Job search application

Document history:

Revision	Date	Author	Comments
o.1	12-Apr-2012	Mark Singeisen	First draft
0.2	13-Apr-2012	Jean-Marc Imbert	Remarks
1.0	16-Apr-2012	Mark Singeisen	Final version
2.1	09-Jul-2014	Marcello Angileri	Added usDisclaimer functionality

Table of Contents

1 Introduction

This document describes the process of how to implement and customize a Novartis standard job search application on Novartis websites (internal & external).

2 Objectives

The objective of this document is to provide Technical owners (TO) with all information needed to:

- Integrate the job search application on their website
- Understand what Information can be searched for
- Send appropriate data to the API
- Retrieve and process data returned by the API
- Customize the search interface
- Adjust the layout & design
- Add custom functionality

3 Prerequisite

- Basic understanding of Javascript and the jQuery library
- Basic knowledge of HTML & CSS
- Website that allows embedding of Javascript

4 Preparation

4.1 Setup

The Novartis job search application is written in Javascript. It is an extension/plugin of the jQuery library which comes shipped with the Novartis standard templates. The job search application is using AJAX request to interact with an external API in order to return job results.

In order to ensure your application can be updated with new releases, please make sure to never make changes to the jQuery plugin itself. Content managers (CM) should use the configuration objects offered by the extension istead to ensure that the core of the application remains the same.

Because of the chosen technology, the job search application runs mainly on the client side. This means that users must have Javascript enabled in order to use the application. Using Javascript also means that there are only minor impacts on your systems (server side technologies).

For most of the browsers cross-site-scripting is disabled for security reasons. Novartis standard browsers (Internet Explorer) allow cross-site-scripting which should help you during the implementation and testing phase.

Please note: Before you publish the application, you want to make sure that all users (especially external people) are able to interact with your application, therefore some settings needs to be adjusted on your webserver (see chapter 4.3 configure Apache).

4.2 Set the scope (get your own Site ID)

The job search application per default is set up to query all open positions posted on the global external gateway.

If you decide to query jobs from other sources like:

- Gateway type other than global external (e.g., internal, referral)
- Gateway language other than global English international (e.g., German, Spanish)

Please get in touch with helpdesk.novartis@kenexa.com to get a Site ID that you can pass to the job search application.

4.3 Configure webserver for cross-site-scripting (Apache)

Per default most Browsers don't allow cross-site-scripting. The job search application uses this method to interact with the search API (3rd party service).

In order to ensure all your users can make use of your application the following settings must be placed in the configuration file of your webserver: Please get in touch with your local IT support for further information on implementation and testing.

```
<IfModule mod_proxy.c>
        ProxyRequests On
        ProxyVia On

        ProxyPass /brassring http://66.77.22.117/WebRouter/
    ProxyPassReverse /brassring http://66.77.22.117/WebRouter/
```

```
        </IfModule>
```

4.4 Download source files

In order to set up a job search application on your website you need to download the following source files from the Novartis Brand Service website:

- jobsearch-v.2.0.zip

The ZIP file contains the following information:

• **/jobsearch/css/jobsearch.css**	**(required)**
• /jobsearch/flash/worldmap.swf	(optional)
• **/jobsearch/images/icon-close.gif**	**(required)**
• **/jobsearch/images/loading.gif**	**(required)**
• index.html	(optional)
• **/jobsearch/js/jquery-1.4.2.min.js**	**(required if not already available)**
• **/jobsearch/js/jquery.jobsearch.js**	**(required – DO NOT CHANGE THIS FILE)**
• /jobsearch/js/swfobject.js	(only required if worldmap is used)

5 Job search application

5.1 Data values

5.1.1 API search fields

API search fields are values that can be searched for (e.g., country for all positions in Spain). In addition those fields can also be used to fill UI elements so that the user can interact with those to search for jobs (e.g., drop down menu with all countries).
The following API fields are available:

- country (default)
- division (default)
- function (default)
- therapeuticarea (default)
- region
- location
- business
- jobtype
- employmenttype
- jobid
- keyword

The fields highlighted as default will be available as a data object so they can be used to fill UI elements. Other data object can be created manually if necessary (see chapter 7.1.4 on custom data).

