Programmer's Manual for the JRC – MYGEOSS app for Invasive Species project 2015.3724

Table of Contents

1 Technology

As identified on the Software Architecture Document (SAD), the application use ionic and cordova, so the web technologies are used (HTML, CSS, Javascript). The AngularsJS framework is used.

To be able to change, modify, update or upgrade the application, the programmer must have experience on common web technologies, AngularJS and know how to use Cordova and Ionic to install plugins and build the app for each platform.

2 Plugins

While updating the HTML, CSS and JavaScript files will, in principle, pose no issue to the application, the used libraries update must be carried out with caution as they might break the actual application. This section explains which are the plugins that can, and should, be updated, and the steps to do it safely.

2.1 Location

The plugins and custom added libraries are located under the application `www/lib` folder. The following list of plugins can be found:

```
ng-cordova.js
angular-base64.min.js
angular-touch.min.js
angular-base64.min.js
angular-thumbnail.min.js
truncate.js
angular-carousel.min.js
leaflet.js
jsencrypt and its  dependencies
(https://github.com/travist/jsencrypt)
framework7 (http://framework7.io/) v. Framework7 1.4.2
hammer.js
```

A patch for iOS9 and AngularJS, iOS9 can provide infinite `$digest` without this patch:

```
angular-ios-9-uiwebview.patch.js
```

The list of all Cordova plugin used is in `package.json` file inside the root folder. And theses plugins are inside the `plugins` folder. The following list of Cordova plugins can be found:

```
cordova-plugin-network-information
cordova-plugin-inappbrowser
org.apache.cordova.file
cordova-sqlite-storage
cordova-plugin-datepicker
org.apache.cordova.geolocation
org.apache.cordova.camera
cordova-plugin-statusbar
cordova-plugin-splashscreen
cordova-plugin-whitelist
cordova-plugin-console
```

```
cordova-plugin-device
```

2.2 Update

To update a plugin, simply replace the existent file with the new one and 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 did not break the application.

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

Command line usage:

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

Plug-in information:

http://ngcordova.com/docs/plugins/

3 MVVM Architecture

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

3.1 app.js

In the `app.js` file we find:

- the router, it shows the controllers used by the template and the corresponding state.
- the .run function. (The code that is first executed). We check if the device is online or offline, try to upload the pending sob. We open the SQLite database and create the table if they don't exist yet.
- all of the controllers.
- The .config functions. Functions used to configure the angular and ngCordova plugins, and the ionic defaults values.

3.1.1 Controllers

Generally one controller per template; refer in routing source code to see which template uses which controller.

Controller	Template	Info
`AppCtrl`	App.html	General controller for the app.
`HomeCtrl`	Home.html	Home page controller.
`SpecieListCtrl`	specieList.html	Liste of species controller with filters.
`SpecieCtrl`	specie.html	Species details controller.
`ReportSightingCtrl`	reportSighting.html	Report a sighting main page controller.
`SOBCtrl`	sob.html	User's observations controller.

MyRecordsCtrl	my_records.html	User's sent records and saved draft controller.
ContactCtrl	contact.html	Contact controller.
LinksCtrl	links.html	Links controller.
AboutCtrl	about.html	About the application controller.
SightingMapCtrl	sightingMap.html	Main map controller.
LoginCtrl	login.html	Login page controller.

More information about the functions inside the controllers is available in section 8.

3.2 Config.js

All the Constant variables are defined here; there is the contact mail address and the web Rest API base address, the authenticationBaseURL and the sessionExpirationTime.

We find also the label for the popup message inside the app.

3.3 services.js

Inside this file, the services, factory and provider for angular. These functions are used to manage data and share data between controllers:

$speciesFactory	Retrieve the species local information.
$easinFactoryREST	CRUD base method to communicate with the REST service.
$photoFactory	Method to manage the photo, (read/convert to base64..)
$easinFactory	Method to send the observation to the REST service.
$easinFactoryLocal	Manage all the user's observations in the database.
$dataBaseFactory	Set/retrieve the database.
$geolocationFactory	User's device geolocalisation.
$dateFactory	Date information and Date picker.
$networkFactory	Retrieve/store network state.
$authenticationFactory	Useful function for the authentication.
$localstorage	Use the device localstorage to store permanent data.

3.4 Directives.js

This file contains all the angularJS custom directives. The directives `speciePhotos`, `specieInformation`, `specieMap` and `specieAddObservation` are used conditionally inside the template `partials/specie.html` to provide a tab system.

`sobPhotos` and `sobInformation` are used conditionally inside the template partials/sob to provide a tab system.

`externalLinks` add a custom attribute to the `<a>` element for open the links with the default device browser.

The directive `specieAddObservation` is used inside the template partials/report.html too.

`specieMap` init a map with leaflet.js (see the documentation http://leafletjs.com/). The data-tap-disable must be used in parameter for the `<ion content>`; failure to do so could create unexpected behaviour with map interaction.

`speciePhotos`	Tab with species pictures.
`specieInformation`	Tab with species information.
`specieMap`	Tab with species sighting on the map.
`specieReportSighting`	Tab with form to report a sighting. (Also used like a normal view in the `ReportSightingCtrl`.
`sobPhotos`	Tab with user's observation photos.
`sobInformation`	Tab with user's observation information.
`backImg`	Directive used to display a dynamic background images within a ionic collection-repeat list. Used in the species list controller.

