In Part 2 of this series, you learn how to detect which eSpots are static and which are dynamic so that the eSpots can be cached appropriately on a CDN (Content Delivery Network). If they are static eSpots then they are consumed by the parent servlet cache and stored on a CDN for faster performance, and if they are dynamic eSpots they are automatically transformed into Ajax requests.

View more content in this series
Introduction

Caching of storefront content on a CDN (Content Delivery Network) is an important part of a sound caching strategy to reduce traffic to the data center and to improve user experience. However, increased personalization drives extra considerations toward a successful CDN caching implementation. In this article, you learn what these considerations are and a guideline is provided outlining how they can be realized with WebSphere® Commerce Aurora storefront with the eSpot (e-Marketing Spot) as an example.

IBM® WebSphere Commerce provides various strategies for caching reusable storefront content. Dynamically rendered HTML pages can be cached as a whole, in fragments, and underlying commands and query results might also be cached. The flexibility of caching enables high cache reuse for different business use cases to maximize caching benefit. However, personalization increases the uniqueness of the content and makes them less static because every user of the site can no longer share them.

The WebSphere Commerce Aurora storefront demonstrates the use of DynaCache, caching servlet’s in an optimistic way whereby caching full page requests and JSP fragments can opt in and out of being part of the full page cache. In WebSphere Commerce, personalized content predominantly lives within eSpot page fragments. Depending on the personalized behavior of the eSpot, you consider it dynamic if it differs between shoppers. If a fragment is dynamic, it opts out of being part of the parent cache. This is done through DynaCache configuration that does not translate directly to a CDN caching methodology.

In these steps, you learn how to detect which eSpots are static versus dynamic so that they can be cached accordingly on a CDN. Static eSpots displays same content to all customers, and these eSpots are consumed by the parent servlet cache and stored on a CDN for faster performance. Whereas, dynamic eSpots display different content whose behavior can depend on the current customer or current context. These eSpots are automatically transformed into Ajax requests. Static and dynamic behavior is determined by the marketing engine when a request is made to retrieve the content. The data that is returned includes an activity behavior flag that indicates whether the eSpot is static or dynamic. An activity behavior of 0 means static, whereas an activity behavior of 1 or 2 means dynamic. This caching solution improves initial page load time and offloads as much content as possible to the CDN. Figure 1 illustrates the proposed solution for caching dynamic eSpots.
This article demonstrates one approach to caching and fragmenting storefront content that is referred to as Automatic determination of dynamic fragments. This approach is a superset of a more common caching strategy whereby developers pre-determine static and dynamic fragments of the page and cache accordingly. The second approach is referred to as Developer pre-determination of dynamic fragments. Listed in table 1 are some pros and cons of both approaches.

### Table 1. Pros and Cons of caching strategy approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
</table>
| Automatic determination of dynamic fragments  | • Business users can use any e-marketing area on a page to run dynamic rules such as customer segment targeting. This allows maximum business flexibility.  
• If a page does not contain dynamic rules, then there is no cache fragmentation because the full servlet result is stored as 1 object. | • Too many dynamic spots on a page, leads to several call backs to the origin that might result in poor performance. |
| Developer pre-determination of dynamic fragments | • Easier to test and understand performance of your pages.  
• Easier to implement and debug as a developer. | • If a page does not contain a dynamic spot, then there is always a callback to the origin. |

### Prerequisites

- This article is based on WebSphere Commerce 7.0.0.8 with Feature Pack 7 Aurora store.
• You must be familiar with the WebSphere Commerce caching policies defined in the defaultcachespec.xml file.
• Ensure that you are familiar with managing layouts in the Management Center Commerce Composer tool.
• Ensure that you have the WebSphere Application Server Cache Monitor up and running.

Transforming dynamic eSpots to Ajax requests for CDN caching

The steps describe adding a new widget to the Management Center Commerce Composer tool, and setting up caching rules for the new widget in your WebSphere Commerce environment. The sample code is an example of a Content Recommendation widget that displays content that is based on dynamic or static rules. If the rule is dynamic, the content is requested from WebSphere Commerce through an Ajax request. If the rule is static, the content is retrieved in line with the page.