5.1.2 Job results placeholders

TOs can chose from the following set of information to show in the search results:

- URL to the job details ($LINK$)

- Unique ID of the position ($JOBID$)
- Position title ($TITLE$)
- Country where the position would be based in ($COUNTRY$)
- Location/City where the position would be based in ($LOCATION$)
- Division where the position would be in ($DIVISION$)
- Employment type of position ($EMPLOYMENTTYPE$)
- Job type of position ($JOBTYPE$)
- Therapeutic area associated with position if available ($THERAPEUTICAREA$)
- Global job function position belongs to ($FUNCTION$)
- Business Unit the position would be in ($BUSINESS$)

The results returned by the job search application can be customized to show different information than what is shown per default (see chapter 7.2.3.2 customize search results).

5.2 Methods (functions)

5.2.1 init

Function that is used to initialize the job search plugin. This function must be called. It will set up the application and register all default values and custom settings.

The function has up to four parameters that can be set to customize several behaiviours of the application:

- interface
- setting
- template
- data

Read chapter 6.2 on how to initialize the job search application
Read chapter 7. on how to customize your setup

5.2.2 clearFilter

The method clearFilter can be called to clear a search field (e.g. Country) that has been set. The method is available after the job search application has be initialized.

The clearFilter function allows a search field as a parameter. If no parameter is given to the function all fields that marked as default will be cleared. See chapter 5.1.1 for available API fields

Please note: If you set a fields as default (see chapter 7.2.2.3 for setting) these filters can not be cleared.

5.2.3 writeKeys

The writeKeys function is used to register a value which will be searched for when the search function will be called. Expects the following parameters:

- search field (mandatory)
- value (mandatory)

Please note: You can not overwrite the value of search fields that have been marked as default. Please read chapter 7.2.2.3. on how to mark search fields as default

5.2.4 readKeys

The function readKeys allows a search field to be passed along to returns its value. If no parameter is being passed to the function, it will return an object with all search fields and their values. See chapter 5.1.1 for available API fields

5.2.5 search

The search function is being called to fire a query to the search API. It will send all registered search fields and their value to the search API and returns the results. The search function allows the following parameters to register a search field before querying the search API:
- search field
- value

If no parameters are being passes to the function it will send a search query with the registered fields.

5.2.6 switchPage

The switchPage function allows to jump to the next page of the search results. The function is expecting a number as a parameter.

6 Installation

6.1 Upload application files

After downloading the source files from NBS the ZIP file must be unpacked and all mandatory files must be uploaded to your file system.

6.2 Initialize job search application

Edit the page where you would like the search application to show up. Add all stylesheets & scripts to the header section of that page:

```
1       <style type="text/css">
2       @import url(css/jobsearch.css);
3       </style>
4       <script type="text/javascript"
src="http://www.novartis.com/feeds/jobsearch/setup.js"></script>
5       <script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
6       <script type="text/javascript" src="js/jquery.jobsearch.js"></script>
7       <script type="text/javascript" src="js/swfobject.js"></script>
```

Line 2: This is the CSS file that contains all the styles for the job search application. This file can be customized to adjust the design & layout to your needs.

Line 4: This file includes all the data object to fill UI elements with values. Please link to the file published on www.novartis.com in order to keep your data in sync.
Line 5: jQuery main library used to run the job search application.
Line 6: Novartis job search jQuery plugin
Line 7: Library to render Flash onto the page. If you don't intend to use the Flash world map this line can be removed.

If you have changed the folder structure during the process of uploading files to your filesystem, please correct all links so they point to the appropriate file.

