KonaKart
Enterprise Java eCommerce

KonaKart JavaScript Tiles

5th January 2015

DS Data Systems (UK) Ltd.,
9 Little Meadow
Loughton, Milton Keynes
Bucks
MK5 8EH
UK

Introduction

What are tiles?

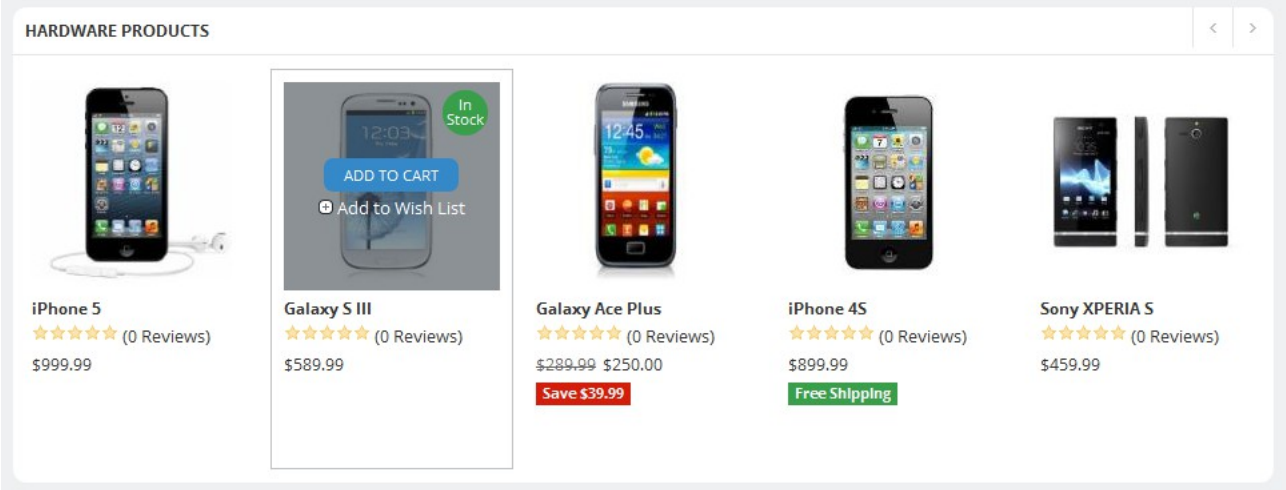
KonaKart tiles can be regarded as building blocks for creating an eCommerce application which can be easily integrated into a front end system such as a CMS (Content Management System) or portal. Each tile (or aggregation of similar tiles) has a template, a style sheet and a JavaScript file which control the look and feel as well as the functionality. The tiles communicate with a KonaKart server using the KonaKart JSON APIs called asynchronously from the JavaScript using AJAX.

What's the purpose of tiles?

KonaKart has been designed to provide loose coupling between the storefront application and the eCommerce server. The eCommerce server is a service (which may also be cloud based) that provides all eCommerce functionality through JSON APIs. This architecture lends itself well for the integration of the functionality into existing front end systems. A typical approach for displaying products and adding a shopping cart into a CMS is to call the KonaKart APIs from the CMS and to manually integrate the data received from KonaKart. KonaKart tiles greatly simplify this process because they provide an integration point at a higher level than the pure APIs. They provide functional widgets that already have a template based UI design and that autonomously capture events and communicate with the KonaKart server.

How can tiles be used?

In the case of a CMS, the tiles may be placed anywhere on a page in order to provide seamless integration with the other content on the page. For example, in one page you may wish to provide a carousel of scrollable products that match the content described on the page.

