



Eusko Jaurlaritzaren Informatika Elkarte
Sociedad Informática del Gobierno Vasco

UDA - Utilidades de desarrollo de aplicaciones

Gestión idiomática

Fecha: 09/02/2012

Referencia:

EJIE S.A.
Mediterráneo, 14
Tel. 945 01 73 00*
Fax. 945 01 73 01
01010 Vitoria-Gasteiz
Posta-kutxatila / Apartado: 809
01080 Vitoria-Gasteiz
www.ejje.es



UDA – Utilidades de desarrollo de aplicaciones by [EJIE](#) is licensed under a [Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 Unported License](#).

Control de documentación

Título de documento: Gestión de properties e internacionalización (i18n)

Histórico de versiones

Código:	Versión:	Fecha:	Resumen de cambios:
	1.0.0	10/02/2012	Primera versión.
	2.0.0	08/08/2012	Gestión layout

Cambios producidos desde la última versión

Gestión layout

Control de difusión

Responsable: Ander Martínez

Aprobado por:

Firma:

Fecha:

Distribución:

Referencias de archivo

Autor:

Nombre archivo:

Localización:

Contenido





	Capítulo/sección	Página
1	Introducción	1
2	Gestión de idioma (antigua)	2
2.1	<i>CookieLocaleResolver</i>	2
2.2	<i>LocaleChangeInterceptor</i>	2
2.3	Problemas detectados	2
3	Gestión de idioma	4
3.1	Servidor	4
3.2	Cliente	6
4	Manual de migración	7
5	Nuevas funcionalidades	10

1 Introducción

El presente documento pretende explicar la manera en la que se gestionaba el idioma de la aplicación en UDA, que problemas se han detectado y la nueva gestión, haciendo hincapié en las mejoras conseguidas con el cambio.

Las aplicaciones desarrolladas con UDA gestionan el idioma a través de una cookie. Además dichas aplicaciones pueden ser multiidioma (i18n) por lo que existe la opción de cambiar el idioma en el que se muestra la aplicación, bien sea mediante una URL directa o a través del componente *rup.language*.

A lo largo del documento se explicarán los flujos definidos (servidor y cliente) a la hora de determinar el idioma utilizando los siguientes elementos:

-  Recepción del parámetro de cambio de idioma
-  Cookie de idioma
-  Idiomas disponibles en la aplicación
-  Idioma por defecto de la aplicación

2 Gestión de idioma (antigua)

El presente capítulo pretende explicar la manera en la que se gestionaba el idioma en UDA antes del cambio a la nueva versión.

2.1 *CookieLocaleResolver*

UDA utiliza una cookie para determinar el idioma de la aplicación. Para ello emplea la clase *CookieLocaleResolver* de Spring declarándola de la siguiente manera en el fichero *mvc-config.xml* del WAR:

```
<bean id="localeResolver"
      class="org.springframework.web.servlet.i18n.CookieLocaleResolver">
    <property name="cookieName" value="language" />
</bean>
```

Una vez se acceda a la aplicación se creará una cookie de sesión (de nombre 'language') que contendrá el idioma en el que se visualiza la aplicación (ej: es, eu, en, fr).

2.2 *LocaleChangeInterceptor*

UDA se basa en la clase de Spring *LocaleChangeInterceptor* para el cambio de idioma. Dicha clase se configura a nivel de WAR en el fichero *mvc-config.xml* de la siguiente manera:

```
<mvc:interceptors>
    <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />
</mvc:interceptors>
```

Dicha clase se declara como interceptora por lo que cada vez que una petición (a la aplicación) envíe el el parámetro declarado en el bean (si no se declara ninguno se usa por defecto 'locale'), entrará en funcionamiento dicha clase, realizando la gestión de idioma.

Un ejemplo de petición de cambio de idioma podría ser la siguiente:

<http://desarrollo.jakina.ejiedes.net:7001/x21aPilotoPatronesWar/?locale=eu>




2.3 Problemas detectados

A continuación se detallan algunos casos detectados en los que pueden generarse problemas debidos a esta configuración:

- Si se realiza una petición de cambio de idioma a uno que la aplicación no soporta (no se han definido sus ficheros idiomáticos)

<http://desarrollo.jakina.ejiedes.net:7001/x21aPilotoPatronesWar/?locale=zh>

- La aplicación falla al no encontrarse el fichero de literales (error en la JSP)
- Al no detectarse los literales se toma por defecto la Locale del servidor (no se sabe en qué idioma está configurado) y se carga dicho fichero.
- Los componentes de RUP al no encontrar los ficheros de literales fallan y no se muestran en pantalla (error de JavaScript).