To initialize the job search application select the DOM element where the application should be placed and call the jquery.jobsearch plugin:

```
<script type="text/javascript" language="javascript">
	jQuery(document).ready(function(){
		jQuery('#jobsearch').jobsearch();
	});
</script>
```

The code shown above selects a DOM element (with the attribute “id” and the value “jobsearch”) and places the application inside this element. Therefore the DOM element has to be available in the body of your page where you would like the job search to be placed:

```
<div id="jobsearch"></div>
	<noscript>
		<p>Javascrpt must be enabled to use this application</p>
		<p class="arrow-link-internal"><a href="/careers/job-search/brassring/index.shtml">Go to the basic job search</a></p>
	</noscript>
</div>
```

If Javascript is not available the user would see a message and a link pointing to the old interface.

Initializing the application like this will load the job search application with all of its default values.

7 Customization of your application

7.1 Options for customization

If you would like to customize parts of the application you have can pass additional options to the application while initializing.

The following options can be customized when you first load the job search plugin:

7.1.1 interface

The interface can be completely customized. TOs can change the interface to remove and/or add UI element in order to adjust the interface to their needs.
e.g.: remove wold map & country list for installation on a Spanish website.

7.1.2 setting

Local settings to customize your installation of the job search application.
e.g.: set Spain as default country for an installation on a Spanish website.

7.1.3 template

The HTML code that will be returned from the job search application can be customized. Not only the information that will show up when searching for jobs but also the HTML code that will be return can be adjusted.
e.g.: don't show Division in job results for installation on a Pharma website.

7.1.4 data

This option allows TOs to create custom data sets. Those can be used to fill user interface elements with default values to interact with the search API.
e.g.: add data for “full time” & “part time” positions so users can filter for this.

7.2 Initialization with custom options

In order to initialize the application with your customizations TOs have to set up objects for these settings, so that the settings can be passed to the application:

```
var interface = {}; // custom interface
var setting = {}; // custom settings
var template = {}; // custom templates
var data = {}; // custom data

jQuery(document).ready(function(){
        jQuery('#jobsearch').jobsearch('init', interface, setting, template, data);
});
```

7.2.1 Customize interface

The interface object contains all UI elements to be loaded when the job search application is loaded at first.

The interface object consist of a set of objects which will define the user interface. Each object has properties that define what data to use, how they look like, how the user interacts with the UI objects and how the UI objects interacts with the search API.

The following UI objects will be loaded per default when the job search application is called without a custom interface:

- **country**: A-Z country list to select a country
- **worldmap**: flash worldmap to select a country
- **keyword:** fulltext search for keywords
- **function**: drop down menu to filter for a function
- **division**: drop down to filter for a division

In order to remove one of the default UI objects the interface object should have an empty value assigned to the UI object.

The following setup would remove all UI elements (country list & worldmap) to select a country:

```
var interface = {
        country : "",
        worldmap : ""
};
```

The following samle code would add a drop down menu to the interface that allows the user to filter by country:

```
var interface = {
        country : {
                dataset : 'country',
                type : 'select',
                wrapper : '<select class="first"><option value="" selected="selected">Select by country</option>$DATASET$</select>',
                template : '<option value="$DATAKEY$">$DATAVALUE$</option>'
        }
};
```

The job search plugin runs through the interface object to generate the user interface. This means the order of the UI objects in the settings object will reflect the oder of elements rendered onto your website.

The example code above shows the interface object containing a UI object (country) that has four different properties.

The properties that are available for UI object are:

7.2.1.1 dataset

(required, only optional for type = input):
UI elements that have a data object assigned will be filled with the appropriate data in order to submit the proper information to the search API. The job search application comes with the following data sets out of the box:

- country
- division
- function
- therapeuticarea

For other data objects see chapter 7.2.4. on how to create new data objects.

7.2.1.2 type

(required):
This value is used to assign a type to a UI element. The following UI types are supported by the job search application:

- select (drop down menu e.g. select by division)
- input (full text search e.g. keyword)
- custom

7.2.1.3 wrapper

(required):
HTML code to be used to wrap around each item of the data object. The HTML code allows TOs to add classes and code to design the interface to their needs.