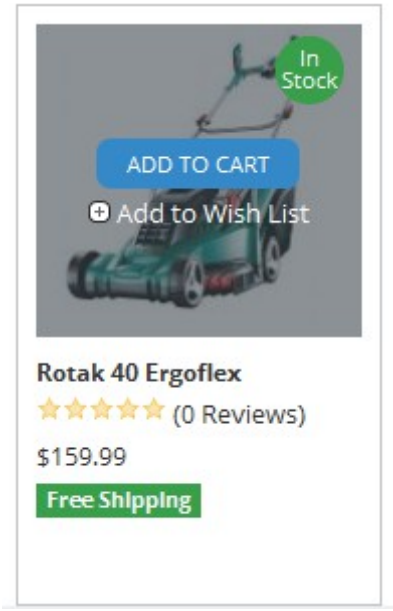


You may decide to include an “Add To Cart” button for each carousel product. In this case, when the button is clicked it will automatically communicate with the KonaKart server to add the product to the cart and also if the popup cart tile is included in the page, it will receive the notification and update itself with the new contents.

Architecture

The tiles are designed so that they can be used independently allowing you to integrate only the eCommerce functionality that you require anywhere on the page. Each tile may be positioned separately.

As an example let's take the product tile which is a tile that displays a product summary as shown below:



It has an underscore template (http://underscorejs.org/) that allows you to easily modify the layout of the tile without touching any of the JavaScript code. It also has a CSS file containing the styles and a JavaScript file that controls the functionality, so that whenever a button is clicked, an action is performed and if present, other tiles are invoked.

In this particular case the CSS and JavaScript files are the same ones that are also used for the product details tile. In some cases, more than one tiles will share the same CSS and JavaScript files although each tile always has it's own template.

Installation

KonaKart tiles are automatically installed during the installation of the Enterprise Extensions. A konakart_tiles webapp is installed under the webapps directory. The webapp contains the following directories:

html directory

This directory contains a list of HTML files which provide examples of how the individual tiles may be used and positioned on a page. Let's continue using the product tile as an example.

```
<body>
    <script>
        $(function() {
            // Component configuration values
            var prodId = 4;
            var selector = $("#prod-tile");
            var addToBasketEnabled = true;
            var wishListEnabled = true;
            // End of component configuration values

            var kkProdTileConfig = new Object();
            kkProdTileConfig.style = "kkpt-single";
            kkProdTileConfig.selector = selector;
            kkProdTileConfig.addToBasketEnabled = addToBasketEnabled;
            kkProdTileConfig.wishListEnabled = wishListEnabled;
            kkProdTileConfig.addHandlers = true;

            kkEng.getProduct(kk.getSessionId(), prodId, kk.getLangId(),
                    function(result, textStatus, jqXHR) {
                        var kkProd = decodeJson(result);
                        kk.renderProdTile(kkProd, kkProdTileConfig);
                    }, null, kk.getKKEng());
        });
    </script>
    <div id="prod-tile"></div>
</body>
```

The script contained in the html file may be seen above. It defines the id of the product that needs to be displayed in the tile and contains a div (* *) where to position the tile. Also, the tile may be configured to decide whether the “add to cart” and “add to wishlist” links should be enabled. The KonaKart engine is called to get the product and then the tile is rendered by calling *kk.renderProdTile(kkProd, kkProdTileConfig).*

html/template directory

This directory contains the tile underscore templates. More information regarding the templates may be found at the following link http://underscorejs.org/#template . Each tile has a template which is used to render the tile. The template is compiled and at render time, it is passed data (e.g. product data or cart data) which it uses to populate the generated HTML. These templates may be modified in order to customize the generated look and feel of each tile.

styles directory

This directory contains a number of CSS files that provide style information for the tiles. If the template of a

tile is modified to include a new style, then this new style should be included in the appropriate CSS file. Some of the CSS files provide styles for more than one tile.

script directory

The script directory contains the JavaScript files that control the individual tiles. Each file may control one or more tiles. Amongst other things, the JavaScript is responsible for creating and populating the tiles with data from the KonaKart engine and also for managing events such as button clicks once the tiles have been rendered.

gensrc directory

All content within the gensrc directory is dynamically created by running the ant build file (build.xml) directly under the konakart_tiles directory. It isn't populated directly after an Enterprise Installation. This directory contains script, html and styles sub directories although in this case all of the CSS and JavaScript files have been compressed and minimized and the templates have been added to the minimized JavaScript file. The files under gensrc are typically what you would use in a production environment.