-  La interrelación entre cambio de idioma, los idiomas disponibles en la aplicación y el idioma por defecto no existe. Se realiza petición de cambio de idioma a chino (zh) pero no se comprueba si es un idioma disponible en la aplicación y en función de si es o no establecer dicho idioma o el de por defecto.
-  Carga de la aplicación en el idioma en el que está configurado el servidor al no encontrarse el idioma por defecto.
-  ...

3 Gestión de idioma

A partir de la versión 2.0.0 de UDA los fallos detectados y comentados en el capítulo anterior han sido subsanados. Se ha mejorado y simplificado la manera de gestionar el idioma, añadiendo ciertas funcionalidades extra como puede ser el que en caso de no encontrar el literal a traducir se muestre la clave en lugar de producir un error.

3.1 Servidor

La nueva gestión de idioma (en la parte servidora) en UDA se realiza a través de la clase *MvcInterceptor* que se incluye en x38. Esta clase viene a sustituir a la clase *LocaleChangeInterceptor* (nativa de Spring) introduciendo más funcionalidad.

Al igual que en la versión anterior, se sigue empleando una cookie para mantener el idioma seleccionado durante la sesión del usuario. Para ello se hace uso de la clase *CookieLocaleResolver* de Spring.






La configuración de ambas clases se realiza a nivel de WAR en el fichero *mvc-config.xml* de la siguiente manera:

```
<bean id="localeResolver"
      class="org.springframework.web.servlet.i18n.CookieLocaleResolver">
    <property name="cookieName" value="language" />
</bean>


<bean id="mvcInterceptor" class="com.ejje.x38.control.MvcInterceptor" >
    <!-- <property name="paramName" value="locale" /> -->
    <property name="defaultLanguage" value="es" />
    <property name="defaultLayout" value="horizontal" />
    <property name="availableLangs" value="es,eu,en,fr" />
    <!-- <property name="portalCookie" value="r0leuskadiCookie" /> -->
</bean>

<mvc:interceptors>
    <ref bean="mvcInterceptor" />
</mvc:interceptors>
```

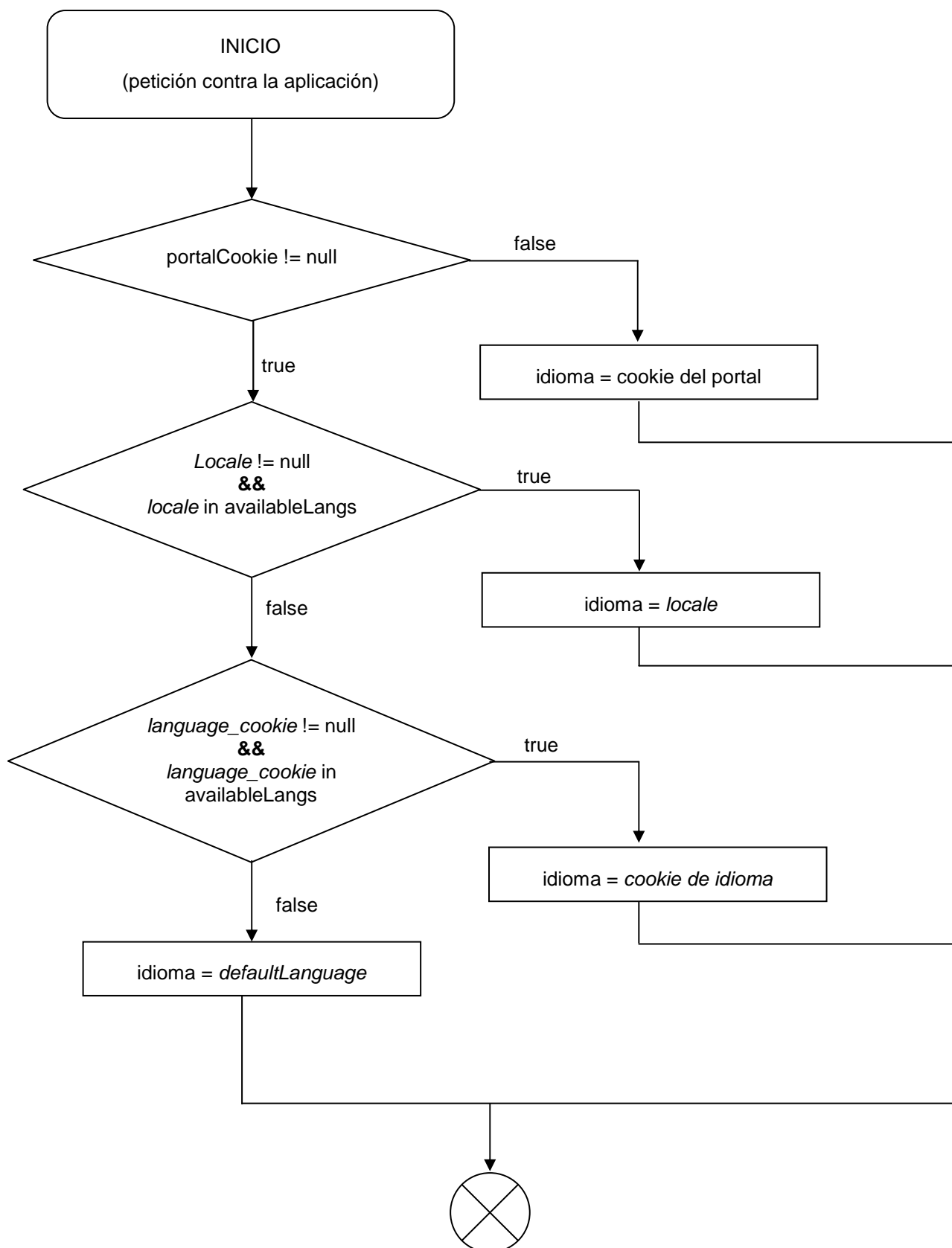
Los valores que recibe como propiedades son los siguientes:

