ORACLE

ATG WEB COMMERCE

An Oracle White Paper
January 2012

Creating Mobile Web Applications with Oracle ATG Web Commerce: A Guide to the Mobile Commerce Reference Store (Web)

ORACLE

Introduction

A recent Forrester report (*Mobile Commerce Forecast: 2011 to 2016,* Forrester Research, Inc., June 17, 2011) showed that 24 out of the top 30 online retailers already have an optimized solution for what are deemed in the report to be key mobile devices. Increasingly, customers look to Oracle ATG for guidance on how best to implement such a solution on the Oracle ATG Web Commerce platform.

The purpose of this document is to introduce the Mobile Commerce Reference Store (CRS-M), which is an extension of the existing Commerce Reference Store (CRS). It provides a new UI comprising JSPs optimized for WebKit-enabled mobile browsers, specifically Mobile Safari.

This new UI takes advantage of existing FormHandlers and ATG servlet beans but combines these with HTML 5 and CSS3 features, as well as many WebKit-specific features, to provide a mobile-optimized version of the existing reference application, CRS.

In this document we will discuss:

- What is CRS-M?
- How to install CRS-M
- The overall architecture of CRS-M
- The rationale behind the user experience (UX) design
- Performance considerations

What is CRS-M?

CRS-M is an extension to CRS that provides a set of new JSPs comprising a mobile-optimized UI. CRS-M is a mobile web application—it is not a native iPhone or Android application—such as would be downloaded from the Apple AppStore or the Android Marketplace.

CRS-M is based almost entirely on the existing CRS application's backend (Nucleus components and configuration) with a handful of extensions to facilitate the new UI layout and Site setup. All of CRS-M's JSPs are new, and there are new sub-modules that encapsulate the CRS-M code. These will be discussed in more detail in later sections.

If you are not already familiar with CRS, please refer to the ATG Commerce Reference Store Overview which can be found via the Oracle ATG Web Commerce documentation page located at http://www.oracle.com/technetwork/indexes/documentation/atgwebcommerce-393465.html.

Version Compatibility

CRS-M is compatible with version 10.0.3.1 and later versions of the Oracle ATG Commerce Reference Store (CRS). The CRS and CRS-M integration with Oracle Recommendations On Demand is available in version 10.1 and later versions.

Installing CRS-M

CRS-M is dependent on CRS. Later in this section we will refer to the existing ATG Commerce Reference Store Installation and Configuration Guide. If you are not already familiar with this document, it can be found via the Oracle ATG Web Commerce documentation page.

Configuration Options

Oracle ATG Web Commerce Search

While CRS-M can run without having Oracle ATG Web Commerce Search (ATG Search) installed, it is required in order to see a demonstration of the more advanced faceted search features of the UI. The installation instructions below will indicate the options that you should select to install ATG Search. If you choose to not install ATG Search, you will still see basic repository search functionality demonstrated by CRS-M.

Oracle Recommendations On Demand

While CRS-M can run without having Oracle Recommendations On Demand installed, it is required in order to see personalized recommendations on the CRS-M homepage. With Oracle Recommendations On Demand installed, CRS-M tracks each click that the user

makes, and then uses this information to provide those personalized recommendations. If you choose not to install Oracle Recommendations On Demand, you will still see personalized content on the homepage provided by content targeters. For more information on the Oracle Recommendations On Demand integration with CRS, see the CRS documentation available through the Oracle ATG Web Commerce documentation page.

DedicatedSite Module or UIOnly Module

To enable merchants to implement the mobile strategy that best fits their unique business needs, CRS-M offers a two-pronged approach with regard to multisite configuration.

CRS provides three sites in its out-of-the-box configuration: ATG Store US, ATG Store Germany, and ATG Home. It also provides a different "skin" for each of these.

CRS-M offers demonstrations of two possible approaches to setting up a mobile offering that builds on the existing CRS application and its out-of-the-box site configuration:

- The DedicatedSite Module demonstrates how to set up a new, additional mobile site through SiteAdmin where:
  - Each desktop site (ATG Store, ATG Store Germany and ATG Home) can be given a mobile-optimized version
  - The catalog of each mobile site can be merchandized separately from the catalog of its desktop version
- The UIOnly Module demonstrates how to provide a mobile-optimized UI that is an alternative UI to the desktop version, for visitors to each of the existing sites from mobile devices:
  - The mobile-optimized UI is just that – there is no separate site configuration required through SiteAdmin, but changes to the catalog for desktop visitors always affect the catalog seen by mobile visitors (because it is the same catalog)

CIM Setup and Configuration

CRS-M is dependent on CRS; therefore, the setup and configuration process for CRS-M is very similar to that for CRS. Here we will identify only the installation and/or configuration steps that **differ** from the CRS steps that are detailed in the ATG Commerce Reference Store Installation and Configuration Guide.

Here we will deal with a development setup only, as detailed in the "Development Installation" section of the Installation and Configuration Guide. All steps in that section should be followed for your chosen environment and add-on options, with some additional required in steps the "Configuring ATG Products" section where CIM is used for setup. These additional required steps are:

1. Choosing the Mobile Reference Store Add-On
2. Choosing the Mobile Site Configuration

Commerce Store Addons Menu

In the "CHOOSE ATG COMMERCE STORE ADDONS" menu, when CRS-M is available, an additional menu option for the "Mobile Reference Store" is presented. It looks similar to the following (depending on your installed products):

-------CHOOSE ATG COMMERCE REFERENCE STORE ADDONS :----------
enter [h]elp, [m]ain menu, [q]uit to exit

Choose ATG Commerce Reference Store AddOns :

[1] Storefront Demo Application
[2] International : Enables multi-country and multi-language functionality
[3] Fulfillment
[4] Oracle Recommendations On Demand Integration
[5] ClickToCall Integration
[6] Mobile Reference Store
[D] Done

Select the Mobile Reference Store option **in addition** to the other add-ons that you require.

Mobile Site Configuration Menu

Further along in the CIM setup process, a menu entitled "MOBILE SITE CONFIGURATION" is presented:

-------MOBILE SITE CONFIGURATION----------
enter [h]elp, [m]ain menu, [q]uit to exit

Mobile CRS Site Configuration

[1] UI Only
[2] Dedicated Site

Here option "[1] UI Only" is for customers who do **not** plan to configure their mobile e-commerce offering as a separate Site. The term "Site" in this case refers to a Site that would be managed through the SiteAdmin tool. Customers in this situation may wish to have the mobile version of their website be a mobile-specific presentation layer (i.e. UI) only, on top of their existing website. The "[1] UI Only" option caters for this situation.

Option "[2] Dedicated Site" is for customers who plan to configure their mobile e-commerce offering as a separate Site, i.e. it would be managed through the SiteAdmin tool similarly to how the CRS Home Store and German Store are managed as Sites. The "[2] Dedicated Site" option caters for this situation.

Select either option [1] **OR** option [2] here, but not both.

High Level Architecture

If you are not already familiar with the CRS architecture, please refer to the CRS documentation, particularly the "Commerce Reference Store Overview" and "Commerce Reference Store Configuration and Installation Guide" available through the Oracle ATG Web Commerce documentation page.

Since CRS-M is based almost entirely on the existing CRS application, we will not delve into its details here, but rather explain how CRS-M sits logically on top of CRS, as an extension to it, and the details of those extensions that CRS-M applies to the existing CRS application.

CRS-M makes extensive use of configuration layering to extend existing CRS classes, a practice that is extremely common in ATG applications. It then provides a set of new Mobile-optimized JSPs on top of the extended CRS backend.

"Mobile" Module Structure

CRS-M exists as a new module of the CommerceReferenceStore: "Mobile".

Within the "Mobile" module you will find 2 sub-modules, Mobile.DedicatedSite and Mobile.UIOnly. These 2 sub-modules represent 2 choices that you have for organizing your mobile store or stores. It is not intended that CRS-M be run with both of these sub-modules installed – it is an either/or choice. The configuration instructions for these sub-modules are discussed in the section entitled "CIM Setup and Configuration".

Mobile.DedicatedSite

Mobile.DedicatedSite provides an example of how to set up a dedicated mobile Site so that you can separately manage your mobile site through SiteAdmin. If you are unfamiliar with multisite, refer to the "Multisite Administration Guide" available through the Oracle ATG Web Commerce documentation page.

Mobile.UIOnly

The Mobile.UIOnly sub-module provides an example of how to set up a collection of mobile-optimized JSPs to serve as the UI for users coming to an existing site from their mobile devices. In this case, the mobile-optimized JSP is simply a UI on the existing site.

General Configuration

The following are general configuration points of note – each of them can be configured to your needs.

Access Control Servlet

The standard Oracle ATG access control servlet specifies the pages that, when requested, should cause the user to be redirected to log in. The CRS-M configuration can be found within CRS-M at /atg/dynamo/servlet/dafpipeline/AccessControlServlet.properties. Additionally, a mechanism to disable CRS-M even after it has been included by your CIM setup is provided by /atg/userprofiling/DisableMobileAccess.properties.

Mobile Interceptor

CRS-M aims to achieve the best possible coverage with mobile-specific pages for incoming requests (i.e. the provision of mobile-specific JSPs for requests from mobile browsers). There are 2 elements to the approach that was taken:

1. The structure of the directory which holds the mobile JSPs
2. The interceptor which determines whether an incoming request came from a mobile browser or not

Directory Structure

The directory structure of CRS-M's JSPs mirrors that of CRS as closely as possible, but mobile-specific JSPs are nested under a "mobile" directory. The name of this directory is configured as the mobile path prefix in atg/store/StoreConfiguration.properties.

