KonaKart jQuery API

28th May 2012

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

Introduction

KonaKart (www.konakart.com) is a Java based eCommerce platform where all functionality is available through APIs. This makes it particularly suitable for integration into front and back end systems. In this document we'll concentrate on integrating store-front eCommerce functionality into any existing or new web based application.

KonaKart has already been integrated with a variety of Content Management Systems (CMS) and portals. Up until now the integration has been achieved either through SOAP APIs or by deploying the KonaKart store-front as a JSR-168 portlet. The jQuery plugin allows the integration to be achieved using lightweight JSON AJAX calls directly from JavaScript. This means that you can now easily integrate KonaKart eCommerce functionality into any browser based application

Note that the jQuery plugin is only available in the Enterprise Version of KonaKart.

The plugin and its dependencies

A standard KonaKart Enterprise installation creates a jquery sub-directory under the konakart webapp directory (i.e. …/KonaKart/webapps/konakart/jquery). The jquery directory contains all required JavaScript files as well a demonstration application.

The KonaKart jQuery plugin consists of a JavaScript file called jquery.konakart-x.x.x.x.js where x.x.x.x is the version of KonaKart that it is compatible with. You also need to include jQuery and the JSON jQuery plugin.

For example:

```
<script type="text/javascript" src="jquery.konakart-6.3.0.0.min.js"></script>
<script type="text/javascript" src="jquery-1.7.2.min.js"></script>
<script type="text/javascript" src="jquery.json-2.3.min.js"></script>
```

After installation you'll note that all of the JavaScript files can be found under the jquery/js directory.

The JavaScript APIs are identical to the standard KonaKart eCommerce engine APIs. The Javadoc for these APIs is within every KonaKart installation kit. It is also available online at http://www.konakart.com/javadoc/server/ . The API calls are the methods of the class KKEngIf. The only difference is that for the JavaScript APIs there are three extra parameters (see below) for each API call, which are added after the standard parameters of the call.

How to Enable JSON for the KonaKart Engine

By default the JSON Server servlet sections in the web.xml file for the konakart webapp are commented out so JSON services are disabled after a standard install. Full instructions for enabling JSON can be found in the KonaKart User Guide.

How to use the plugin

It's relatively easy to make a call to the KonaKart eCommerce engine. These are the steps to take:

Create an engine object which will be passed to all API calls. Note that the store id only needs to be set when working in multi-store mode.

```
// Define the url and store id of a running KK engine
var conf = new engineConfig('http://localhost:8780/konakart/konakartjson');
conf.storeId = "store1";
conf.protocol = "json";
var eng = new kkEng(conf);
```

You can configure the protocol to be JSON or JSONP. JSONP or "JSON with padding" is a complement to the base JSON data format. It provides a method to request data from a server in a different domain, something prohibited by typical web browsers because of the Same Origin Policy where a web page served from server1.example.com cannot normally connect to or communicate with a server other than

server1.example.com.

When using JSONP, the browser does a GET rather than a POST. Some of the API calls send a large amount of data so you may have to modify the configuration of your web server to increase the maximum length of the HTTP GET request.

Once you have an engine object you can make an API call.

```
var getCategoryTreeCallback = function(result, textStatus, jqXHR) {
          var categories = decodeJson(result);
};

// Get the category tree from the KK engine
kkEng.getCategoryTree(/*langId*/-1, /*getNumProds*/true, getCategoryTreeCallback, null,
                                                    eng);
```

You define a Callback for every API call. Every API call includes the standard parameters for that call followed by these three parameters:

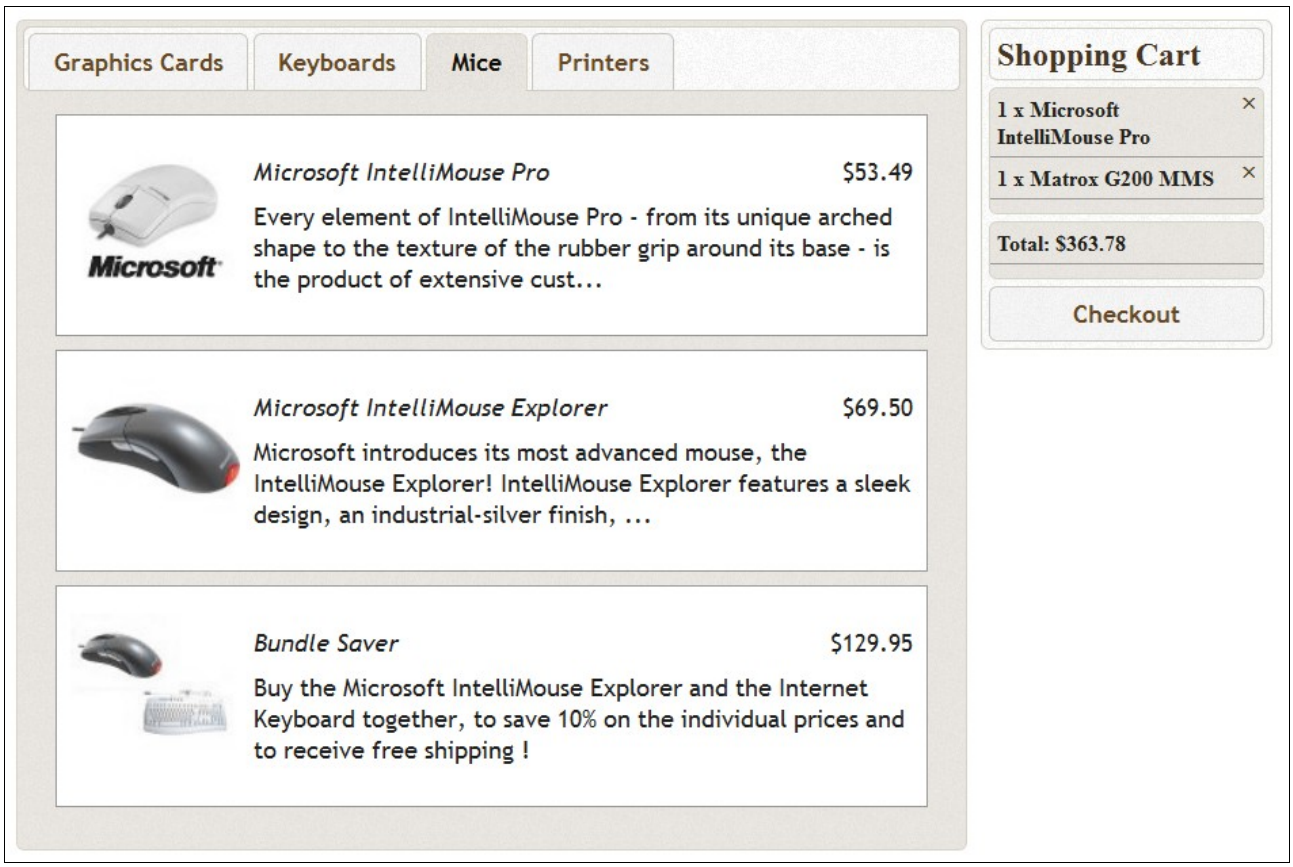
1. Callback – In the callback you can get the result of the API call. The result is a JSON string, so in order to be able to use it efficiently, you call the method decodeJson() which returns an object.
2. Context – Null in the above example. You may pass a context object which can be addressed in the callback as “this”. This is useful for making data available to the callback without having to make it global. e.g. When retrieving a list of products belonging to a category, you may want the callback function to have a copy of the Category object.
3. Engine object – The engine object contains information regarding the location of the KonaKart eCommerce engine so that we know where to send the JSON request. It also contains the store id of the request to be used in a multi-store scenario.

Example Demo

A commented example application is supplied in order to demonstrate the use of some of the API calls. This application consists of a single HTML file called kkCart.html and can be found under the /KonaKart/webapps/konakart/jquery/html directory. In order to run it after a standard install (and after having enabled the JSON APIs on the engine), you just have to enter this URL in a browser: http://localhost:8780/konakart/jquery/html/kkCart.html .

In order to work correctly there must be a running KonaKart system which has been installed with the demo database. After a default installation, the demo is configured to automatically use the local KonaKart engine. Category and product information is retrieved through API calls and used to create tab folders and populate them. In order to add a product to the cart, you must drag it onto the Shopping Cart area.

The checkout functionality hasn't been implemented in this demo although the functionality is present because it forwards you to the checkout procedure of the standard KonaKart store-front application.



Depending on where KonaKart has been installed, you may change the root directory as shown below.

For the default local installation:

```
var kkRoot = 'http://localhost:8780/konakart/';
```

The sequence of the API calls in the demo is as follows:

1. A temporary customer id (negative number) is fetched from the KK engine. This id is used to identify the customer even though he hasn't logged in.
2. The category tree is fetched from the KK engine and 4 hardware categories are used to create 4 tab folders.
3. Each tab folder is populated with products which again are retrieved from the KK engine.
4. Any product may be added to the cart by dragging it onto the cart tile. When this occurs, the code calls the KK engine to add a product to the cart.
5. Once the product has been added to the cart, an API call is made to retrieve all of the cart items for the temporary customer which are then displayed in the cart tile.
6. If a product is removed from the cart, the KK engine is notified through an API call and a new updated list of cart items is retrieved from the KK engine and displayed.
7. When the checkout button is clicked, the temporary customer id is saved by the engine in an SSO token and a UUID is returned to identify the token.
8. The current window location is changed in order to redirect the customer to the standard KK store-front demo application. The UUID is passed to the struts action so that code within the store-front app can look up the SSO token containing the temporary customer id. Using this id, the store front app can display the cart items and start the checkout process.