-  **paramName** (opcional): Determina el nombre del parámetro empleado en las peticiones para el cambio de idioma. Por defecto es 'locale'
-  **defaultLanguage**: Idioma por defecto de la aplicación
-  **defaultLayout**: Layout por defecto de la aplicación
-  **availableLangs**: Idiomas permitidos por la aplicación (separados por comas)
-  **portalCookie**: Nombre de la cookie de idioma para aplicaciones integradas en portal.

A continuación se muestra mediante un diagrama de flujo la gestión que realiza *MvcInterceptor*:

-  **locale** → se refiere al parámetro de cambio de idioma que se envía en la petición
<http://desarrollo.jakina.ejjes.net:7001/x21aPilotoPatronesWar/?locale=eu>

✚ *language_cookie* → se refiere a la cookie de UDA para la gestión de idioma (configurada con el nombre *language* en mvc-config.xml)

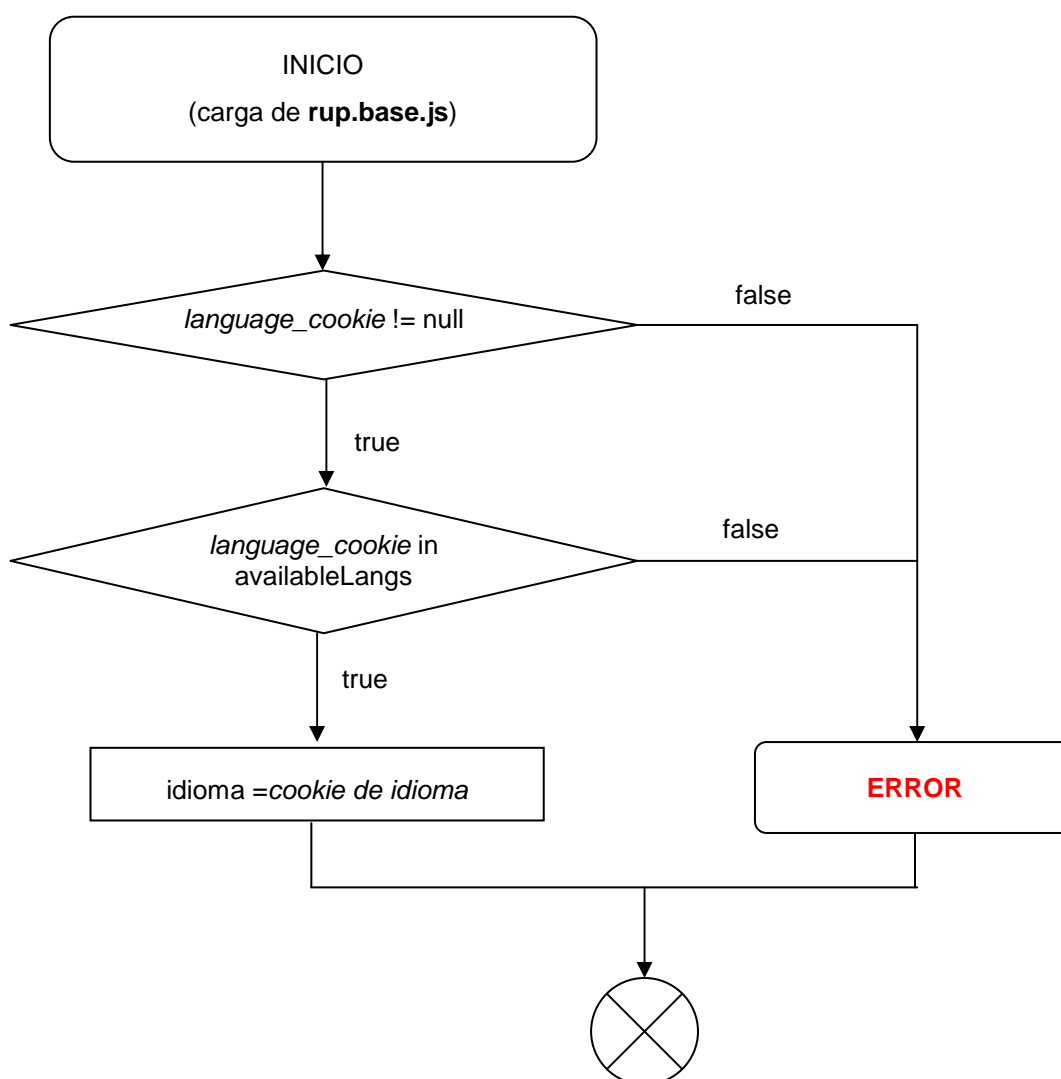


3.2 Cliente

Junto con los cambios en la parte servidora de cara a mejorar el sistema de gestión de idioma, se han realizado ciertos cambios en la parte cliente (componentes de RUP) manteniendo la compatibilidad hacia atrás.

La nueva gestión de idioma en RUP se realiza a través del fichero **rup.base-2.0.0.js** que determina el idioma a emplear en los componentes RUP mediante el diagrama que veremos a continuación.

🚦 *language_cookie* → se refiere a la cookie de UDA para la gestión de idioma (configurada con el nombre *language* en mvc-config.xml)



