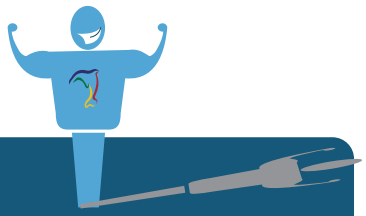
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

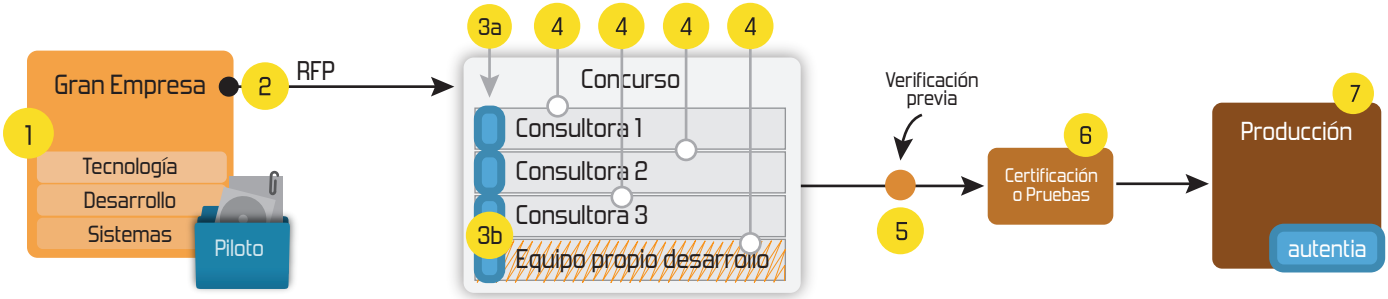
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.

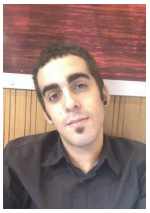


4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com



Rafael Macías Rodríguez

Rafael es un alumno becario de prácticas en Autentia S.L
procedente del I.E.S. Rey Fernando VI

Twitter: Seguir a @Rafa_g3n 114 seguidores

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2013-04-26 Tutorial visitado 11 veces Descargar en PDF

Internacionalizar una aplicación creada con Ember.

0. Índice de contenidos.

1. Entorno

Este tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Intel Core 2 CPU T7200 @ 2.00GHz x 2
- Sistema Operativo: Ubuntu 12.04 LTS x32
- Sublime Text 2
- Google Chrome
- GitHub

2. Introducción

Después de haber explicado cómo modularizar nuestra aplicación con Require.js (Ver tutorial), voy a enseñaros como internacionalizar nuestra aplicación.

Código de ejemplo: Enlace a github

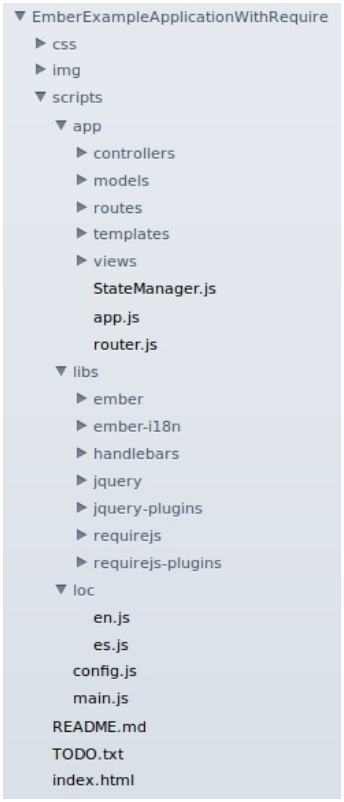
Para traducir nuestra aplicación podemos apoyarnos en alguna librería de internacionalización, Ember nos provee una: enlace.
La descargaremos, y la incluiremos como dependencia.