Eclipse Project

The webapp includes a .classpath file and .project file so that it can easily be imported into Eclipse for development.

Enable JSON

In order for the tiles to send and receive information from the KonaKart engine, you must ensure that the JSON APIs have been enabled. The process for enabling the APIs is explained in detail in the standard User Guide so here we will just give brief instructions.

A convenient way to enable JSON services is to run the *enable_JSON* ANT task provided in the build.xml file in the custom directory of the standard installation as follows:

```
C:\Program Files\KonaKart\custom>bin\kkant enable_JSON
Buildfile: build.xml
enable_JSON:
enable_JSON_warning:
enable_JSON_enterprise:
[echo] Fix konakart web.xml to start-up JSON
BUILD SUCCESSFUL
Total time: 0 seconds
```

Instructions for modifying the web.xml file manually are in the User Guide.

Try a tile

The most complete tile to try out is store.html which assembles all of the tiles into a storefront application. The URL for this tile after a standard install on port 8780, is http://localhost:8780/konakart_tiles/html/store.html . The same tile using the minified JavaScript and CSS can be found at http://localhost:8780/konakart_tiles/gensrc/html/store.html after performing a build (explained later in this document).

Configuration

The JavaScript file used to configure the tiles is called kk-configure.js. The configuration variables are annotated and hopefully easy to understand.

```
/*
 * The root URL of the KonaKart server engine.
 * The value can be changed depending on where KonaKart is running.
 */
var kkRoot = 'http://localhost:8780/konakart/';
//var kkRoot = 'http://www.konakart.com/konakart/';

/*
 * Default language id (id of language record in KK database)
 */
var KK_DEFAULT_LANG_ID = 1;

/*
 * Default language locale
 */
var KK_DEFAULT_LOCALE = "en_GB";

/*
 * Default ISO currency code
 */
var KK_DEFAULT_CURRENCY_CODE = "USD";

/*
 * Default store country (3 digit ISO code)
 */
var KK_DEFAULT_COUNTRY_CODE = "USA";
```

As can be seen above, you can configure the location of the KonaKart engine where the JSON requests are sent and other parameters such as the default locale and default currency.

If you set kkRoot to http://www.konakart.com/konakart/ it will use the JSON APIs of the demo application running on the KonaKart server.

By default, the configuration variable *KKSolrEnabled* is set to null so the value is looked up from the KonaKart database. If you know that Solr will (recommended) or will not be enabled then it is more efficient to set the value of this variable to *true* or *false*. The current implementation of the facets tile only shows manufacturer and other facets when Solr is enabled.

Development

As mentioned previously the konakart_tiles webapp includes a .classpath file and .project file so that it can easily be imported into Eclipse for development.

Most of the JavaScript files control one or more tiles. At the top of each file there is a short description informing you which tiles it controls. e.g. for kk-prodTile.js the comment is:

```
/**
 * JavaScript for the product and product details tiles.
 * <ul>
 * <li>product tile using the template prodTile.html</li>
 * <li>product details tile using the template prodDetailTile.htm</li>
 * </ul>
 */
```

Some exceptions are:

kk-tile.js

This contains code that is common to all tiles. The utility methods include methods to get a template, to manage the session, to retrieve messages from the message catalog and to create a common template context which is the data passed to all templates when they generate the tile HTML.

kk-formTiles.js

This file contains mainly form validation code as well as some other utility methods related to forms. It is used by the tiles that contain forms.

kk-configure.js

This file contains configuration variables that may be modified to configure your system.

kk-store.js

This file is only used by store.html which is a test HTML file that uses all of the tiles to create a storefront. It contains code that starts the backbone router (http://documentcloud.github.io/backbone/#Router) and that sets up the front page.

I18N

Messages

All messages are retrieved from a JavaScript message catalog under script/i18n. The name of a message catalog is in the format kk-{locale}.js so for example could be kk-en_GB.js or kk-es_ES.js . The message catalog also defines the format for dates and the decimal point and thousand separators for numbers. The messages are grouped by tile.

```
kkMsgMap["en_GB"] = {
    // Common messages for all tiles
    "exception.render.tile" : "%{0} must be called before calling %{1}",
    // Date format for displaying dates
    "datepicker.date.format" : "dd/mm/yy",
    "dateformat.date.format" : "dd/mm/yyyy",
```

A typical message catalog uses the format shown above. The locale of the catalog must be entered in the first line which could for example be kkMsgMap[“es_ES”] for Spanish.