As an example, the product details JSP for CRS is located at /browse/productDetail.jsp in the Storefront module, while the product details JSP for CRS-M is located at /**mobile**/browse/productDetail.jsp, also in the Storefront module. Since the designated mobile path prefix for CRS-M is configured to be the string "mobile" by StoreConfiguration.properties, there is a clear mapping from CRS JSP to equivalent CRS-M JSP, i.e. the CRS path should be prepended with the designated mobile prefix ("mobile") to determine the CRS-M path for a given page request. You should configure your designated mobile prefix to match your directory structure should you choose to take this approach for your implemention.

Interceptor

A configuration layer was added in CRS-M in order to forward requests for non-mobile CRS URLs, made by mobile user-agents (browsers), for processing by CRS-M. This ensures that mobile user-agents are provided with a mobile-specific presentation of each requested page.

The user-agent detection and request forwarding is done on the server side by a new tail pipeline interceptor—atg.projects.store.mobile.link.MobileBrowserInterceptor.java—which is configured into the Oracle ATG tail pipeline. Its corresponding configuration is located at /atg/dynamo/servlet/dafpipeline/MobileDetectionInterceptor.properties. This new interceptor's position in the tail pipeline is determined by /atg/dynamo/servlet/dafpipeline/TailPipelineServlet.properties and this position is important to ensure that multisite features function correctly, i.e. it should always be listed before the VirtualContextRootInterceptor and the WebAppLayeringInterceptor.

The interceptor performs simple Java regular expression pattern matching on the incoming URL using the configured mobile path prefix (from atg/store/StoreConfiguration.properties) to determine whether:

- The URL is already a mobile URL and hence can pass through untouched, or
- The URL is a non-mobile URL and hence the type of user-agent that it came from should be checked
    - If that user-agent matches our definition of a mobile user-agent, forward the request to an equivalent but mobile-specific page
    - If that user-agent does not match our definition of a mobile user-agent, let it pass through

It is important to note that the interceptor does not take any action on requests whose URLs already contain the designated mobile prefix (i.e. the URL is already a mobile URL), or that match the file types listed by the "excludedFileTypes" property of the interceptor. Both the mobile prefix and the excluded file types can be easily configured to meet your needs.

To determine if the user-agent of the incoming request is a mobile user-agent that we want to forward, a UserAgentType class is defined (atg.projects.store.mobile.link.UserAgentType.java). It has a "patternStrings" property which, in the case of CRS-M, is configured to declare patterns matching a specific set of user-agent strings. At the time of writing these are:

.*Version/4[\\.0-9]* Mobile[/0-9A-Z]* Safari.*
.*Version/5[\\.0-9]* Mobile[/0-9A-Z]* Safari.*
.*AppleWebKit/5[.0-9]*.*Mobile.*

These patterns can of course be configured to your needs.

Mobile Form Handler Extensions

CRS-M extends 2 existing CRS form handlers in order to provide functionality required specifically for the mobile UI design. These 2 form handlers are:

1. MobileStoreProfileFormHandler.java which extends StoreProfileFormHandler.java and is configured by MobileB2CProfileFormHandler.properties
2. MobileBillingInfoFormHandler.java which extends BillingInfoFormHandler.java and is configured by MobileBillingFormHandler.properties

The main purpose of these new form handlers is to handle the creation and editing of credit cards in CRS-M. Traditionally in the CRS application, a credit card and billing address have been created on the same page or screen of the UI, requiring just one form submission. The CRS-M design called for the credit card and billing address creation to happen on separate screens, hence the need for the new form handler logic. The task of splitting the credit card and billing address creation across 2 screens is accomplished by storing the (encrypted) credit card information into a temporary session bean until the billing address information has been entered. If successfully validated, then the credit card and billing address are associated with one another and committed to the repository.

International Configuration

The profile tools configuration file within the Mobile.International module ensures correct internationalization behavior within the MobileProfileTools component (/atg/projects/store/mobile/userprofiling/MobileInternationalStoreProfileTools.java). All other internationalization support is provided by CRS. For more information about internationalization features demonstrated by CRS, see the ATG Commerce Reference Store Overview, which can be found via the Oracle ATG Web Commerce documentation page.

Recommendations Configuration

The recommendations configuration file within the Mobile.Recommendations module simply registers the search results pages of CRS-M with the recommendations auto-tagging logic. All other auto-tagging for recommendations support is provided by CRS. For more information about the Oracle Recommendations On Demand functionality demonstrated by CRS, see the ATG Commerce Reference Store Overview, which can be found via the Oracle ATG Web Commerce documentation page.

CRS-M JSP File Structure

The CRS-M JSP files exist in the Store.Storefront sub-module in store.war/mobile.

The directory structure under store.war/mobile intentionally mirrors that of CRS. An example of the use case that this setup caters for might be described as follows: A retail customer receives an email from a friend containing a link to a product details page. The retail customer (recipient of the email) then clicks that link, e.g. http://localhost:8180/crs/storeus/browse/.... from his or her mobile device. When this happens, the retailer typically wants to take that customer from the desktop version of that product details page to the mobile-optimized version of that page in order to provide the best possible user experience. This mirroring of directory structure combined with simple mobile browser detection facilitates this forwarding action. This is discussed in more detail in the section entitled "Mobile Interceptor" within "General Configuration".

CRS-M External Media Location

External media items (images) specifically for CRS-M are located at Store.Storefront/j2ee/storedocroot.war/content/mobile/images.

Low Level Architecture – How Pages Work

Here we will examine in more detail how pages are composed, focusing on areas of interest including the homepage, product details page and search pages.

Homepage

Overview

The homepage of CRS-M (Storefront/j2ee/store.war/index.jsp) takes advantage of touch gestures as well as Oracle ATG Web Commerce personalization features to demonstrate one approach to providing users with an interactive and personalized homepage experience.

The page is divided into 2 distinct sections, lying between the standard header and footer. Each of these sections slides or scrolls horizontally. Figure 1 shows the top 2/3 of the page containing a **promotional content item** carousel or "slider", and the remaining 1/3 of the page containing a **product** carousel or "slider".

The promotional content items and products have a parent-child relationship: each promotional content item in the top section of the page has a set of associated products that appear in the carousel below it in the lower section of the page. CRS-M provides several [promotional content, product set] pairs that are defined by content targeters and utilize the sample user segments provided by CRS.

If you are not already familiar with content targeters, please see the Oracle ATG Web Commerce Personalization Guide available through the Oracle ATG Web Commerce documentation page. As with all content targeters, those provided by CRS-M can be customized through the BCC.

Figure 1 - Homepage

Architecture

Personalized Content

The personalized content on the homepage is retrieved by executing server-side targeters. The targeters are located alongside the existing CRS targeters in the personalization UI. They are each prefixed with "Mobile":

- MobilePromotionParent1
- MobilePromotionParent2
- MobilePromotionParent3
- MobilePromotionChild1
- MobilePromotionChild2
- MobilePromotionChild3

These targeters form three pairs. Each pair is made up of a parent that represents the PromotionalContent items in the top carousel and a child that is made up of a set of products that are related to that PromotionalContent item, displayed in the bottom carousel. An example of such a pair would be MobilePromotionParent1 and MobilePromotionChild1, whose items are displayed in Figure 1.

The pairs will return different items for the different user segments that are already configured for CRS. For example, an anonymous user might see PromotionalContent for free shipping, homeware items, and featured products, while a user belonging to the "Fashionista" segment may see recommendations from Oracle Recommendations, women's clothing items, or other items.

Fetching Personalized Content

Several approaches are available to generate/fetch the promotional content and product data from the targeters for use on the homepage of CRS-M:

- The Oracle ATG Web Commerce REST module could be used to call methods on, or to fetch, custom server-side components that would aggregate the data required for the homepage into the required response format. Making these calls through REST could automatically format the output as JSON. These components could use the OOTB tools such as TargetingServices to easily generate targeting results.

- A combination of TargetingServices and REST features such as filtering, aliasing and PropertyCustomizers could be used to achieve the same result.

- JSPs that generate JSON from ATG servlet bean responses using the JSON tag library could be utilized. This technique facilitates the reuse of OOTB servlet beans, as well as any custom servlet beans and/or page code that may already exist.

CRS-M demonstrates JSP generation of JSON approach. Your method of choice however, should be determined based on your specific business needs and existing or future application architecture.

Structuring the Carousels

Each carousel on the homepage can be broken down into 5 basic components:

- An outermost element representing either the top 2/3 of the page or the bottom 1/3 of the page—mobile_store_homeTopSlot or mobile_store_homeBottomSlot—respectively

- Inside this, an inner element that will "slide" - either mobile_store_homeTopSlotContent or mobile_store_homeBottomSlotContent
- Inside that, another containing elements of class "cell", representing the individual items - either promotional content items or individual products
- The base JavaScript slider object that provides the sliding effect
- Also inside mobile_store_homeTopSlotContent or mobile_store_homeBottomSlotContent, customized content contained in another

In its basic form (without styling or content), the homepage when rendered would appear as follows, where there can be an arbitrary number of elements of class "cell":

```
<div id="mobile_store_homeTopSlot">
  <div id="mobile_store_homeTopSlotContent">
    <div class="cell"></div>
    <div class="cell"></div>
    <div class="cell"></div>
  </div>
  <div id="mobile_store_circlesContainer"></div>
</div>
<div id="mobile_store_homeBottomSlot">
  <div id="mobile_store_homeBottomSlotContent">
    <div class="cell"></div>
    <div class="cell"></div>
    <div class="cell"></div>
  </div>
  <div id="mobile_store_homeBottomSlotProductDetails"></div>
</div>
```

Component 1 – Outermost container

Component 1 above—the outermost element—allows each slider and its corresponding page elements to be cleanly divided and defined. Sliders can be added and removed independently of one another.

Component 2 – Inner container

Component 2 above—the inner element—is used by the JavaScript object representing the carousel to provide the sliding effect by changing the relative position of this within its container. As the user swipes to the left, the "left" attribute of this decreases based on:

- the duration for which the finger is on the screen, and

- the distance for which the finger travels

The JavaScript code uses the distance and duration information to calculate the momentum of the carousel and to calculate which cell the carousel should settle on as the central item when the swipe is complete.

