Programmer's Manual
JRC – MYGEOSS app for Protected Areas/Sites Natura 2000
project 2016.2433

Table of Contents

1 Technology/Installation

1.1 Introduction

As identified in the Software Architecture Document (SAD), the application uses both Ionic and Cordova. Therefore, the web technologies, HTML, CSS, and JavaScript are used with the AngularsJS framework.

To be able to change, modify, update, or upgrade the application, the programmer must have experience with these common web technologies, along with AngularJS, Cordova and Ionic to both install plugins and build the application for each platform.

Specific hardware requirements for the development include an Apple Mac for the iOS.

1.2 Installation of your workspace

To set up your working environment, you will need to import the following into your working directory:

- Config.xml
- Hooks
- Ionic.project
- Package.json
- Resources
- Scss
- www

Once imported run "`npm install`". This will install all of the modules that ionic needs. Once this has completed run "`ionic prepare`" (or `cordova prepare`). Platforms versions and the plugins version saved in the `config.xml` file will then be installed.

This project uses Sass to generate the CSS files. You can add your own CSS file but the main one `www/css/ionic.app.css` is generated by `./scss/ionic.app.scss`. Please refer to the Ionic documentation to know how to setup Sass on your workspace.

2 Plugins, libraries and platforms

While updating the HTML, CSS, and JavaScript files will, in principle, pose no issue to the application, updates to the platforms and plugins used must be carried out with caution as updates to the actual application may also be required to ensure continued compatibility. This section explains which plugins can, and should, be updated, and the steps to do so safely.

2.1 Platforms

The application uses the Cordova iOS platform v.4.1.1 and the Cordova Android platform v.5.2.2. The developer must evaluate the impact to potentially outdated plugins, Ionic libraries, or the core application, prior to updating the platform.

2.2 Plugins

2.2.1 Plugins list

The list of all Cordova plugins used is saved in the `config.xml` file inside the root folder. These plugins can be found inside the `plugins` folder and are listed below:

```
com.telerik.plugins.nativepagetransitions 0.6.5 "Native Page
Transitions"
cordova-plugin-android-permissions 0.10.0 "Permissions"
cordova-plugin-camera 2.2.0 "Camera"
cordova-plugin-compat 1.0.0 "Compat"
cordova-plugin-console 1.0.4 "Console"
cordova-plugin-device 1.1.3 "Device"
cordova-plugin-file 4.3.0 "File"
cordova-plugin-geolocation 2.4.0 "Geolocation"
cordova-plugin-globalization 1.0.4 "Globalization"
cordova-plugin-inappbrowser 1.4.0 "InAppBrowser"
cordova-plugin-network-information 1.3.0 "Network Information"
cordova-plugin-screen-orientation 1.4.2 "Screen Orientation"
cordova-plugin-splashscreen 3.2.2 "Splashscreen"
cordova-plugin-statusbar 2.1.3 "StatusBar"
cordova-plugin-whitelist 1.2.2 "Whitelist"
cordova-plugin-winstore-jscompat 0.0.1 "Windows Compatibility"
cordova-sqlite-storage 1.4.8 "Cordova sqlite storage plugin"
cordova.plugins.diagnostic 3.2.2 "Diagnostic"
eu.europa.ec.ecas 0.0.5 "ECASMobile"
ionic-plugin-keyboard 2.2.1 "Keyboard"
```

2.3 JavaScript / AngularJS modules

Custom added JavaScript libraries/modules and font can be found under the application `www/lib` folder.

ionic-native-transitions.min (https://github.com/shprink/ionic-native-transitions)
leafet.js v0.7.7 (http://leafletjs.com/)
esri-leaflet v2.0.2 (https://github.com/Esri/esri-leaflet)
ngCordova v0.1.27-alpha (http://ngcordova.com/)
angular-translate.min v2.9.0 (https://github.com/angular-translate/angular-translate)
angular-translate-loader-static-files.min v2.13.0 (https://github.com/angular-translate/bower-angular-translate-loader-static-files)

2.4 Update

To update a plugin, simply replace the existing file with the new one, keeping the same name.

