Advanced country selector for forms.

Country Selector Plugin - Reference Documentation

Authors: Matouš Kuera

Version: 1.0

Table of Contents

1 Introduction

1.1 Change Log

1.2 Roadmap

1.3 Known Issues

2 Configuration

2.1 Countries i18n

2.1.1 Countries Customization in i18n

2.1.2 Supported Languages

3 Utility Clases

3.1 CountrySelectorService

4 Autocomplete

4.1 jQuery Autocomplete

4.2 Twitter Bootstrap Autocomplete

4.3 Custom Autocomplete

1 Introduction

This plugin provides advanced country selector for Grails. It is suitable for all forms where the user nee address form. Contrary to traditional html select with a huge list of countries, it serves a user normal input type his country. When he starts typing it suggest him possible countries he could fill in.

It uses JavaScript autocomplete to give the user right help. The autocomplete takes into account country different relevance for different countries (see Countries i18n chapter). Moreover, it allows i18n localiz alternatives and also relevancy.

The plugin is based on JavaScript implementation of country selector of Baymard Institute, see project Git

1.1 Change Log

Date	Version	Notes
2013-11-13	1.0	Release of 1.0. TagLib for display of country name based on its code. Locale attribu
2013-07-03	0.2.6	Improved dependencies. Resource plugin is not required by default.
2013-07-02	0.2	Bug fixes. German translation.
2013-07-01	0.1	Smart country selector - <cs:countrySelector /> tag. jQuery autocomplete im Bootstrap autocomplete implementation. Country setup in i18n.

1.2 Roadmap

- Cashing of parsed i18n country codes
- Translations to CZ

1.3 Known Issues

2 Configuration

There are few configuration options for the plugin:

Name	Default	Description
grails.plugin.countrySelector.messageSource.useOnlyCustom	false	True if only custom i18n coun used. See more in Countries Cust

2.1 Countries i18n

The plugin is dependent on i18n Grails plugin and allows to set all language/location properties need Therefore, simply changing the user locale can change the country selector behavior.

It allows to set-up following properties for each country:

1. **Country Code** - Plugin uses ISO 3166-1 alpha-3 codes for countries but you can set your own. Cour returned when the user submit the country selector. They could be easily mapped to domain class pro *DEU* , etc.

2. **Country Name** - It is the actual country name the user sees in autocomplete. Examples: *United King*

3. **Country Alternatives** - Alternative strings which should be used for autocomplete. When th alternative he should receive a proper country name. Examples: *GB Great Britain England UK W Ireland* for United Kingdom, *DE Bundesrepublik Deutschland* for Germany.

4. **Relevancy** - The relevancy specifies the order of results in autocomplete. Country selector sorts relevancy first (higher relevancy ~ higher position in autocomplete) and than alphabetically. E relevancy *3.5* , United Kingdom relevancy *2.5*

Having these properties in i18n allows to set different values for different locale. It allows for example to for different locale, i.e. Sweden would have higher relevancy than Switzerland in Sweden locale but this Switzerland locale.

2.1.1 Countries Customization in i18n

The structure of i18n country properties is following:

```
org.grails.plugin.countrySelector.CZE=Czech Republic
org.grails.plugin.countrySelector.CZE.alternatives=CZ eská Ceska
org.grails.plugin.countrySelector.CZE.relevancy=1.5

org.grails.plugin.countrySelector.DNK=Denmark
org.grails.plugin.countrySelector.DNK.alternatives=DK Danmark
org.grails.plugin.countrySelector.DNK.relevancy=1.5
```

The country code is the last part of the first property key, the value specifies the country name. Alternati names. Similarly, relevancy is used for country position in the country selector.

By default custom property files are merged with the plugin specified. You can set all country specificati relevancy) or just few of them. The custom specification has higher priority during merging. New countri is required.

You can also decide to use your custom i18n settings by default (for example, this can be used when y custom country codes, or just reduce the number of possible countries). Set following in Config.groovy

```
grails.plugin.countrySelector.messageSource.useOnlyCustom = true
```

2.1.2 Supported Languages

English ang Germany are the only one supported languages right now. The whole setup of country relevancy is in *i18n/messages.properties* file of the plugin core.

3 Utility Clases

3.1 CountrySelectorService

allowedCountryCodes() This method allows to get all allowed country codes. The general usage is for do

Example domain class Address.groovy

```
class Address {
   def countrySelectorService

String country

static constraints = {
      country(nullable: false, blank: false, validator: {val, obj ->
               def allowedCountryCodes = obj.countrySelectorService.allowedCountr
               if(!allowedCountryCodes.contains(val))
                  return "org.grails.plugins.countrySelector.country.notAllowed"
            })
   }
}
```

⚠ The Address class is usually an embedded class placed in groovy/src for a certain domain cla
the countrySelectorService, you need to inject the service differently:

```
import grails.util.Holders

class Address {
   String country

static constraints = {
      country(nullable: false, blank: false, validator: {val, obj ->
               def allowedCountryCodes =
obj.countrySelectorService.allowedCountryCodes()
               if(!allowedCountryCodes.contains(val))
                  return
"org.grails.plugins.countrySelector.country.notAllowed"
            })
   }

private static getCountrySelectorService() {
      def grailsApplication = Holders.getGrailsApplication()
      grailsApplication.mainContext.countrySelectorService
   }
}
```

4 Autocomplete

JavaScript autocomplete engine is an important part of this plugin. It allows the user to see the recommend
typing. By default, the jQueryAutocomplete is used by the plugin.

4.1 jQuery Autocomplete

jQuery autocomplete is the default autocomplete. Therefore, this plugin depends on jquery-ui plugin.

If you don't want to use this autocomplete and jquery-ui dependency than simply exclude it from BuildCor

```
compile ':country-selector:0.1' {
   exclude 'jquery-ui'
}
```

4.2 Twitter Bootstrap Autocomplete

The Twitter Bootstrap Autocomplete is an alternative implemented within the plugin.

You can set this implementation using the resources (Resource Plugin) in ApplicationResources.groovy:

```
overrides {
   'country-selector-plugin-js' {
      dependsOn 'country-selector-plugin-bootstrap-js'
   }
}
```

The Twitter Bootstrap country selector is dependent on *bootstrap-typeahead* therefore Twitter Bootstrap G

```
plugins {
…
    runtime ':twitter-bootstrap:2.3.2'
…
}
```

4.3 Custom Autocomplete

Similarly to Twitter Bootstrap Autocomplete you can set your custom autocomplete JavaScr
ApplicationResources.groovy:

```
'country-selector-plugin-custom-js' {
   dependsOn 'jquery'
   resource url: 'js/application/customCountrySelectorAutocomplete.js', dispositi
}

overrides {
   'country-selector-plugin-js' {
      dependsOn 'country-selector-plugin-custom-js'
   }
}
```

The custom JavaScript autocomplete should follow the implementation of Baymard Institute JavaScrip
Github for more details).