Please note: In order to support full functionality and fill the UI element with the proper values the placeholder $DATASET$ should be present in your HTML template. This will be replaced with the items form the data object.

7.2.1.4 template

(required for type = select):
HTML code to be used as a template for every item in the data object. The HTML code allows TOs to add classes and code to design the interface to their needs.

Please note: There are two mandatory elements which must exist. $DATAKEY$ will be replaced with the data object's value used by the search API. $DATAVALUE$ will be replaced with the reader friendly version of the data object's item.

7.2.1.5 func

(optional):
Function that will be called after UI object has been initiated. This can be used to add event listeners to track the usage of an UI object for statistical purposes.

The type “func” also allows TOs to add customized UI objects to the interface that are not supported out of the box. If the type is set to “custom”, the property “func” can be used to generate the UI object, handle data that should be associated with it and deal with interactions between Users and search API.

7.2.2 Customize settings

The setting object allows TOs to set some default values for the job search application. The properties of the settings object are:

7.2.2.1 siteId

If you need to change the siteId because your querying a different gateway than the default one (see chapter 4.2) this setting can be overwritten by defining a new value for siteId.

In order to query the default gateway for global external positions in English the following setting can be used:

```
var setting = {
        siteId : "5260"
};
```

7.2.2.2 apiAddress

If you have already have changed your webserver to enable cross-site-scripting for this application the apiAddress key can be used to reference to the internal url.

7.2.2.3 urlapendix

In order to track how many traffic is being generated by your application the settings object allows you to add a url apendix. This value will be added to every link shown in job results. This allows analytics systems (e.g., WebTrends, Google ananlytics) to check for the url apendix to get filtered traffic.

If you would like to set up a url apendix please contact helpdesk.novartis@kenexa.com to get the value to associates to urlapendix for your site.

The url apendix used for the job search application on the global corporate website uses the following setup:

```
var setting = {
        urlapendix : "novcom",
};
```

Tip: For more advanced analytics systems you may want to use the func property of the UI object which allows TOs to set up event listeners to further enhance the tracking and usage of the job search interface.

7.2.2.4 defaults

This object allows TOs to set default data that will be sent to the search API. Every item of this object contains a KEY and a VALUE. This will be registered and the interface will not be able to overwrite the default settings. The KEY must match the field available in the search API.

If the application is installed in a country the default settings can be adjusted in order to always submit a county VALUE to the search API in order to get only positions from within this country. e.g.:

```
var setting = {
        defaults : {
                county : "CH"
        }
};
```

7.2.2.5 usDisclaimer

This object allows TOs to enable the US-Disclaimer functionality.
If the value is 1, the functionality is enabled;
If the value is 0, the functionality is disabled. (DEFAULT)

When the usDisclaimer functionality is enabled:
- If after a query the results list contain at least “one” job in the USA, then a link appears in the page (above the jobs list);

It’s possible to customize the “Text” and the “Link” that appears on the page.

```
var setting = {
        apiAddress: 'http://www.novartis.com/brassring/WebRouter.asmx/route',
        usDisclaimer: 1,
        usDisclaimerLinkText: 'Learn more',
        usDisclaimerLinkUrl: 'http://www.novartis.com'
};
```

7.2.3 Customize template

The template object allows TOs to customize the output of the job search application. Not only what information will be generated but also how the output is coded (HTML). This allows TOs to adjust the output to CSS code and coding style.

7.2.3.1 header

The header template will be used to show the number of the results as well as a placeholder for the searches/filters applied. The HTML code if changed should contain the following placeholders in order to support all functionality:
- $NUMRESULTS$ (will show the number of positions returned)
- $FILTERS$ (will be replaced with the active filters applied)

7.2.3.2 results

The results returned by the job search application can be customized to show different information along with the job listings. The results object includes the following two templates which must be present:
- wrapper
- template

The wrapper property includes the HTML code that should be wrapped around the job listings. The only mandatory element is the placeholder $JOBRESULTS$ which will be replaced with the job listings.
The template property includes the HTML that will be used to render each job listing onto the page. TOs can chose from a selection of placeholders to display the information that should be shown (see section 5.1.2 on job results placeholders).