As the plugins can have many changes, fixes, or even deprecation, the recommendation is to test the application after each individual update. This will ensure the update will not break the application.

For the Cordova plugins, please use the Cordova or Ionic command line tool to add/remove/update a Cordova plugin.

For the Cordova platforms please use the Cordova or Ionic command line tool to add/remove/update a platform.

Command line usage:

https://cordova.apache.org/docs/en/latest/guide/cli/index.html

3 Architecture

The application has been entirely written using a Model View View Model (MVVM) architecture, based upon AngularJS. This allows application dynamic binding, which means that the views can refresh without reloading the application.

3.1 app.js

In the `app.js` file we find:

- The router. Showing the controllers used by the template and the corresponding state.
- The .run function. The code that is first executed.
- The .config functions. Functions used to configure the angular and the ionic defaults values.

3.2 constants.js

This file contains the configuration constants of the application and the static text that can be found in the application.

In this file you can configure which server the app will connect to and the parameters for the alerts.

- staticfolder: The ENDPOINT with all html static files content.
- server: The ENDPOINT where send and retrieve users' observations/overall feedbacks
- nopic: SVG URL encoded picture to display when no picture is available for one site

3.3 directives.js

This file contains the custom directives.

`bindUnsafeHtml`	The directive inside this file enables the data-binding in the template with an HTML code retrieved from an external source.

3.4 Filter.js

This file contains the custom filters.

`filterOnlineReps`	Filter sent reports.
`filterDrafts`	Filter saved draft.
`filterSites`	Filter site list.

3.5 controllers.js

Generally, there is one controller per template; please refer to the routing source code in `app.js` to have more details.

Template	Controller
`app.html`	`AppCtrl`
`home.html`	`HomeCtrl`
`about.html`	`AboutCtrl`
`list.html`	`ListCtrl`
`sitesmap.html`	`MapCtrl`
`site-menu.html`	`SiteMenuCtrl`

site-pics.html	SitePicsCtrl
site-info.html	SiteInfoCtrl
site-feedback.html	SiteFeedbackCtrl
site-map.html	SiteMapCtrl
report-inner.html	SiteReportCtrl
report.html	ReportCtrl
report-observation.html	ReportObservationCtrl
report-feedback.html	ReportObservationCtrl
myreports.html	MyReportsCtrl
mysentreports.html	MySentReportsCtrl
login.html	LogInCtrl

Controller	Description
AppCtrl	Generic controller for the entire app, used to setup the application and the menu modal.
HomeCtrl	Controller for the home page.
AboutCtrl	Controller for the about page.
ListCtrl	Controller to display the list of sites embedded in the application.
MapCtrl	Controller for the contact page.
SiteMenuCtrl	Main controller for all the sites information tab.
SitePicsCtrl	Site picture tab controller.
SiteInfoCtrl	Site information tab controller.
SiteFeedbackCtrl	Site feedbacks tab controller.
SiteMapCtrl	Site map tab controller.
SiteReportCtrl	Site add an observation/feedback tab controller.
ReportCtrl	Controller for choosing either to report an observation or an overall feedback.
ReportObservationCtrl	Controller for the report an observation page.
ReportObservationCtrl	Controller for the report an overall feedback page.
MyReportsCtrl	Controller for the locally saved reports.
MySentReportsCtrl	Controller for the reports already sent to the server.
LogInCtrl	Controller for the login.

3.6 service.js

This file contains the services of the app. This is the common function and logic of the app.

n2db	Setup and manage database. Retrieve static content from the server, send and save reports. Process login/logout

`n2serv`	Get local data and server observations/feedbacks.
`$language`	Get users preferred device language and store selected language.
`$localStorage`	Use the device local storage to store persistent data.

3.7 site.js

Javascript file containing an array variable with all the available natura sites. (Extract from: http://www.eea.europa.eu/data-and-maps/data/natura-7)

3.8 extra_data.js

Array of available country for site filtering.