1. Ensure that your WebSphere Commerce server is stopped.
2. **Download** the sample code and extract it to a temp directory. For example, `c:\temp`
3. Deploy the sample Content Recommendation widget into your toolkit environment. Before you run the Data Load utility, ensure that you know the identifier of the store you want to deploy the sample widget into. You are prompted to enter the store identifier while you are running the utility.
   **Tip:** The identifier of your store can be found in the STOREENT table.
   a. In a command prompt window, navigate to your WebSphere Commerce Developer bin directory. For example: `c:\WCDE_InstallDir\bin`.
   b. Run the Data Load utility. For example: `dataload c:\temp\Dataload\widget\wc-dataload-widget.xml`.
4. Copy the contents of `c:\temp\LOBTools` to `c:\WCDE_InstallDir\workspace\LOBTools`.
5. Copy the contents of `c:\temp\Stores` to `c:\WCDE_InstallDir\workspace\Stores`.
6. Copy the file `c:\temp\xml\AccessStoreControl.xml` to `c:\WCDE_InstallDir\xml\policies\xml`.
7. Open `AccessStoreControl.xml` and modify the ActionGroup name to match the action group of your store.
8. Load the access control policy. For example: `acload AccessStoreControl.xml`.
9. Define the caching policy for the content recommendation JSP fragment. Add a `<cache-entry>` element for the CDN Content Recommendation JSP fragment to always cache the fragment and consume it with the parent page that it belongs to. Include the following code in the cachespec.xml file:

**Listing 1. cache-entry for content recommendation JSP**

```xml
<cache-entry>
  <class>servlet</class>
  <name>/Widgets-myCompany/com.mycompany.commerce.store.widgets.CDNContentRecommendation/CDNContentRecommendation.jsp</name>
  <!--
  Set both do-not-consume and do-not-cache to false to ensure that the JSP fragment is cached and consumed by the parent page.
  -->
  <property name="do-not-consume">false</property>
  <property name="ignore-get-post">true</property>
</cache-entry>
```
<property name="do-not-cache">false</property>
<property name="save-attributes">false
<exclude>javax.servlet.forward.path_info</exclude>
<exclude>requestURIPath</exclude>
<exclude>requestServletPath</exclude>
<exclude>jspStoreDir</exclude>
<exclude>env_jspStoreDir</exclude>
<exclude>emsNameLocalPrefix</exclude>
<exclude>pageloaded_quickInfoPopupJSPF</exclude>
</property>
<sharing-policy>not-shared</sharing-policy>
<cache-id>
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="langId" type="parameter">
    <required>true</required>
  </component>
  <component id="emsName" type="parameter">
    <required>true</required>
  </component>
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
  <component id="DC_curr" type="attribute">
    <required>true</required>
  </component>
  <component id="DC_cont" type="attribute">
    <required>true</required>
  </component>
  <component id="dontCreateRefreshArea" type="parameter">
    <required>false</required>
    <not-value>true</not-value>
  </component>
  <metadatagenerator>com.ibm.commerce.marketing.cache.EMarketingSpotMetaDataGenerator</metadatagenerator>
</cache-id>
</cache-entry>

10. Add a `<cache-id>` element to cache the ECAction Servlet for the AjaxContentRecommendationView to ensure that the servlet does not have to execute again. When a content recommendation rule is determined to be dynamic, an Ajax request is created to call AjaxContentRecommendationView to get the actual content. The servlet is cached but the underlying JSP is not cached due to the personalized nature of the spot. Include the following code in the `cachespec.xml` file under the ECActionServlet cache entry:

**Listing 2. cache-id entry for AJAX content recommendation servlet**

```xml
<cache-id>
  <component id="" type="pathinfo">
    <required>true</required>
    <value>/AjaxContentRecommendationView</value>
  </component>
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="langId" type="parameter">
    <required>true</required>
  </component>
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
  <component id="emsName" type="parameter">
    <required>true</required>
  </component>
</cache-id>
```
As reference, the URL this cache ID executes against looks like:

```
/storesId=<storeId>&catalogId=<catalogId>&emsName=<emsName>&langId=<langId>&urlLangId=<urlLangId>
```

11. Include a `<cache-entry>` for the Ajax content recommendation JSP fragment in the `cachespec.xml` file to ensure that the fragment is not cached.

```
<cache-entry>
  <class>servlet</class>
  <name>/Widgets-myCompany/com.mycompany.commerce.store.widgets.CDNContentRecommendation/
  AjaxContentRecommendation.jsp</name>
  <property name="do-not-consume">true</property>
  <property name="do-not-cache">true</property>
  <!-- Ideally cache-ids are not needed here since these entries are neither consumed nor cached. But if cache-ids are empty, then WAS may not treat this as a valid cache-entry. -->
  <cache-id>
    <component id="storeId" type="parameter">
      <required>false</required>
    </component>
  </cache-id>
</cache-entry>
```