Ver cómo

3. Crear nuestros ficheros de textos

Para internacionalizar nuestra app tendremos que crear unos ficheros de texto:

- Vamos a aprovechar la modularización que teníamos con Require:



- Por lo tanto para internacionalizar crearemos un fichero para cada lenguaje, en nuestro ejemplo la internacionalización va a ser con Español e Inglés, ver siguiente paso.
- Vamos a tener definidas unas variables comunes para cada lenguaje, pero con un valor concreto, es decir, si definimos un título de bienvenida tendríamos:

```
//en.js
main.welcome.msg: 'Welcome'
//es.js
main.welcome.msg: 'Bienvenido/a'
```

- Estas variables las utilizaremos después en nuestro template, y funcionarán acorde al lenguaje seleccionado.
- Así quedarían nuestros ficheros de texto con todo el contenido de la aplicación traducido, para nuestro ejemplo son ficheros cortos, pero el hecho de estar separados es para evitar si fueran más largos llenar la memoria al cargar un fichero grandísimo con ambos lenguajes.

```
//es.js
define(function(){
    loc = {
        'main.application.title': 'Primera Aplicacion con Ember.js',
        'index.login.fail.mismatch': 'Los nombres no coinciden',
        'index.login.fail.empty': 'Completa los campos',
        'index.login.loginButton': 'Conectar',
        'index.login.nameLabel': 'Introduce tu nombre: ',
        'index.login.nameRepeatLabel': 'Repite tu nombre: ',
        'menu.login.noLog': 'No ha introducido ningun nombre',
        'menu.login.backToIndex': 'Volver',
        'menu.loged.welcomeMsg': 'Bienvenido',
        'menu.button.colorListButton': 'Ver Lista Colores',
        'menu.button.selectedColorButton': 'Ver Color Seleccionado',
        'menu.selectColor.selectColorText':'Seleccione un color: ',
        'menu.selectColor.selectedColorText':'Color Seleccionado: ',
        'menu.selectColor.selectColorButton':'Seleccionar',
        'menu.watchColor.selectedColorText':'Color Seleccionado: ',
        'menu.watchColor.selectColorButton':'Volver al menÃº',
    };
    return loc
})
//en.js
define(function(){
    loc = {
        'main.application.title': 'First App with Ember.js',
        'index.login.fail.mismatch': 'The names do not match',
        'index.login.fail.empty': 'Required fields',
        'index.login.loginButton': 'Sign in',
        'index.login.nameLabel': 'Type your name: ',
        'index.login.nameRepeatLabel': 'Re-type your name: ',
        'menu.login.noLog': 'You have to enter an username',
        'menu.login.backToIndex': 'Back',
        'menu.loged.welcomeMsg': 'Welcome',
        'menu.button.colorListButton': 'Show color list',
        'menu.button.selectedColorButton': 'Selected color',
        'menu.selectColor.selectColorText':'Select a color: ',
        'menu.selectColor.selectedColorText':'Selected color: ',
        'menu.selectColor.selectColorButton':'Select',
        'menu.watchColor.selectedColorText':'Selected color: ',
        'menu.watchColor.selectColorButton':'Back to menu',
    };
    return loc
});
```

» Instalación de Redmine (Bitnami) e integración con Subversion.

» Introducción a Require.JS

» Conexión con mysql desde iSeries

Últimos Tutoriales del Autor

» Uso de Requirejs para modularizar una App creada con Emberjs

» Primeros pasos para conocer Emberjs

Últimas ofertas de empleo

2011-09-08
Comercial - Ventas - MADRID.

2011-09-03
Comercial - Ventas - VALENCIA.

2011-08-19
Comercial - Compras - ALICANTE.

2011-07-12
Otras Sin catalogar - MADRID.

2011-07-06
Otras Sin catalogar - LUGO.

4. Declarar nuestras dependencias y aplicar las traducciones a la aplicación

A la hora de crear nuestra app tenemos que declarar las dependencias a estos ficheros para poder utilizarlos más adelante:

```
    require(['jquery', 'cookies', "App", "ember", "i18n",
    "controllers/LoginController", "app/StateManager",
    "app/locHelpers","locEs", "locEn"],
    function($, cookies , App, Ember, i18n, LoginController, StateManager, locHelpers){
});
```

- locHelpers contendrá una serie de funciones que nos ayudarán en el manejo de selección de lenguaje, así como cargar por defecto las opciones. Como podéis ver definimos jquery y cookies, y es por que ese será nuestro método de persistir los datos, más adelante entenderéis porqué utilizamos cookies.

Cookies es un plugin de jquery, que nos permite crear, modificar, y leer cookies para nuestro navegador. Si quieres más informacion de este plugin de jquery visitar: Enlace información plugin jQuery para Cookies.