4 Config.xml

This file at the root is used to configure the application. Some of this information can be overridden prior to application store publication by opening the iOS xcodeproject, the android graddle files or the Windows visualstudio project files.

`<widget>` set the bundle.id and the version number. More information can be found inside the Publications Office document "*Guidelines for mobile apps publishing on the common European Union accounts*".

You can configure some native parameters on each platform `<platform name="ios"><platform name="android">` such as the minimum OS version device etc. See the Cordova documentation for this file:

`https://cordova.apache.org/docs/en/latest/config_ref/index.html`

You can retrieve the platform version and the plugin version from this file. See plugin and platform management from the doc:

`https://cordova.apache.org/docs/en/latest/platform_plugin_versioning_ref/`

5 Local databases

The application uses an SQLite local database to store persistent data as cached documents or saved documents. This is handled by the `cordova-sqlite-storage` plugin. More information about the fields of the database can be found in the Software Architecture document.

6 Assets

All images and icons are inside the `www/img` folder. The fonts are inside the `www/lib/fonts` folder.

CSS files are generated from SASS. (Please refer to the Ionic document to know how to enable scss in ionic project).

Icons and splashscreens are in the root folder into `./resources`. The `config.xml` file defines the path to these assets for each platform

7 Upgrade

To upgrade the app, once you have modified your files, use the `cordova/ionic` command line tool to provide the files for each platform and build it to provide the native application file. Ensure the same `bundle.id` is in `config.xml` and change the version number.

Command line usage:

https://cordova.apache.org/docs/en/latest/guide/cli/index.html

Configuration information:

https://cordova.apache.org/docs/en/latest/config_ref/index.html

Plugin and platform versioning:

https://cordova.apache.org/docs/en/latest/platform_plugin_versioning_ref/

8 Adding a new language

Available language at the 22nd of December, 2016:

- English
- German

Language code are decalred in ISO-639-1: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

In `www/js/controller.js`, add the new language in the `$scope.languageList` array variable:

```
$scope.languagesList = [
    {label: 'de - Deutsch', idL: "de"}
];
```

In `www/js/service.js`, `$language` factory. Add the language in the list of available language. (To prevent the application of setting up the language of the app to the device users language if it not exists). `obj.availableLanguageKey = ["en", "de"];`

Create the translated entry file in the `www/languages` directory.

`locale-[language code].json`

Add the translation to the corresponding key. (Be sure your JSON file is valid or it can crash the application).

Create the SVG pictures translated, from the source code. (`.ai` source file is included in the deliverables). Then place the created SVG in `www/images/home_icon/[file_name]-[language code].svg`.

For the static content updated from the server (about, contact, link, disclaimer, acknowledgment…). Place a copy of each html file in `www/static/[language code]`. And on the server you should have the same architecture. One folder per languages.

Currently the statics files are here: http://digitalearthlab.jrc.ec.europa.eu/files/app/mynatura2000/

Note that for the moment any user's observation is translated.

9 General comments

More information about the logic can be found in comment inside the source code, and the function and variable have explicit names.

9.1 Updating static content

To update static content, just update the HTML file on the server. The app will check if the file is newer or not and then the database will be updated.
http://digitalearthlab.jrc.ec.europa.eu/files/app/mynatura2000/

- About.html
- acknowledgment.html
- disclaimer.html
- legalNotice.html
- privacyStatement.html

9.2 Detecting nearby areas

The application can detect if an user is close to a protected area and display an alert. This feature is only active when the application is running.

To detect a nearby area, the application retrieves the user position (this is managed by the main controller `'AppCtrl'`) and checks (function '`getNearbySite`' in the '`n2serv`' service) if there is any site which has its center, whose coordinates are specified in the `site.js` file, close to that position. The current threshold to define a site as 'nearby' is set to a distance minor or equal to 5km from the user position – this value can be changed in the abovementioned service. If more than one site is close to the user, only the closer one is passed back to the controller. Then an alert is shown which optionally redirects the user to the site details page.

END OF DOCUMENT

EPUB_Lot1_MAN_2016.2433_PM_v010