12. Add the `AjaxContentRecommendationView` in the double-click handler exclusion list to ensure that when multiple requests are made to the same view, some of those requests are not blocked. In the `wc-server.xml`, search for `DoubleClickMonitoredCommands` and add the view in the `excludeCommands` list:

```
<command name="AjaxContentRecommendationView"/>
```

13. Start the WebSphere Commerce server.

Verifying your customization

Complete the following steps to verify the customization:

1. Design a new layout for any page on your site. When you add a widget to the page, you will notice the new CDN Content Recommendation widget in the palette of choices as shown in Figure 2. Add 2 of these widgets to your page and ensure that you add content that is dynamic and another piece of content that is static.
2. Open the WebSphere Application Server Cache monitor to inspect what is being cached. Figure 3 illustrates what is cached if static eSpots are included in a page, while figure 4 illustrates what is cached if you have a static spot and dynamic spot in a page.

**Figure 3. Cached page with two static content recommendation spots**

![Figure 3. Cached page with two static content recommendation spots](image-url)
3. Load the page and inspect the requests. You will see the Ajax call for the dynamic eSpots as shown in Figure 5.
Figure 5. Aurora storefront firebug output of the Ajax call

Conclusion

This article series demonstrates how to detect if an eSpot is dynamic or static and how to cache appropriately with a CDN and DynaCache based on this attribute. This concept can be applied to other types of e-marketing spots.
Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample code for caching strategy</td>
<td>SampleCode.zip</td>
<td>21.3KB</td>
</tr>
</tbody>
</table>
Resources

Learn

- Overview of e-Marketing Spot JSP caching based on activity behavior
- Mastering DynaCache
- WebSphere Commerce Ajax framework
- Setting access control for a page
- Overview of Data Load utility
- In the developerWorks Commerce area, get the resources that you need to advance your knowledge of Commerce products.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools as well as IT industry trends.
- Follow developerWorks Commerce communities.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.

Get products and technologies

- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.
About the authors

Victoria O. Odeyemi

Victoria Odeyemi is a WebSphere Commerce developer at the IBM Toronto Lab, Canada. She has over five years experience developing industry leading e-Commerce software focusing on SEO, business user tooling, content management and storefront development. She holds a Bachelor of Engineering Science degree from the University of Western Ontario and is pursuing a Masters degree in Computer Engineering at the University of Toronto.

---

Marco Deluca

Marco Deluca is a Software Architect with the WebSphere Commerce Development team at the IBM Toronto Lab, Canada. He brings over ten years of progressive experience to the enterprise software domain. Currently, Marco is responsible for architecting, designing and developing solutions for WebSphere Commerce.

---

Andon Salvarinov

Andon Salvarinov is a Managing Consultant with the WebSphere Commerce - Software Services for Industry Solutions, specialising on performance engineering. He has over 15 years of IT experience, with large scale software development as well as IT solution implementations and customer advisory roles.

---

Howard Borenstein

Howard Borenstein is a Software Architect with the WebSphere Commerce Development team at the IBM Toronto Lab, Canada. He has over 23 years of experience in the IT industry in software development tools and enterprise commerce solutions. Currently, Howard is responsible for designing and developing the marketing and promotion components for WebSphere Commerce.
Kevin Yu
Kevin Yu is the Smarter Commerce Performance Architect with over sixteen years’ experience in the IT industry. Kevin has worked on application server, enterprise commerce and database products. He has led multi-disciplinary teams to solve critical customer situations and guided many to achieve new records in transaction volumes and sales on their busiest online shopping days of the year.

Ali Asghar
Ali Asghar is the WW Technical Lead for Smarter Commerce Performance Consulting, with IBM Software Group and a trusted technology advisor to numerous Fortune 500 CIOs. Ali has held several technical leadership roles in his thirteen year IT career, now, with Key focus on improving the IBM Smarter Commerce Eco-System, and enabling well performing and highly scalable eCommerce Solutions for IBM Customers.

© Copyright IBM Corporation 2014
Trademark
(www.ibm.com/developerworks/ibm/trademarks/)