Acorde a la anterior aplicación debemos incluir en nuestro fichero config.js los siguientes paths:

```
/* Ficheros de textos */
'locEs': 'loc/es',
'locEn': 'loc/en'
```

Ahora vamos a ver como definimos nuestra app:

Si queremos que el plugin de cookies de jQuery admita recibir objetos en formato JSON es necesario incluir esta línea en nuestro código:

```
$.cookie.json = true
```

El resto:

```
var loc = null;
var options = locHelpers.loadOptions();
loc = loadLoc(options);

$.cookie('options', options);

Em.I18n.translations = loc;

root[app_name] = App = Ember.Application.create(App);
```

- Línea 1 : inicializamos una variable loc, que luego utilizaremos para las traducciones, línea 7
- Cargamos una serie de opciones que le ponemos a nuestra App, linea x.
- Si no tenemos creada una cookie cargara una serie de opciones por efecto.
- Linea x, definimos el valor de la variable loc, como esta establecido en las opciones.
- Si no esta establecido en la cookie, la funcion tratará de cargar un loc en funcion del lenguaje del navegador, más tarde le daremos al usuario la opción de cambiar el lenguaje manualmente.
- En la linea x vemos como crear cookie con el plugin de jQuery, funciona como un método establecedor(setter).

  En ella guardamos las opciones con el loc cargado.
- Es importante establecer el lenguaje con *Em.I18n.transaltions* antes de crear la App. (linea x)
- Y después creamos la App con el contenido que le asignabamos en app/app.js

  Enlace para ver app.js en github

5. Funciones establecedoras y calculadoras del lenguaje acorde al lenguaje del navegador

Ahora voy a mostraros las funciones de nuestro *locHelpers*, están autoexplicadas en los comentarios:

```
define(['cookies', 'locEs' ,'locEn'],function(){
    var locHelpers = {};
    /*
    Función que carga el idioma, que considere correcto, la aplicación
    ( basándose en factores como el lenguaje del navegador y los idiomas disponibles ),
    escribe una cookie en el navegador con la información sobre el lenguaje seleccionado.
    */
    locHelpers.loadLoc = function(options){
        var loc;
        var options = options;
        var locSelected = options.locSelected;
        var language;
        if(locSelected != null){
            language = locSelected;
        }else{
            language = locHelpers.guessLanguage();
        }
        switch(language){
            case 'es':
                options.locSelected = "es";
                loc = require('locEs');
            break;
            case 'es-ES':
                options.locSelected = "es";
                loc = require('locEs');
            break;
            case 'en':
                options.locSelected = "en";
                loc = require('locEn');
            break;
            case 'en-UK':
```

```
                options.locSelected = "en";
                loc = require('locEn');
            break;
            case 'en-US':
                options.locSelected = "en";
                loc = require('locEn');
            break;
        }
        return loc
    }
    /*
    Función interna de loadLoc que establece el lenguaje que considera correcto,
    tiene establecido un idioma por defecto (inglés)
    */
    locHelpers.setLanguage = function(userLang){
        var selectedLang;
        var supportedLangs = ['en-US', 'en-UK', 'en', 'es', 'es-ES'];
        if(supportedLangs.contains(userLang)){
            selectedLang = userLang;
        }else{
            selectedLang = "en";
        }
        return selectedLang
    }
    locHelpers.guessLanguage = function(){
        var lang = this.setLanguage(navigator.language);
        return lang;
    }
    /*
    Función que carga las opciones almacenadas en la cookie,
    devuelve un objeto con notación JSON con las opciones
    */
    locHelpers.loadOptions = function(){
        var options = $.cookie('options');
        if(!options){
            //En caso de que no exista la cookie, creamos el objeto options con las opcio
            options = {
                isLogged : false,
                username : null,
                colorSelected : 'red',
                loc : null
            }
            // Y guardamos la cookie con las opciones por defecto
            $.cookie('options', options)
        }
        return options
    }
    return locHelpers;
});
```

De esta manera lo definimos como un módulo.

6. Cargar nuestros textos en un template

A la hora de cargar estos textos lo haremos mediante los templates:
Handlebars nos provee de un helper, al utilizar i18n, para internacionalizar nuestra app. Se trata de *{{t nombre.variable}}*, esto cargará el valor de "nombre.variable" acorde al loc que le hemos establecido anteriormente.