7.2.3.3 footer

The footer object has three properties that have to be present:

- wrapper
- template
- number

The wrapper property includes the HTML code that should be wrapped around the footer/pagination. There is no mandatory field but the template must only include a single DOM element where the pagination will be place in.

The template will be used to render the pagination. There are two mandatory placeholders that have to be present:

- $STATUS$ (used in a class to mark current page active)
- $PAGE$ (shows the number of the page)

The number property is used to define how many steps ahead/reverse the user should be see from the active page in the pagination.

7.2.4 Customize data objects

Every data object contains pairs of KEY and VALUE that will be submitted to the search API. If you would like to set up custom data set please contact helpdesk.novartis@kenexa.com to get the proper KEYS and VALUES.

The data object must exist as a field in the search API (check chapter 5.1.1. for default API fields).

To set up a data object for the employment type the following data set would be created. This data object can now be used by the interface to create a drop down menu or alike.

```
var data = {
        employmenttype : {
                Permanent : "Permanent",
                Temporary : "Temporary",
                Internship : "Internship"
        }
};
```

8 Sample code

8.1 Job search application for country (example: Switzerland)

This is some example code how a job search interface can be set up for Switzerland to query jobs across all divisions and functions. Country selectors (list and map) are removed from the interface. A new default setting for Switzerland is set and the application gets initialized:

```
var interface = {
        country : "",
        worldmap : "",
};

var setting = {
		defaults : {
			country : "CH"
		},
	};

	var template = {};
	var data = {};


	jQuery(document).ready(function(){
		jQuery('#jobsearch').jobsearch('init', interface, setting, template, data);
	});
```

8.2 Job search application for division & employment type (example: Pharma)

This sample code shows a job search interface that only returns open positions in Pharma filtered by employment type. The UI for Division gets removed and a new UI for Employment type gets added. Default value for Pharma get set and the data container for the new UI element is being filled with values:

```
	var interface = {
		division : "",
		employmenttype : {
			dataset : 'employmenttype',
			type : 'select',
			wrapper : '<select class="first"><option value=""
selected="selected">Select by employment type</option>$DATASET$</select>',
			template : '<option value="$DATAKEY$">$DATAVALUE$</option>'
		}
	};

	var setting = {
		defaults : {
			division : "Pharma"
		},
	};

	var template = {};
	var data = {
			employmenttype : {
				Permanent : "Permanent",
				Temporary : "Temporary",
				Internship : "Internship"
			}
		};


	jQuery(document).ready(function(){
		jQuery('#jobsearch').jobsearch('init', interface, setting, template, data);
	});
```

8.3 Latest jobs for division in Spain (example Corporate)

This sample code shows the latest jobs in Corporate based in Spain. First the entire interface gets disabled. The default settings for Spain and Corporate get set. The templates for the job results are slightly modified. Finally the job search gets initialized and the search gets triggered to show the results:

```
	var interface = {
		function : "",
		country : "",
		keyword : "",
		division : "",
		worldmap : ""
	};
	var setting = {
		defaults : {
			country : "ES",
			division : "Corporate"
		}
	};
	var template = {
		header : ""
	};
	var data = {};

	jQuery(document).ready(function(){
		jQuery('#jobsearch').jobsearch('init', interface, setting, template, data).jobsearch('search');
	});
```

9 Glossary

AJAX	Asynchronous Javascript and XML
API	Application programming interface
CSS	Cascading Style Sheet
DOM	Document Object Model
JS	Javascript
TO	Technical owner
UI	User interface
User	Site visitors who interact with your application