Component 3 – Cell elements

Component 3 above—the elements of class "cell"—each represent a section (i.e. cell) of the carousel. These elements can contain images, links, or any other elements that are needed. The JavaScript object that represents these carousels—the slider (item 5)—looks for these elements in order to add CSS classes and attributes to the cells. These attributes and classes ensure that the cells are correctly and statically positioned inside their container. It is important to understand that when the user slides a finger on the screen, these cells do not move independently but as a single group that slides back and forth. This grouping is the function of item 2 above.

Component 4 – The slider

Component 4 above – the slider itself – is the base JavaScript slider that provides methods for sliding/swiping between the cells. The definition of this object can be found at /Storefront/j2ee/store.war/mobile/js/atg/sliders.js. Each of the 2 carousels on the homepage is an instance of this slider, and 2 customizations of the base slider are demonstrated here.

Component 5 – Custom content

Component 5 above demonstrates how to customize the base slider to add additional/specific content. These customizations will be discussed in more detail in the "Extending the Sliders" section.

Populating the Sliders with Data

On document ready, the homepage uses jQuery to request the JSON targeter results from the JSP pages. You can follow the execution flow starting with the "loadHomepage" JavaScript function call in index.jsp. The JavaScript code uses a predefined jQuery HTML template to convert the JSON into HTML. That HTML will be inserted into the page.

The advantage of using a template is that it can be sent in the initial request and it cuts down on the size of the payload that the targeting JSP files need to send. This also makes it easier to switch between mechanisms for fetching data, i.e. from JSP files that produce JSON to components invoked through REST that produce JSON. The client is expecting JSON in either case; however, the format should remain the same.

Performance Considerations

These PromotionalContent items have been specifically created for presentation on mobile devices. There is no text embedded in the promotional content item images. PNG images with translucent backgrounds of the same dimensions overlay the images rather than embedding text. This was done in order to attain a higher level of compression without blurring the text. While this may not appear to be a best practice (i.e. images of text rather than actual strings) it allows for higher quality, crisper text and makes it possible to align the text over the images in exactly the required format for the cleanest appearance and hopefully the best first impression to users of the homepage.

The products and promotional content items are stored in the browser's HTML 5 sessionstorage cache. The cache is currently used to store only the products that are not currently in view and to redraw the sliders during orientation changes. The redraw is needed in order update properties on the JavaScript slider instance that change due to the change in screen width, which is in turn due to the orientation change.

This caching is of course a performance optimization, and in the current implementation the page empties the cache on every load. However, this functionality could be extended to allow more aggressive caching between browser sessions for better homepage performance, should a particular case be deemed suitable for such behavior.

While jQuery and jQuery templates were used in the creation of the homepage, none of the common/most popular third party touch-specific frameworks were used in the creation of the homepage carousels, however. Not choosing a particular touch framework frees implementers to use whichever mechanism is most suitable to their specific needs, while providing an example of what can be done independently of such frameworks.

Extending the Sliders

Some configuration options have already been mentioned in the preceding sections with regard to caching using HTML 5 session storage, modifying the personalized content through the appropriate targeters and adding additional sliders. Here we will explore the details of the sliders' implementation that are most pertinent to customization of the sliders.

The JavaScript slider object contains all methods needed for a simple sliding carousel. Provided that the slider's HTML is structured correctly, very little code is required to create new carousels. Just one function needs to be implemented in a new JavaScript literal notation object: "createSlidePanel" in sliders.js.

The slider object code takes advantage of a new feature in ECMAScript 5 called Object.create(). This method allows the creation of an instance of both the base JavaScript slider object and a customized object without code duplication. If one type of carousel already exists, just one call to Object.create() is required for each instance on the page. Carousels can be of different sizes and can contain different numbers of objects.

CRS-M demonstrates two extensions to the base slider functionality – the promotional content item carousel (the top 2/3 of the page) and the product carousel (the bottom 1/3 of the page).

For the promotional content item carousel, a element was added to indicate the current page of the carousel and the total number of pages in the carousel. This element has an id attribute of **mobile_store_circlesContainer** and is updated when the promotional content items carousel comes to a stop (i.e. on the touch end event).

For the product item carousel, a element was added to display the product information of the product located in the center of the carousel. This element has an id attribute of **mobile_store_homeBottomSlotContent** and is effectively injected with the information for the product that is located in the center of the carousel when it comes to a stop.

For both carousels the CSS attribute "display" is set to "none" for items that are not on the screen so that a screen reader will not read off-screen content. To facilitate customizations such as this, the slider allows a JavaScript callback function to be passed to it, which will be invoked after the touch move or touch end event occurs (see the "postTouchEnd" function in sliders.js). If no such customization is required, no callback function need be passed to the slider.

A final note about the homepage is that it currently does not have a mechanism to handle the case where HTML session storage is not available. Such behavior would be encouraged for mobile websites targeting a wider range of devices and browsers than those targeted by CRS-M.

Search

"Simple" Search

As mentioned in the Configuration Options section, if ATG Search is not installed, users of CRS-M will still be able to search the product catalog using the **/atg/commerce/catalog/ProductSearch** component, but will not be presented with refinement options. All JSPs used by "Simple" Search are located in the **/Storefront/j2ee/store.war/mobile/search/**directory. Figure 2 below shows a sample set of results on the simple search results screen. For more information on "Simple"

Search, see the section titled "Catalog Searching" in the ATG Commerce Guide to Setting up a Store, available through the Oracle ATG Web Commerce documentation page.

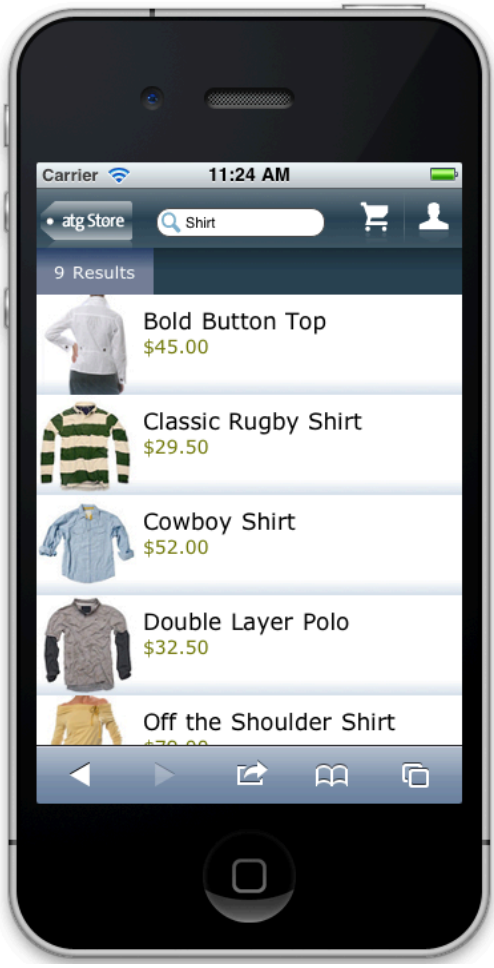

Figure 2 – "Simple" Search Results Screen

Paging

Search results are paged based on the **defaultPageSize** property of the **/atg/multisite/Site** component. If this property is not configured, the page size defaults to 6. When a user executes a search request that results in two or more pages of results, a "Show X More" area appears at the bottom of the first page of search results, where X is the smaller of remaining search results and page size. When the user clicks on this area, an AJAX request retrieves the next page of results and inserts them below the current page of results. If there are yet more results to display, another "Show X More" entry will be displayed below the new page.

ATG Search

As mentioned in the Configuration Options section, when ATG Search is installed in the current environment, search requests will be routed through the ATG Search Engine using the **/atg/commerce/search/catalog/QueryFormHandler** component. All JSPs used by ATG Search are located in the **/Storefront/j2ee/store.war/mobile/atgsearch/** directory. For more information on how ATG Search processes search requests, see sections 2 and 3 of the ATG Search Query Guide, available through the Oracle ATG Web Commerce documentation page.

Paging

ATG Search uses the same paging mechanism as "Simple" Search, described above.

Facets

CRS-M uses the facets that are included in the stock CRS data imports to provide search refinement functionality. These facets are used to sort search results based on the properties of items in the result set, e.g. price, feature, or color. Figure 3 shows the search results screen with the "Refine" option that is displayed when ATG Search is installed.

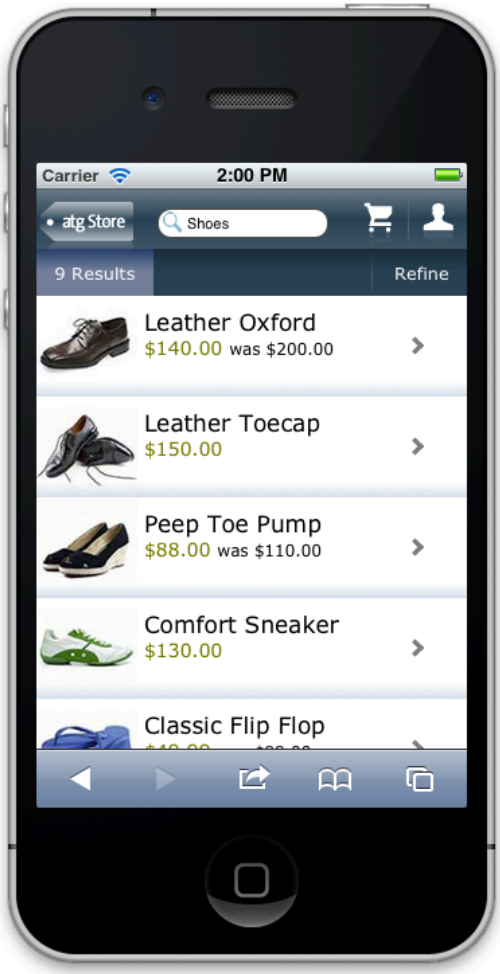

Figure 3 - "ATG Search" Results Screen with "Refine" Option

Differences between CRS and CRS Mobile with regard to facets

Although CRS Mobile uses much of the same page logic as CRS to display facets, the approach to filtering search results based on selected facets is fundamentally different. In CRS, search results are refined iteratively. When the user selects a facet, the search results and facets are automatically refreshed to reflect the new selection. This means that at any given time, the user only sees the facets that apply to the current result set. Any further refinements apply to the current result set, not the initial search results. Due to the fact that the list of available facets is refreshed upon every new selection, facets that do not apply to the current results will not be displayed. Consequently, it is impossible to select an invalid combination of facets that would lead to an empty result set.