The default locale is set in kk-configure.js and there must be a matching message catalog for this locale.

KonaKart uses the Polyglot JavaScript library (http://airbnb.github.io/polyglot.js/) to manage the message catalogs.

In order to dynamically change locale there is a method in kk-tile.js:

```
/**
 * Called to change the locale. The locale passed in must be in the format
 * en_GB, es_ES etc. The language Id isn't mandatory but if it is passed then we
 * don't have to do a database lookup.
 */
kk.changeLocale = function(callback, locale, langId) {
```

Currencies

The currencies are defined in a JavaScript file called kk-currencies.js under script/i18n.

```
kkCurrencyMap["GBP"] = {
    "symbol" : "£",
    "format" : "%s%v", // output: %s=symbol, %v=value/number
    "precision" : "2"
};
kkCurrencyMap["EUR"] = {
    "symbol" : "€",
    "format" : "%s%v", // output: %s=symbol, %v=value/number
    "precision" : "2"
};
kkCurrencyMap["USD"] = {
    "symbol" : "$",
    "format" : "%s%v", // output: %s=symbol, %v=value/number
    "precision" : "2"
};
```

The default currency code is defined in kk-configure.js and a currency mtching the default currency must exist in kk-currencies.js. If your default currency is not GBP, EUR or USD, you may add a new entry. KonaKart uses the accounting JavaScript library (http://openexchangerates.github.io/accounting.js/) to format currencies.

In order to dynamically change currency you must set the variable *kkCurrencyCode* to the currency code of the new currency.

KonaKart Engine APIs

The JavaScript calls to the KonaKart engine are identical to the standard KKEngIf API calls. The Javadoc for these API calls may be found at http://www.konakart.com/javadoc/server/com/konakart/appif/KKEngIf.html . When called from JavaScript they are asynchronous and use a callback function to return results.

Production

In development mode the tiles include many JavaScript files and many CSS files. Also the templates are read as files from disk when a tile is rendered.

Once you have finished customizing and developing your own tiles you can run an Ant task that performs the following:

1. Creates a gensrc directory under konakart_tiles with all files required for production
2. Creates a single minimized JavaScript file for all of the tile JavaScript files. All of the templates and message catalogs are included in this single file called kk-tile-gen.min.js.
3. Creates a single minimized CSS file for all of the tile CSS files. It is called kk-tile-gen.min.css.
4. When copied over to the gensrc area, all of the tile HTML files are modified to remove the old includes and include the new minimized includes.

```
C:\Program Files (x86)\KonaKart\webapps\konakart_tiles>..\..\custom\bin\kkant
Buildfile: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\build.xml

clean:
     [echo] Cleanup...

createGensrc:
     [echo] Creating directories for generated source...
    [mkdir] Created dir: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc
    [mkdir] Created dir: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\script
    [mkdir] Created dir: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\html
    [mkdir] Created dir: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\styles
    [mkdir] Created dir: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\images
    [mkdir] Created dir: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\bin

compileBuildUtils:
     [echo] Compiling TileBuildUtils
    [javac] Compiling 1 source file to C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\bin

minimizeJS:
     [echo] Creating JavaScript file from templates
     [echo] Creating single JavaScript file
     [echo] Minimizing JavaScript

minimizeCSS:
     [echo] Creating single CSS file
     [echo] Minimizing CSS

processHTML:
     [echo] Processing HTML files

copyJS:
     [echo] Copying JS files
     [copy] Copying 8 files to C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\script

copyCSS:
     [echo] Copying CSS files
     [copy] Copying 34 files to C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\styles

copyImages:
     [echo] Copying image files
     [copy] Copying 31 files to C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\images

tidyUp:
     [echo] Tidy Up
   [delete] Deleting: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\script\kk-template-gen.js
   [delete] Deleting: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\script\kk-tile-gen.js
   [delete] Deleting: C:\Program Files (x86)\KonaKart\webapps\konakart_tiles\gensrc\styles\kk-tile-gen.css

build:

BUILD SUCCESSFUL
Total time: 5 seconds

C:\Program Files (x86)\KonaKart\webapps\konakart_tiles>
```

