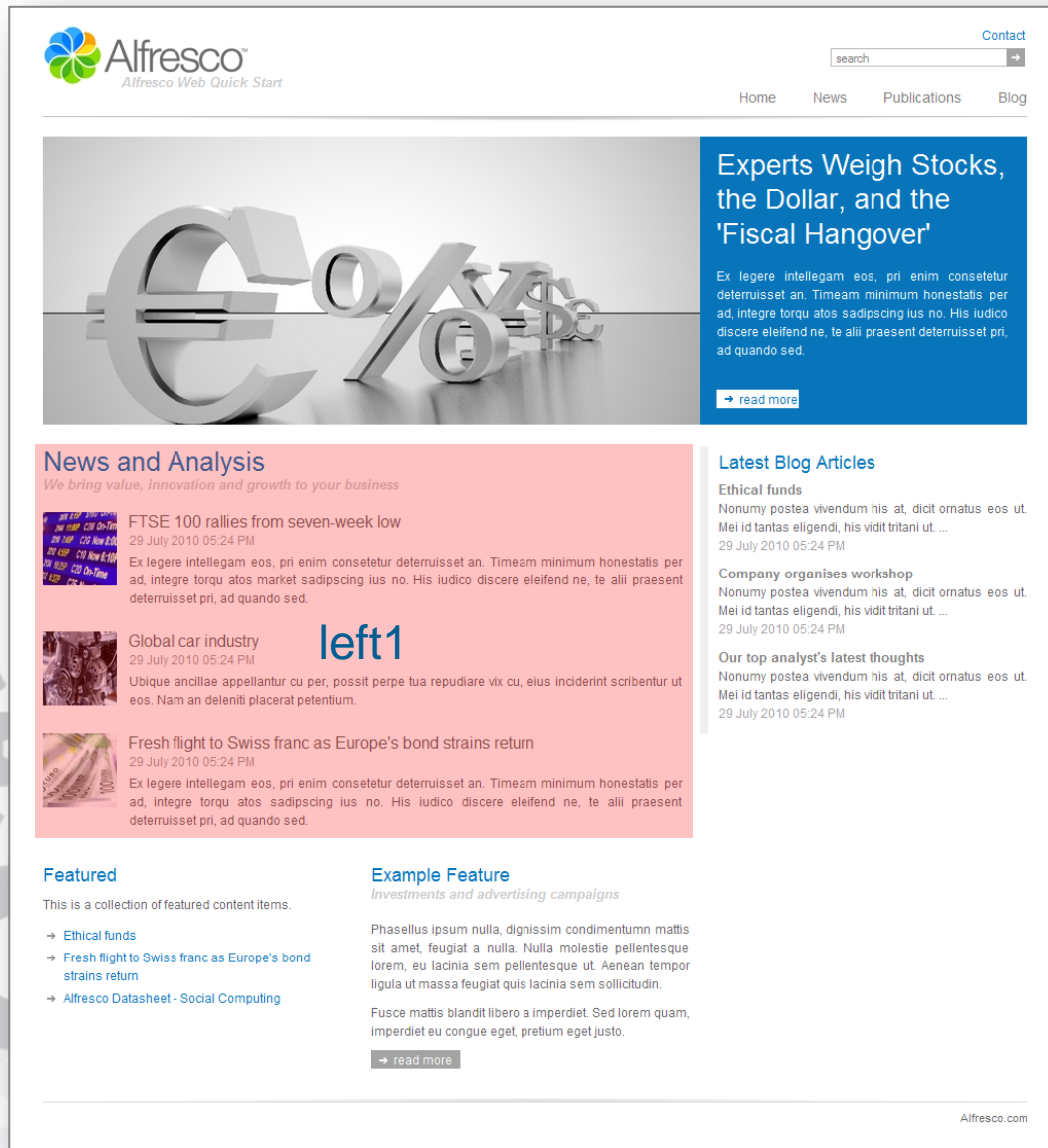
webscripts/list/wide.get.html.ftl

```
<#if articles??>
  <#if articles.title??>
    <div class="clearfix">
      <h2>${articles.title}</h2>
      <#if articles.description??><p class="header-desc">${articles.description}</p></#if>
    </div>
  </#if>

  <div class="interior-content">
    <#if articles.assets?size == 0>
      ${msg('list.none')}
    <#else>
      <ul class="newslst-wrapper">
        <#list articles.assets as article>
          <li>
            <#if article.relatedAssets['ws:primaryImage']??>
              <#assign image=article.relatedAssets['ws:primaryImage'][0]>
              <a href="@makeurl asset=article/"></a>
            </#if>
            <h4><a href="@makeurl asset=article/">${article.title!'no title'}</a></h4>
            <span class="newslst-date"><#if article.properties['ws:publishedTime']??>
              ${article.properties['ws:publishedTime']?string(msg('date.format'))}</#if></span>
            <p>${article.description!'no preview'}</p>
          </li>
        </#list>
      </ul>
    </#if>
  </div>
</#if>
```

The “list/wide” component loads the named asset collection (“news.top” in this example), and then renders information about each asset in that asset collection

The anatomy of a page



Learn More

wiki.alfresco.com/wiki/Web_Quick_Start

forums.alfresco.com/en/viewforum.php?f=52

twitter: @Alfresco, @brianremmington