CRS-M, on the other hand, modifies this behavior in order to minimize the number of round trips to the server in a small space of time. It allows the user to select multiple facets simultaneously and always displays the list of available facets for the initial result set.

Therefore faceting requests are only issued when the user touches the "Done" button, instead of every time a new facet is selected.

It also means that it is quite possible for the user to select an invalid combination of facets for which there are no results. Presenting these alternative approaches across CRS and CRS-M allows implementers to decide on the best approach for their situation. Table 1 below shows the differences between the search and facet results using the search term "shoes" in both CRS and CRS-M.

TABLE 1 COMPARISON OF CRS AND CRS-M SEARCH FOR "SHOES"

CRS		CRS-M	
Results (9)	Facets (3)	Results (9)	Facets (4)
Leather Oxford Leather Toecap Peep Toe Pump Comfort Sneaker Classic Flip Flop Classy Slingback Elegant Sandal Varsity Trainer Leather Slip-ons	Price: $25-$50 $50-$100 $100-$150 $150-$250 Size: 5 6 7 8 9 10 11 12 Color: Black Blue Brown Green Red White	Leather Oxford Leather Toecap Peep Toe Pump Comfort Sneaker Classic Flip Flop Classy Slingback Elegant Sandal Varsity Trainer Leather Slip-ons	Category: Shoes Price: $25-$50 $50-$100 $100-$150 $150-$250 Size: 5 6 7 8 9 10 11 12 Color: Black Blue Brown Green Red White

An item of note in Table 1 is the addition of the "Category" facet in CRS-M. This facet is hidden by default in CRS search results, and instead is reserved for category browsing. Because CRS Mobile does not currently feature category browsing, the "Category" facet is made visible during search, albeit limited to top-level categories (those whose parent is the root category).

Table 2 below shows the results of refining the search shown in Table 1, using the color facet with a value of "brown". Figure 4 shows how the facets are displayed to the user.

TABLE 2 COMPARISON OF CRS AND CRS-M SEARCH FOR "SHOES", FACETING ON COLOR "BROWN"

CRS		CRS-M	
Results (3)	Facets (3, limited values)	Results (3)	Facets (4, unchanged from Table 1 above)
Leather Oxford Leather Toecap Leather Slip-ons	Price: $100-$150 $150-$250 Size: 8 9 10 11 12 Color: Brown	Leather Oxford Leather Toecap Leather Slip-ons	Category: Shoes Price: $25-$50 $50-$100 $100-$150 $150-$250 Size: 5 6 7 8 9 10 11 12 Color: Black Blue Brown Green Red White

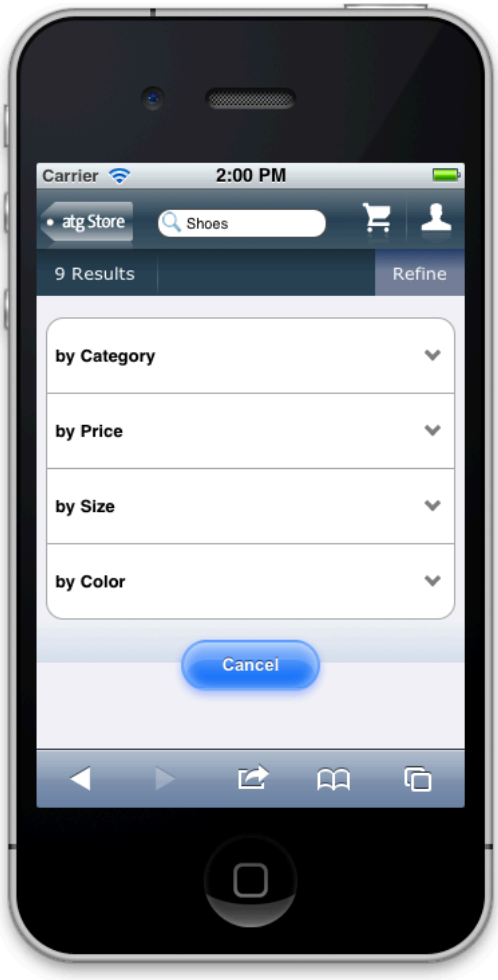

Figure 4 - CRS-M Facets include the Category Facet

Submitting a Refinement Request – The Facet Trail

ATG Search utilizes the **facetTrail** parameter of the **/atg/search/repository/FacetSearchTools** component to keep track of currently selected facets and filter search results. In CRS, manipulating the facet trail is done entirely on the server in page code. Due to the fact that selecting a facet triggers an immediate update, each facet link has a resulting **facetTrail** string constructed with the **CommerceFacetTrailDroplet**. When clicked, the link updates **FacetSearchTools** to use its string.

CRS-M requires a different approach. The state of the **facetTrail** is indeterminate at any given moment due to the possibility of multiple facet selections, so the **facetTrail** string

must be constructed by the client and submitted to the server. On page load, the current **facetTrail** string is parsed into a JavaScript object which then monitors the state of the **facetTrail** as the user makes facet selections. When the user touches the "Done" button to submit the search refinement request, the JavaScript object constructs the appropriate **facetTrail** string and adds it to the request URL.

Submitting a Refinement Request – AJAX

Once the **facetTrail** string has been constructed, a refinement request is submitted to the server using jQuery's **ajax** method. ATG Search then processes the request and renders the refined results via JSP (see **atgsearch/gadgets/searchResultsRenderer.jsp**). The resulting markup is inserted into the DOM using jQuery, replacing previous search results. The user may then continue refining his/her search until finding the desired product.

Submitting a Refinement Request – No Results

Due to the fact that available facets are updated only upon submitting a refinement request, it is possible to select a combination of facet values that leads to an empty result set. When this happens, the user is presented with the option to either clear all current facet selections or further adjust them. The former option re-submits the original search request with an empty **facetTrail**, while the latter opens the facet selection pane. Figure 5 shows the empty result set screen.

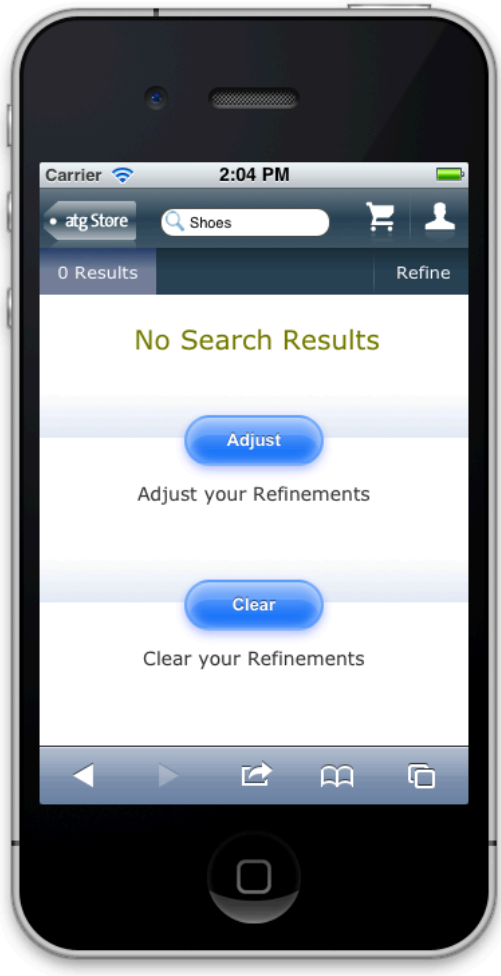

Figure 5 - Empty Result Set Screen

Shopping Cart

The CRS-M shopping cart page contains the information and features common to most cart pages. As discussed in the "User Experience Design" section, it was designed to allow individual rows in the cart (representing products) to be expanded downwards when the user touches them. When a row is touched and it expands downwards, the user is presented with several options:

1. Email the product to a friend
2. Remove the item from the cart
3. Change the item in the cart

In addition to displaying the items in the cart as these expanding rows, the order summary information is displayed, as well as a text box for inputting a coupon code.

Structure

The top-level cart JSP can be found at **/Storefront/j2ee/store.war/mobile/cart/cart.jsp**. It caters for:

1. An empty cart (emptyCart.jsp)
2. A cart containing one or more items (cartContent.jsp).

The cart.jsp file also contains the code that creates the "modal" effect when the user chooses to either email an item or remove an item from the cart. The associated CSS and JavaScript can be found in **/Storefront/j2ee/store.war/mobile/css/cart.css** and **/Storefront/j2ee/store.war/mobile/js/atg/cart.js** respectively.

Empty Cart

In order to make an empty cart page more engaging to users, a horizontal slider similar to that on the homepage is displayed towards the bottom of the cart page. It contains "featured products", which are products that are returned by the existing CRS featured product targeters. They are displayed in a "slider" which is a new instance of the "slider" used on the homepage. Figure 6 shows an empty cart with the featured products slider.