More information about the functions inside the directives is available in section 8.

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 or the android graddle 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

5 Assets

The pictures and thumbnails for each species are in the `www/data` folder. All other images and icons are inside the `www/img` folder.

5.1 CSS and SCSS

To compile the SCSS, be sure to modify the file `gulpfile.js` to point the SCSS file inside the `www/scss` folder not the one at the root.

If you change the CSS file location ensure that the `www/index.html` is also updated.

6 Upgrade

To upgrade the app, modify 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

7 API Rest DATA

7.1 Example of json object to send to the JRC server:

For the moment the API authorize just to get all the data: `GET /reports`, or one entry: `GET /reports/idReport`. There are no filters, and the images are stored in base64 format inside the json file, so the time to retrieve the entire file could be extensive.

```
{
    "properties": {
        "LSID": String,
        "ICCID": String,
        "OAUTHID": String,
        "Abundance": String,
        "Precision": String,
        "Comment": String,
        "Image": Array,
        "Status": String,
        "Anonymous": Boolean,
    },
    "geometry": {
        "coordinates": Array of Integer,
        "type": String
    },
    "type": String
}
```

properties: Object

LSID: The species JRC LSID. (See http://easin.jrc.ec.europa.eu/use-easin/species-search/species-search-by-name)

ICCID: Device UUID (not possible to obtain the ICCID)

OAUTHID: An identifiant for the login, not yet implemented, just enter a empty string.

Abundance: Number of species + unit.

Precision: Precise or estimated count.

Comment: SOB Comment

Image: An array of base64 picture

geometry: Object

coordinates: Array of coordinate. For the moment just x;y

type: For the moment “Point”.

type: For the moment “Feature”.

7.2 Example of json object receipt from the JRC server:

```
POST /reports
{
      "_id": "563b3c91c1f235e130bb6d6c"
      "createdAt": "2015-11-05T11:25:05.382Z"
      "updatedAt": "2015-11-05T11:25:05.382Z"
      "__v": 0
      "properties":
            {
            "ICCID": "2948592846928745"
            "OAUTHID": "cba@gmail.com"
            "LSID": "urn:lsid:alien.jrc.ec.europa.eu:species:R02004:4.6"
            "Abundance": "35"
            "Precision": "Approximate"
            "Comment": "whatever I find worth adding"
            "Anonymous": false
            "Image": ["images/img1.jpg"]
            "Status": "Validated"
            }
      "geometry":
            {
            "coordinates":
            0:  47.940380134831656
            1:  14.793847073186715
            "type": "Point"
            }
      "type": "Feature"
}
```

8 API Authentication/Registration

For detailed information on the REST service authentication END POINT, please see the EASIN.AUTHENTICATION.WEB.API.doc provided by the JRC.

Data need to be encrypted prior to sent it to the server. (RSA encryption). To do it we use the `jsencypt.js` library. It is only a one-way encryption, we don’t have to decrypt any data received.

9 Code Documentation & logic

To change the header color dynamically, use this code:

```
$scope.$on('$ionicView.beforeEnter', function() {
        $rootScope.headerColor = '#27393D';
    });
```

The ionic cache is disabled to avoid conflict issues.

For further documentation and application logic, please refer to the application source code.

9.1 REST custom cache

For this release, the backend server does not propose any filter system or pagination system. It could result in a high download time when the app requests all the observation. To avoid requesting many time the same resource, all the GET request are cached using a custom cache system provided by angular. (See `$cacheFactory('customQueryCache');`). This custom cache is deleted only when the user sends an observation, so the data are refreshed and he can see his observation.

9.2 Carousel/ Photo browser

To perform the ”native iOS like photo browser”, we use a custom implantation of the photo browser component from the framework7 library. http://framework7.io/docs/photo-browser.html.

By default the photo browser component create a photo browser that override our view. To include it in your body we need to modify the famework7.js file.

Add this parameter: `domInsertion: "body"`

```
if (pb.params.type === 'standalone') {
    $('body').append(htmlTemplate);
}
```

---

is changed too :

```
if (pb.params.type === 'standalone') {
    var dom = pb.params.domInsertion;
    $(dom).append(htmlTemplate);
}
```

With this, you can select where in your DOM you want the code to be injected. By default we were limited to the body. (It appeared like a modal.)

On Android the pinch zoom does not work. Use the `hammer.js` library, when you initiate your photo browser instance in the controller (or whatever you want to instantiate it), in the parameters, change the `onOpen` callback:

```
onOpen: function (pb) {
  var target = pb.params.loop ? pb.swiper.slides : pb.slides;
```

```
  target.each(function( index ) {
    var hammertime = new Hammer(this);
    hammertime.get('pinch').set({ enable: true });
    hammertime.on( 'pinchstart', pb.onSlideGestureStart );
    hammertime.on( 'pinchmove', pb.onSlideGestureChange );
    hammertime.on( 'pinchend', pb.onSlideGestureEnd );
  });
}
```

END OF DOCUMENT

EPUB_Lot1_2015.3724_PM_v020.doc