Part of the build is performed by a Java class called TileBuildUtils.java found under konakart_tiles/src/com/dsdata/util . It has several methods called by Ant during the build. The JavaScript and CSS files that are added to the minimized files are defined in Static variables within TileBuildUtils.java as shown below:

```
/*
 * Update to add all javaScript files that should be added to the minimized KonaKart javaScript
 * file. Don't remove ../gensrc/script/kk-template-gen.js since this is a generated file
 * containing all of the templates.
 */
private static final String[] jsFileArray = new String[]
{ "../gensrc/script/kk-template-gen.js", "kk-dateFormat.js", "kk-configure.js", "kk-tile.js",
        "kk-prodTile.js", "kk-productsTile.js", "kk-cartTile.js", "kk-wishListTile.js",
        "kk-carouselTile.js", "kk-verticalCarouselTile.js", "kk-reviewsTile.js",
        "kk-formTiles.js", "kk-loginTile.js", "kk-registerTile.js", "kk-accountTile.js",
        "kk-customerInfoTiles.js", "kk-onePageCheckoutTile.js", "kk-orderDetailTile.js",
        "kk-ordersTile.js", "i18n/kk-en_GB.js", "i18n/kk-currencies.js", "kk-router.js",
        "kk-store.js" };

/*
 * Update to add all css files that should be added to the minimized KonaKart css file
 */
private static final String[] cssFileArray = new String[]
{ "kk-tile.css", "kk-prodTile.css", "kk-reviewsTile.css", "kk-productsTile.css",
        "kk-facetsTile.css", "kk-catMenuTile.css", "kk-searchTile.css", "kk-cartTile.css",
        "kk-wishListTile.css", "kk-carouselTile.css", "kk-verticalCarouselTile.css",
        "kk-formTiles.css", "kk-loginTile.css", "kk-registerTile.css", "kk-accountTile.css",
        "kk-customerInfoTiles.css", "kk-onePageCheckoutTile.css", "kk-orderDetailTile.css",
        "kk-ordersTile.css", "kk-breadcrumbsTile.css" };
```

You may modify the included files depending on how many tiles you are using and whether you have created any of your own tiles or added new message catalogs etc.

store.html

This a tile that aggregates all tiles into a storefront application. As can be seen below, the html defines all of the tile placeholder and calls kk.storeInit() which is in kk-store.js.

```
<body>
    <script>
    $(function() {
            kk.storeInit();
    });
    </script>
    <div id="kk-page-container">
        <div id="kk-page">
            <div id="kk-top-bar-container">
                <div id="kk-top-bar">
                    <div id="kk-options-container">
                        <div id="kk-options">
                            <div id="kk-popup-cart-tile"></div>
                            <div id="kk-popup-wishList-tile"></div>
                            <a id="kk-account-link"></a>
                            <a id="kk-logout-link"></a>
                        </div>
                    </div>
                </div>
            </div>
            <div id="search-tile"></div>
            <div id="main-menu"></div>
            <div id="breadcrumbs-tile"></div>
            <div id="kk-products-container">
                <div id="facets-tile"></div>
                <div id="kk-content" class="narrow">
                    <div id="products-tile"></div>
                </div>
            </div>
            <div id="body-tile"></div>
            <div id="carousel-tile"></div>

        </div>
    </div>
    <div class="kk-modal"></div>
</body>
```

The initialization code starts the backbone router (http://documentcloud.github.io/backbone/#Router) as well as rendering the front page and setting the position for each tile.

The backbone router allows you to use the forward and back browser buttons as well as to get to any tile using a URL. It isn't enabled by default since it may not be required if the tiles are integrated within a CMS.