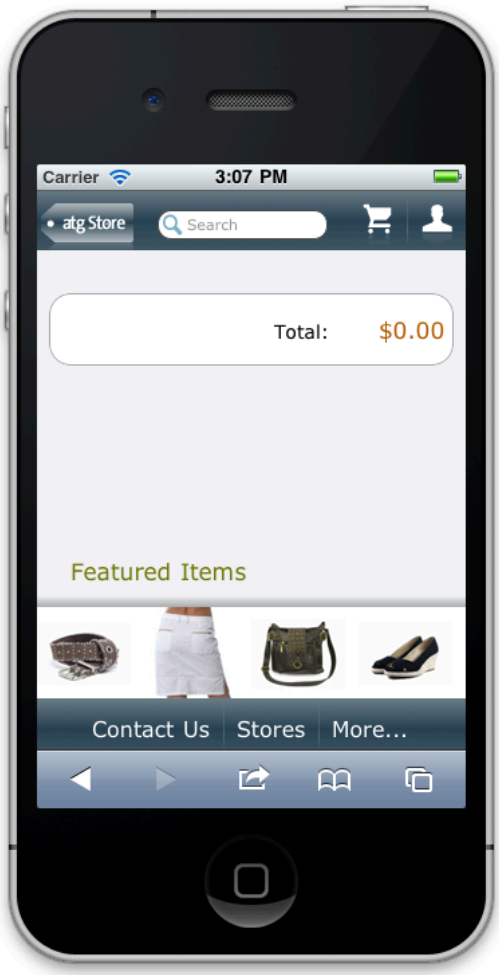

Figure 6 - Empty Cart Page with Featured Products Slider

While inspecting cart.js and other JavaScript files in CRS-M, it is worth noting that the CRS-M build script (not distributed as part of CRS-M) combines individual JavaScript files into one "mobile.js" file. This is discussed later in "Performance Considerations", but the rationale is simply to minimize the number of server calls for data such as JavaScript and CSS files, hence improving the performance of the application. A consequence of this is that code in cart.js has access to variables declared in other JavaScript files that are combined into mobile.js as part of the CRS-M build. The "sliders" variable used in the creation of the featured products slider is one such example.

Cart Items

In the case where the cart contains one or more items, cartContent.jsp uses **/Storefront/j2ee/store.war/mobile/cart/gadgets/cartItems.jsp** to display the items in that cart. This in turn uses **/Storefront/j2ee/store.war/mobile/cart/gadgets/cartItem.jsp** to display individual items in the cart. Since CSS and JavaScript are externalized as much as possible here, as in all areas of CRS-M, the details of how the data is displayed once it has been retrieved using traditional page code can be found in cart.css and cart.js. The **mobile_store_cartItem** CSS class is responsible for styling an item in the cart, while the **initCommerceItemBox** JavaScript function makes an item in the cart (a "row" of the cart) "touchable".

Actions on a Cart Item

When the user touches an item in the cart, the row containing that item expands downwards to become slightly larger, and the image of the item is enlarged and moved to the background. Additional options for emailing this item to a friend, removing this item from the cart, or changing this item are also presented when the cart item is touched.

If the user chooses to either email the item or remove the item from the cart, the chosen option becomes modal in the foreground while the background is darkened. As mentioned above, the code for this behavior is located in the top-level cart.jsp.

If the user chooses to change the item, he/she is returned to the product details page and that particular item is displayed. Once a change has been made, the button text changes to "Back to Cart", and touching this button will take the user to the cart page once again where the most recent changes will be reflected. Figure 7 shows a cart containing 2 items, one of which has been selected for editing.

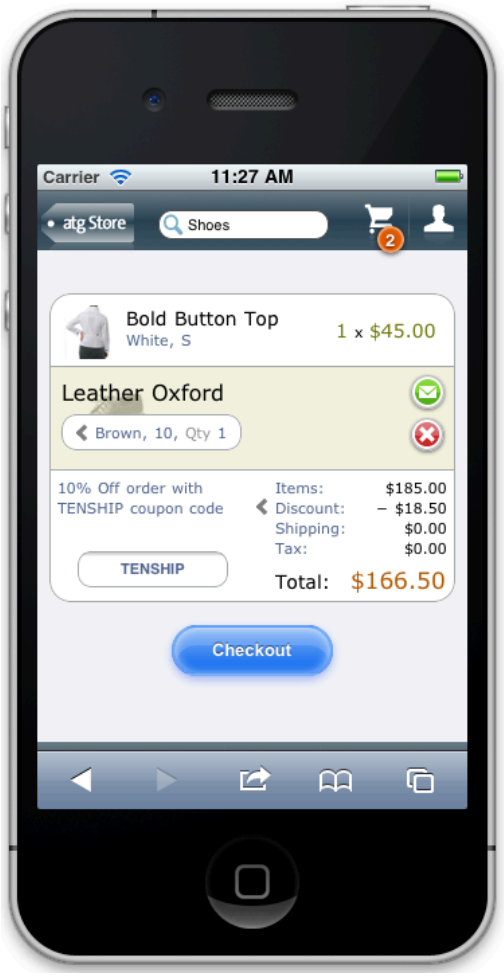

Figure 7 – Ready to edit an item in the cart

Figure 8 shows the modal "Delete" popup that is displayed when the user touches the Delete button.

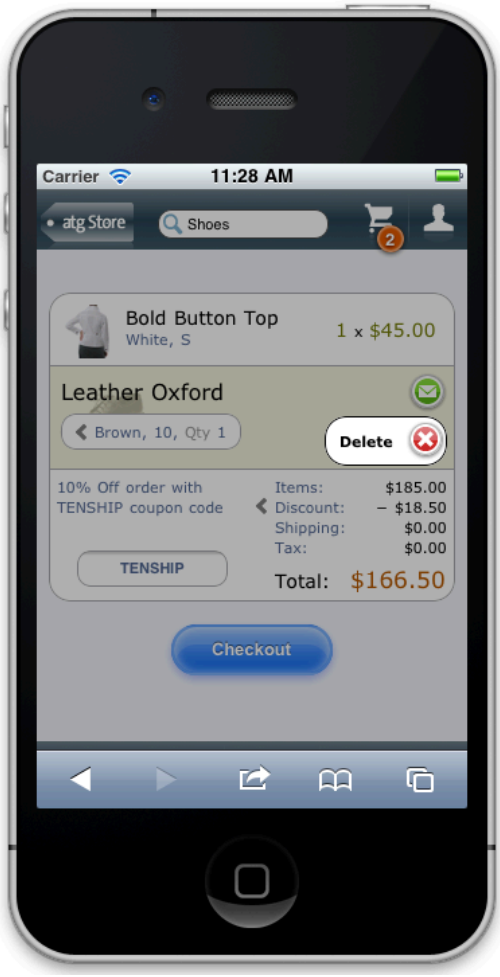

Figure 8 - The modal "Delete" popup

Order Summary Information

Displayed below the cart items on the cart page is the order summary information, such as the cart/order total, discounts applied, shipping cost, taxes etc. In Figures 7 and 8, the coupon "TENSHIP" has been applied to the order. The logic for displaying this summary information is encapsulated within **/Storefront/j2ee/store.war/mobile/cart/gadgets/orderSummary.jsp**, which in turn includes **/Storefront/j2ee/store.war/mobile/cart/gadgets/promotionCode.jsp** that contains the logic for applying a coupon code to the order.

Not Yet Supported

As mentioned in "Feature Comparison with CRS", gift wrap is not supported by CRS-M in this release. This means that the option to add gift wrap is not presented to the user during the checkout flow. There are 2 areas of CRS-M for which this has consequences:

1. Viewing an order through order history: the user is prompted to go to the "full site" (CRS) to view that order
2. Viewing a cart that already contains gift wrap: gift wrap is not displayed on CRS-M's cart

Should a user start adding items, including giftwrap, to his/her cart via CRS, then switch over to CRS-M to view the cart before completing the checkout process, gift wrap items will not be displayed on the CRS-M cart.

Product Details

The product details page of CRS-M displays a large product image that can be further enlarged by touch, dropdown menus for SKU selection, a product description and a horizontal slider containing related products, where available. Figure 9 shows the product details page displaying a clothing SKU.

Given its purpose, the product details page should be flexible enough to display many different types of product and SKU. Since it is an essential component in the flow of a commerce application, additional goals set out for the page were for it to be intuitive and performant. The task of achieving these goals presented an interesting design choice: while the general product data needed to be available upon page load (performant), SKU configuration information (price, availability) was only relevant once that SKU had been selected (intuitive). In keeping with CRS's design, the initial design iteration called for loading such information via AJAX only *after* a SKU had been selected by the user.

However, this became a performance issue in the mobile environment where round trips to the server are often the most expensive operation. While waiting for data from the server on SKU selection, the UI of the product details page would be in an indeterminate state until the AJAX request was complete. For this reason it was decided to use a design similar to the homepage whereby all SKU data for the given product was fetched on the initial page load. This would increase initial load time slightly but eliminate awkward UI pauses during SKU selection, caused by multiple server round trips.

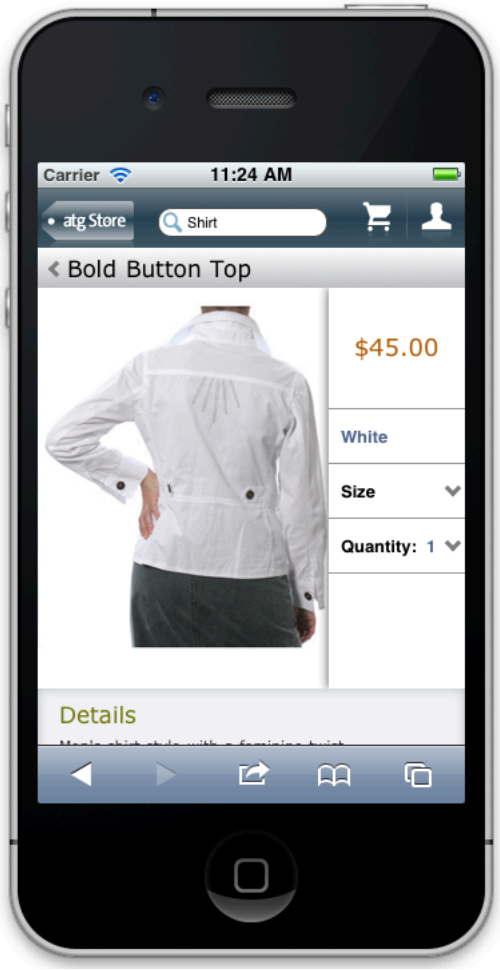

Figure 9 - Product Details page displaying Clothing SKU

Architecture

Client-Server Interaction

As discussed above, the product details page was designed to minimize the number of round trips to the server. All information pertaining to a given product such as prices and features of the product's child SKUs, is loaded immediately with the rest of the page. Subsequent round trips to the server require the user's input and as such cannot be further optimized, e.g. adding an item to the shopping cart.