Ejemplo de traducción en el login:

```
{{#if App.loginController.hasError}}
    <div class="error">
    {{#if App.loginController.errorEmpty}}
        {{t index.login.fail.empty tagName="p"}}
        {{else}}
        {{#if App.loginController.errorMismatch}}
        {{t index.login.fail.mismatch tagName="p"}}
        {{/if}}
    {{/if}}
    </div>
{{/if}}
<div class="inputGroup">
    {{t index.login.nameLabel tagName="label"}} {{view Ember.TextField valueBinding="Apg
</div>
<div class="inputGroup">
    {{t index.login.nameRepeatLabel tagName="label"}} {{view Ember.TextField valueBindi
</div>
<button {{action="" attemptlogin="" target="App.loginController" }}="" class="signInButt
    {{t index.login.loginButton}}
</button>

```

Línea 1 a 11, forma parte de un control de login que tenemos establecido, simplemente cargará un mensaje de error si el login es incorrecto.

Para ver como cargar un template ver este tutorial:

Enlace al tutorial

7. Añadir opción de cambiar lenguaje por el usuario

Ahora vamos a crear una serie de botones para que el usuario pueda seleccionar entre los lenguajes que dispone nuestra aplicación.

- Vamos a crear un *ArrayController* que nos provee Ember, para tener una lista de lenguajes en nuestra aplicación.

Este tendrá como dependencia *Localization*, módulo creado para crear objetos de esa clase.

- Modelo:

```
define(["require","ember"], function(require, Ember){
    return Ember.Object.extend({
        name : "",
        value: "",
        imgLink: ""
    });
});
```

- Controlador:

```
define(["require","ember", "jquery", "cookies",  "models/Localization"],
function(require, Ember, $, cookies){
    var Localization = require("models/Localization");

    var LocalizationsController = Ember.ArrayController.extend({
        localizationList: [],
        init : function (){
            this._super();
        }
        /* Lista de localizaciones soportadas por la aplicación */
        var es = Localization.create({
            name : 'Español',
            value : 'es',
            imgLink : 'img/es.png',
            isSelected : false
        });

        var en = Localization.create({
            name : 'English',
            value : 'en',
            imgLink : 'img/en.png',
            isSelected : false
        });
        /*
        Este if sirve para cargar una clase u otra para estilar el lenguaje selecci
        */
        if( App.stateManager.locSelected == 'es' ){
            es.isSelected = true;
        }else{
            en.isSelected = true;
        }
        this.localizationList.push(es);
        this.localizationList.push(en);
        },
        changeSelectedLoc : function(loc){
            /*
            Función encargada de cambiar el lenguaje en nuestro manejador de estad
            guardar el estado en la cookie y recargar la página.
            */
            if(loc.value != App.stateManager.locSelected){
                App.stateManager.set("locSelected", loc.value);
                App.stateManager.saveState();
                location.reload();
            }
        }
    });
    return LocalizationsController;
});
```

Nota: En el método changeSelectedLoc recargamos la página, ya que el template se carga antes de cambiar el lenguaje, y al cambiarlo NO se actualiza la vista.
De ahí que busquemos una forma de presistir el estado, y utilicemos las cookies.

- Creamos la plantilla

```
{{#each controllers.Localizations.localizationList}}
    <li {{bindattr="" class=":locItem isSelected:selected" }}="" {{action="" "change
        <img {{bindattr="" src="imgLink" }}="">
        {{name}}
    </li>
{{/each}}
```

Por cada localización creada en nuestro controlador, (each) crearemos un elemento a la lista de idiomas, al que le vamos a asignar una *action*. Esta llamará al método changeSelectedLoc de this.

this hace referencia al controlador de Localizations ya que se encuentra dentro de ese *{{#each }}* ver linea 1.
De este modo tendremos cargada en nuestra plantilla una lista de elementos (idiomas), en los que registraremos el evento de cambio de idioma. Si os fijáis en la función recibimos un objeto *loc*, y es que el this nos proporciona esa vinculación a cada objeto loc, que es en definitiva cada *Localization* que tenemos creado en el controlador.

8. Conclusiones

Con todo esto tendremos una aplicación simple internacionalizada, a partir de este ejemplo podréis internacionalizar otras aplicaciones.

Cualquier duda o sugerencia podéis comentarla.

Por favor, vota +1 o compártelo si te pareció interesante

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» **Registrate** y accede a esta y otras ventajas «