Product Configuration Data

All data associated with a product's child SKUs is rendered by getSkuJSON.jsp. This data is then processed during the jQuery 'ready' event, and transformed into event handlers that close over the SKU data and DOM elements that will be modified as the user makes their selections. The resulting functions are highly optimized, as any calculations that do

not depend on the execution context have been abstracted to the 'ready' event and bound in local scope. Figure 10 shows the same product while the user is selecting a size.

Figure 10 – Selecting a Size

Page Hierarchy

The user arrives at the product details page by clicking a link to a product. When such a link is generated, it points to one of four template pages:

1. productDetailColorSizePicker.jsp: for products with color and size properties
2. productDetailSingleSku.jsp: for products with only a single SKU
3. productDetailMultiSku.jsp: for products with multiple unique SKUs
4. productDetailWoodFinishPicker.jsp: for products with the "wood finish" property

Each of these template pages is responsible for deriving an appropriate set of parameters that are passed to productDetail.jsp and used to render the page:

- product: the product being viewed (required)

- picker: the appropriate picker for the product (required – see section "Pickers – JSP")
- selectedSku: the currently selected SKU
- selectedQty: the currently selected quantity
- selectedColor: the currently selected color
- selectedSize: the currently selected size
- availableSizes: all available sizes for the product
- availableColors: all available colors for the product
- oneSize: true if the product has exactly one size
- oneColor: true if the product has exactly one color
- ciId: the commerce item being edited

The only required parameters are product and picker; the rest may be derived depending on what is appropriate for the given product. For example, productDetailColorSizePicker.jsp derives all of the parameters listed above, whereas productDetailMultiSku.jsp only derives product, picker, selectedSku, selectedQty, and ciId, as those are the only parameters that pertain to multi-SKU products. Additional template pages and their required parameters could be easily added here to customize or modify CRS-M's product details page to your specifications.

Once the correct parameters have been passed to productDetail.jsp, it utilizes the mobilePageContainer tag (see "Common Elements") to include the SKU data (getSkuJSON.jsp), page content (productDetailsContainer.jsp), out-of-stock notification form (emailMeForm.jsp), related items slider (relatedProducts.jsp), and—depending on the context—either the add-to-cart form (addToCartForm.jsp) or update cart form (updateCartForm.jsp).

Figure 11 shows the product details page displaying a single SKU item, rendered by productDetailSingleSku.jsp.

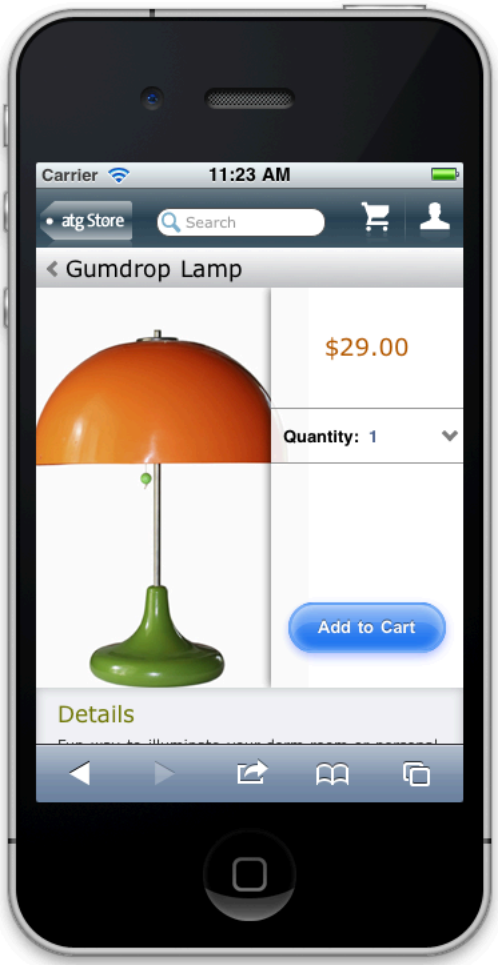

Figure 11 - A Single SKU Product

Pickers - JSP

The term "pickers" is used within CRS-M to refer to the SKU selectors, i.e. the pickers that are seen on the product details page. In terms of page code, the term "pickers" refers to a JSP that can display those SKU selectors.

The picker parameter passed to productDetail.jsp (see above) is a string representing the relative path of the JSP that should be used to render the pickers for the given product. Although this path can point to any JSP on the site, there are two picker JSPs included with CRS-M: colorSizePickers.jsp and multiSkuPicker.jsp. It is worth noting that there is not a one-to-one correspondence from product details template JSP to picker JSP as two of the template pages—wood finish and color/size—can use the same picker, and single-SKU products do not require any pickers.

Similarly to the product details template JSPs, each picker JSP is also a template. They each derive the appropriate parameters to then pass to the generalized pickerList.jsp, which renders a generic picker. Currently, pickerList.jsp accepts the following parameters:

- collection: the collection of picker items to render
- valueProperty: the name of the item property that should be used as the picker 'value'
- type: a string used to identify this picker, which must correspond to exactly one of the values passed to 'pickerSelectFunction' in product.js (see "Pickers — JavaScript")
- callback: the JavaScript function to call when the picker value has changed
- id: the DOM id of the picker
- selectedValue: the currently selected value
- disabled: disables the picker
- defaultLabel: the default label to display when no value has been selected
- labelProperty: the name of the item property that should be use as the picker 'label'. If not specified, the valueProperty is used.

Again, you can extend this to add additional picker JSPs and render modified or additional pickers for your purposes.

Pickers – HTML and CSS

The generated markup for the pickers is intentionally simple. An individual picker is of the following form:

```
<label for="${id}" …>
  <select id="${id}" …>
    <!-- Default option -->
    <option value="" label="${defaultLabel}" disabled/>
    <option value="…" label="…"/>
    …
  </select>
</label>
```

It is worth noting that the select control is nested inside its own label. This was included primarily for accessibility considerations, as every form control must have an associated label.

CSS styles are applied to the select control to customize its appearance while utilizing the browser's native <select> element functionality. This greatly reduced the amount of work that would have been necessary to create a completely custom select control without sacrificing the desired functionality. The customized appearance of the select elements can be seen in Figures 9, 10 and 11.

Pickers – JavaScript

As noted above in the section entitled "Product Configuration Data", all functions necessary for picker functionality are created and bound when the page loads. These functions are members of the skuFunctions object, which follows the JavaScript "module" pattern. The signature of the skuFunctions "module" is as follows:

- cartLinkClick: called when the "Add To Cart" button is clicked. This function either updates the cart or adds a new item to the cart, depending on the current context.
- emailMe: called when a SKU is out of stock. This function displays a modal dialog that offers to notify the user when the item is back in stock.
- emailMeSubmit: called when the form in the aforementioned modal dialog is submitted. This function executes an AJAX request that submits the form and depending on the content that the user has entered, either flags invalid fields or shows a confirmation dialog.
- outOfStock: called when an out-of-stock SKU is selected. This function updates the displayed price, the text on the "Add To Cart" button, and the hidden SKU id field in the aforementioned email notification form.
- colorSizePickerSelect, multiSkuPickerSelect, woodFinishPickerSelect: these functions are called when a picker value has been updated. They all use the same module-local function, pickerSelectFunction, which updates the displayed price, checks the availability of the selected SKU, changes the text displayed on the "Add To Cart" button (if necessary), and sets the hidden SKU id field in the add-to-cart form. The only difference between these functions is a string that fixes which type of SKU to look for – color/size, multi-sku, or wood finish.
- quantitySelect: this function is called when the value of the quantity picker has been updated. It updates the quantity field of the add-to-cart form and, if necessary, changes the text on the "Add To Cart" button.
- available, preorder, backorder: these functions are used to generate appropriate function closures, which are then associated with a given SKU during the data-loading step described in the section "Product Configuration Data".

Again, these functions can be modified or additional functions added in order to meet your requirements.

Common Elements

There are several UI elements that are reused throughout the CRS-M. Here we will examine the most important of these.

Navigation and the Custom MobilePageContainer.tag

As in CRS, all pages in CRS-M are enclosed within a "page container". This custom tag automatically includes the content common to every page, including JavaScript and CSS imports, as well as common page elements such as the header and footer.
The tag accepts:

- A string parameter—**titleString**—that represents the title of the current page
- A page fragment parameter—**modalContent**—that renders content that may be display in a custom modal dialog
- A Boolean-valued parameter—**displayModal**—that flags the modal content to be visible immediately upon page load

Modal Content

The structure of each page that is imposed by the page container tag requires that modal content be specified separately from other page content to ensure full compatibility with the rest of the site. Specifically, CSS constraints require that in order for modal content to display correctly in front of the page content it must be separated from that content. This is demonstrated by the following page structure:

```
<body>
  <div id="mobile_store_pageContainer">
    <div id="mobile_store_header">
      <!-- header content -->
    </div>
    <div id="mobile_store_container">
      <!-- regular page content -->
    </div>
    <div id="mobile_store_footerWrapper">
      <!-- footer content -->
    </div>
    <div id="mobile_store_modalOverlay">
      <!-- modal content -->
    </div>
  </div>
</body>
```

Additionally, the JavaScript function **toggleModal** is provided to automate the process of displaying and hiding modal content. By default, when **toggleModal** is used to *show* the modal content, it only affects the visibility of the mobile_store_modalOverlay . Any modal content that needs to be displayed must have its visibility set separately in

order to be displayed when **toggleModal** is called. Conversely, when **toggleModal** is called to *hide* modal content it will also modify the visibility of all the immediate children of the mobile_store_modalOverlay , setting the CSS display property to "none", in order to reset the state of the modal display. Figure 13 shows an example of what is meant by "modal content", i.e. the Contact Us menu is modal in this case.

Header

The header consists of four components. From left to right, they are: the store logo, the search bar, the cart button, and the profile button. With the exception of the search bar, touching any of these components takes the user to the appropriate page (the homepage for the store logo, the user's shopping cart for the Cart button, the "My Account" area for the profile button).

Touching the Search bar causes it to expand, at which point the user may enter a search query. Once the search query is submitted, the user is taken to the Search Results page.

Additionally, when there are items in the Cart, a small badge will appear on the Cart button denoting the number of items in the cart. Figure 9 shows the appearance of the header when the user is typing a search term, and there are 3 items in the cart.

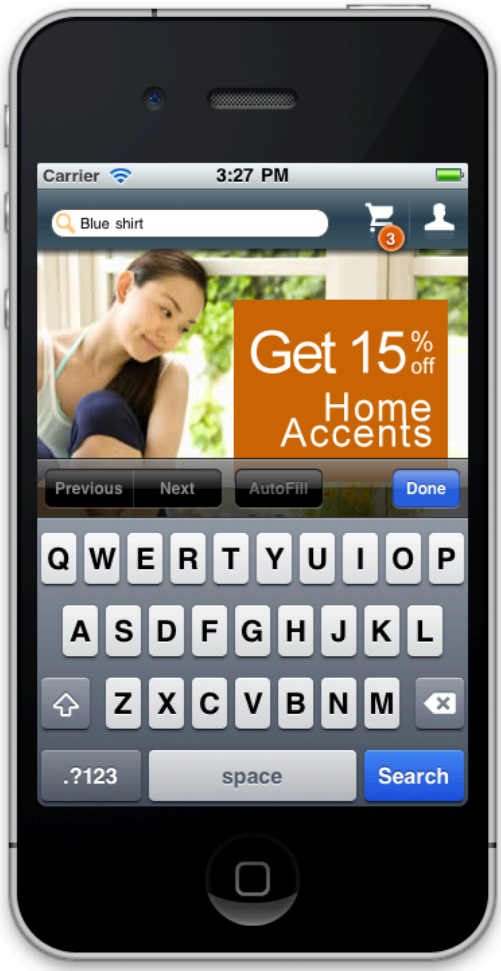

Figure 12 - Header appearance while typing Search Terms

Footer

The footer contains the site copyright as well as useful links for the user. Touching the "Contact Us" area displays phone and email options, both of which will handoff to the device's native functionality for making calls or composing email, as shown in Figure 10.

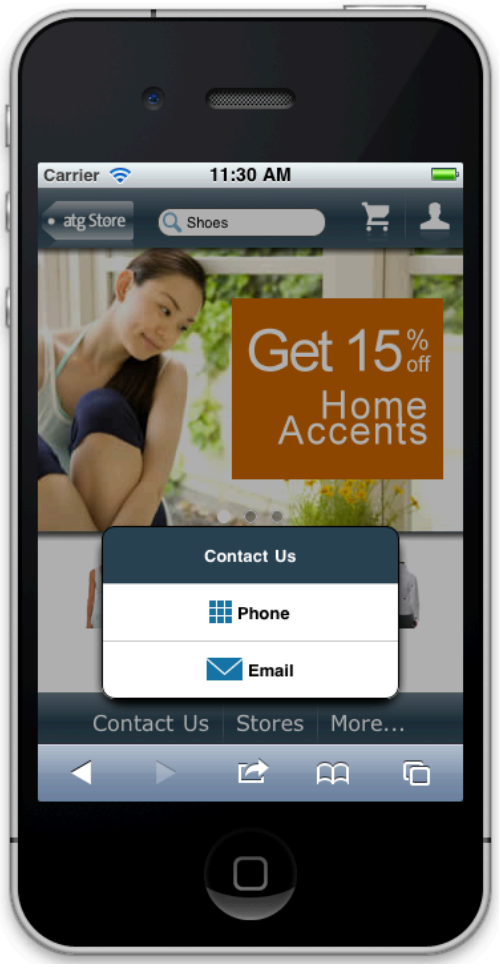

Figure 13 - Contact Us Popup

Touching the "Stores" area takes the user to the Stores page, which lists retail locations for the current site, as shown in Figure 11.

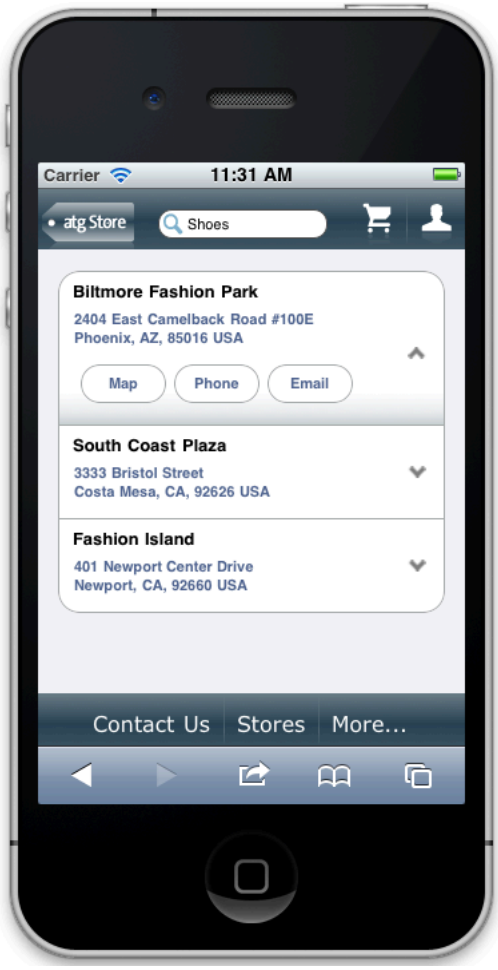

Figure 14 - Stores Page

Touching the "More" area takes the user to a page containing other useful options, including viewing the store's privacy policy and selecting a language, as shown in Figure 12.

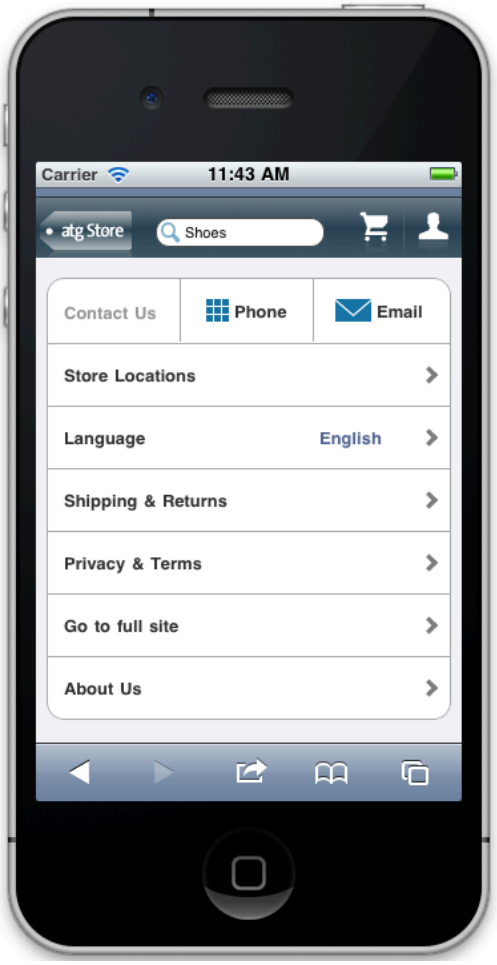

Figure 15 - "More" Area

User Experience (UX) Design

Principles

As with all other aspects of CRS-M, its UX design is intended as an example of how to approach the UX design of a mobile site. It can be customized or adapted to meet specific business needs.

Much thought and effort have been invested in the UX design of CRS-M. Some of the most important goals included:

- Providing a distinct mobile-specific UI
- Optimizing that UI for mobile browsers
- Ensuring that the UI's navigation is intuitive to users of touchscreen devices
- Enabling speedy checkout

In order to achieve these goals, many aspects of the shopping experience were revisited and many design principles were adapted to apply specifically to the mobile environment. During the UX design process the following traits emerged as desirable for mobile UIs:

- Simple
- Utilitarian
- User-centric
- Different from desktop

Simple

It was found that the task of simplifying a desktop UI for use on a mobile device was relatively easy up to a point, but realizing this goal to the point that user expectations were met required significant additional effort. In the mobile environment where space is limited, every task has to be sufficiently clear so that task-specific instructions are not necessary.

Internet Retailer's *2012 Mobile Commerce Top 300* report quotes successful mobile retailers in the field as stating that the keys to their success are simplicity and speed. One such retailer proposed that typically mobile device users are multitasking while using their mobile devices to shop.

To that end, several aspects of the CRS-M UI and underlying architecture are designed with speed in mind. These are discussed in each section as applicable and specifically in the section entitled "Performance Considerations".

Utilitarian

Also in Internet Retailer's *2012 Mobile Commerce Top 300* report, a successful mobile retailer stated that the act of making it fast and easy for customers to find what they want is essential in order to be successful. To this end, CRS-M's design is currently search-driven rather than browse-driven. Furthermore, the search process aims to reduce the number of server roundtrips in order to speed up this process.

The "refine" process allows the user to refine their search results, while the facets that are used to perform this refinement provide users with the opportunity to see how the catalog is organized. The search and refinement process is discussed in detail in the section entitled "Search" under "Low-Level Architecture".

Also a utilitarian trait, CRS-M aims to keep the number of form fields to a minimum in order to reduce the amount of manual input required by users. Required fields are not explicitly flagged, as all fields displayed are in fact required. This eliminates the need for the user to distinguish between required and optional fields.

It is worth noting that making a UI more utilitarian does not necessarily mean adding more powerful features, but rather may mean removing anything that is not directly useful to the user in order to complete the task at hand.

User-Centric

In the case of CRS-M, user centricity was interpreted as the need to make the process of purchasing an item using CRS-M as simple as possible. This is particularly evident in the checkout flow of the application, where checkout is significantly expedited for logged-in users. User centricity also drove the provision of clear decision points, more pre-populated fields and fewer input fields where possible.

Different to Desktop

Desktop and mobile UIs differ not only in their dimensions but also in the physical environments in which users interact with them. Desktop UIs are typically interacted with from a laptop or desktop computer, which by nature implies a stationary physical location. Mobile UIs by their nature can be interacted with in a variety of physical locations (as discussed in the "Simple" section).

Desktop UI paradigms do not always apply directly to touchscreen devices. Since CRS-M's UI is targeted at touchscreen devices, its UI takes advantage of this more discovery-centric environment, i.e. gestures are familiar to users of touchscreen devices and those users readily experiment with gestures. Therefore the CRS-M design relies more on swiping than it does on traditional buttons with instructions as the means to explore and interact with the UI. To this end, CRS-M's UI uses horizontal touch sliders and touch-responsive page areas as well as traditional vertically scrolling pages.

Examples

Homepage

The homepage was designed with 4 specific goals in mind:

- Making a good first impression
- Introducing the user to the primary mechanisms of the site
- Providing the ability to showcase several products and promotions
- Easy customization

CRS-M aims to make a good first impression by providing a visually and physically engaging homepage through the use of promotional content displayed in horizontally scrolling panels. This content is personalized for each user, which serves to further engage him/her.

The personalized content is provided by Oracle ATG Web Commerce content targeters, and if available, Oracle Recommendations On Demand. This allows easy customization through the BCC or the recommendations service. This is discussed in detail in the section entitled "Homepage" under "Low-Level Architecture".

Product Details

The product details page traditionally presents the highest number of competing priorities of any page in an e-commerce application. Minimizing this competition can be essential to the user's ability to make a purchasing decision about a given product. While the CRS-M approach to the product details page may not match your specific business model, it may serve as a starting point for this page's UX design.

The decision was made to feature the product photograph prominently and just large enough to help the user to decide if they are interested in this product at first sight. The user can choose to see (and wait for) a larger image by touching the photograph.

The product's property pickers (e.g. color, size, quantity, finish) expand only after the user touches them. This user-initiated expansion allows the product photograph to be larger and more dominant in the page's default/initial state, as it does not compete with the property pickers for the user's attention when he/she first arrives on the page.

Shopping Cart

The shopping cart page was designed to allow individual rows in the cart (representing products) to be expanded downwards when the user touches them. This provides more display space for your specific editing options, in addition to (or different from) those provided by CRS-M. The goal here is to provide an example design that can scale to different business needs so that retailers can offer their own cart editing mechanisms.

Checkout

Enabling speedy checkout was a major goal of the UX design for CRS-M. Throughout the checkout flow the number of questions posed to the user on any one screen is reduced by separating long forms with compound tasks into individual pages with focused tasks.

This allows the checkout flow experience to feel faster and more focused to the user. In some cases this separation of long forms into individual pages requires more page loads, but in all cases only the pages that are relevant to the user's current task are loaded. This modular approach to the various steps of the checkout flow allows retailers to more easily configure the checkout process to their requirements.

Internationalization

CRS-M currently supports two languages: Spanish and German. This was accomplished using the traditional means of externalizing all UI strings to resource bundles, which are shared with CRS, so that CRS-M re-uses existing resource strings, and so that translations are managed in a central location.

A language picker is provided in CRS-M's UI, and it leverages the built-in locale and language support of Oracle ATG Web Commerce.

Translations for the promotional content items that appear on CRS-M's homepage are not located in resource bundles, but rather in the Store.Storefront/data/catalog-i18n.xml file, leveraging the same mechanism as CRS for this purpose. For more information on this see the "Internationalization" chapter in the "ATG Commerce Reference Store Overview" available from the Oracle ATG Web Commerce documentation page.

Performance Considerations

Testing the performance of a mobile web application can be a challenge for many reasons, one of which is the difficulty in setting up a realistic test environment. End users of mobile e-commerce web applications will access those applications, most likely, over 3G networks, meaning that any test done over WiFi will not be accurate.

However, there are some commonly agreed upon techniques that can be employed to improve the performance of a mobile web application. At the time of writing, the W3C provides a recommendation on Mobile Web Best Practices which is available from http://www.w3.org/TR/mwabp/.

Use Appropriate Client-Side Storage Technologies for Local Data

As discussed in the Low Level Architecture – Homepage section, CRS-M uses HTML 5 session storage to cache the homepage data on the client. Also discussed in that section is the question of whether a particular situation would benefit from more aggressive caching. This may be worth considering for each particular implementation.

A separate note, also mentioned in the homepage section, is that CRS-M's homepage currently does not have a mechanism to handle the case where HTML 5 session storage is **not** available. Such a mechanism would be preferable for mobile websites targeting a wider range of devices and browsers.

Enable Automatic Sign-in

CRS-M uses the same cookies as CRS, and hence the same behavior with regard to automatic sign-in applies.

Minimize Application and Data Size

CRS-M adds a target to its build file that combines its CSS and JavaScript files into one CSS file (mobile.css) and one JavaScript file (mobile.js) respectively, then compresses ("minifies") these combined files. The "minified" files are imported only once in the pageStart.jsp file. This build file is not distributed with CRS-M in this release.

Minimize Perceived Latency

When loading search refinements, a WebKit animated loading icon ("spinner") is displayed to the user.

Use Meta Viewport Element To Identify Desired Screen Size

CRS-M configures the viewport in pageStart.jsp as follows:
<meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0; user-scalable=0;" />

Prefer Server-Side Detection Where Possible

See the section entitled "Mobile Interceptor" under "General Configuration" for a description of server-side user-agent detection.

Customizations

In the preceding sections we have touched on many areas that can be customized to your needs.

In the "High Level Architecture" section we discussed the Mobile.DedicatedSite and Mobile.UIOnly options. Having chosen the option that best suits your needs, you could then customize other aspects of CRS-M. These include:

- Access control for your pages – allowing fewer or more pages to be accessed before the user is required to log in
- Mobile user-agent interception – choosing which mobile browsers to forward to your mobile-specific pages, and also the directory-mirroring strategy
- Mobile billing- and profile- form handlers – these provide an example of how you can customize a flow or part of a flow to meet your specific needs

In the "Low Level Architecture" section we discussed in deeper detail some topics that were previously introduced. The most relevant of these with respect to customization are:

- Homepage personalization – how to customize the targeted content to your needs, and how to reuse or extend the JavaScript slider
- Search configuration with facets – how to take a slightly different approach to facet selection to improve user experience on a mobile device
- Product details – how to engineer a product details page that can accommodate many different types of SKU, on a page that dynamically updates as the user makes selections

In the "Performance Considerations" section we discussed some topics for improving performance that can be customized for your implementation:

- How to configure the viewport, and different ways that this can be done – i.e. allow the pinch-zoom gesture, disable pinch-zoom, configure for different screen sizes
- Caching of data on the client – the CRS-M homepage demonstrates one approach but this could be customized to be more or less aggressive based on your needs

An additional item of note that is useful for customization is what is referred to as site-specific CSS. CRS demonstrates how to use site-specific CSS to give different sites a different look-and-feel on its different site configurations. While this is not implemented by CRS-M, it certainly could be by following CRS' lead in this regard. For information about site-specific CSS in CRS please refer to the "ATG Commerce Reference Store Overview", which can be found via the Oracle ATG Web Commerce documentation page.

Feature Comparison with CRS

CRS-M does not currently have full feature parity with CRS. Features that are present in CRS but **not** present in CRS-M include:

- Support for Gift Lists
- Support for Wish Lists
- Support for Gift Wrap
- Support for shipping to multiple addresses
- Support for splitting a payment across store credit and a credit card
- Click to call integration
- Traditional catalog browsing/drill-down
- Display of recommendations on pages other than the homepage (e.g. product details page, cart page)
- Sorting of products (neither search results nor category landing page sorting)
- Support for recently viewed products
- Choosing of the gift wrap and/or gift note options

- Viewing orders having:
    - Payment split across store credit and a credit card
    - Multiple shipping addresses
    - Gift wrap items
        - (In each of these cases the user will be given the option to view the order on the "full site" meaning the desktop version of CRS)

- Product Comparisons
- Viewing all promotions on a single page
- Searching across sites (i.e. the "include [site x] products" checkbox for search)
- Express Checkout

Conclusion

As online retailers strive to provide the best possible experience to their mobile customers, they increasingly look to Oracle ATG for guidance. The Mobile Commerce Reference Store Web version (CRS-M) aims to decrease time to market for online retailers on the Oracle ATG Web Commerce platform, by providing a mobile reference application designed and developed with current best practices in mind. Configuration and customization options mean that CRS-M can be used as the basis of a mobile e-commerce site, significantly reducing the cost and time required to launch or re-launch a site. It can also be used as an example of how an existing Oracle ATG Web Commerce offering can be extended to provide a mobile offering. You can start today by going to http://edelivery.oracle.com to download version 10.0.3.1 or higher of the Oracle ATG Commerce Reference Store which will contain CRS-M as an extension of CRS. This document can be used for guidance as you navigate the source code to find specific areas of interest to you, or to find customization and configuration points that can help to make this application your successful mobile offering.

A Guide to CRS-M
December 2011
Author: Edel Hartrey-Choi

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together