En caso de que se llegue al proceso “ERROR” se mostrará un mensaje por pantalla (alert) indicando la. No se realizará la petición de los ficheros idiomáticos ya que al no estar el idioma bien configurado provocará un error. A pesar de ello los componentes de RUP no fallarán aunque los literales de los mismos no se traducirán y se mostrará la clave utilizada para su resolución.

4 Manual de migración

El capítulo que nos ocupa pretende recopilar los cambios a realizar en una aplicación para utilizar la nueva gestión de idioma. A continuación se detalla proyecto a proyecto los cambios necesarios (en el siguiente capítulo se detallará las mejoras aportadas con dichos cambios):

➤ <xxxConfig>

- **xxx.properties** : Contiene variables para la gestión del idioma y del layout que ya no se utilizan ya que se declaran a nivel de WAR en su mvc-config.xml (podrían eliminarse).

```
x21aPilotoPatronesWar.default.language=es
x21aPilotoPatronesWar.default.layout=horizontal
```

➤ <xxxEAR>

- Sustituir la librería de x38 por la nueva versión **x38ShLibClasses-2.0.0.jar**

➤ <xxxEARClasses>

- Modificar en el fichero **service-config.xml** el bean *appMessageSource* añadiendo las líneas destacadas.

```
<bean id="appMessageSource"
      class="org.springframework.context.support.ReloadableResource
      BundleMessageSource">
  <property name="basename" value="x21a.i18n" />
  <property name="defaultEncoding" value="UTF-8" />
  <property name="useCodeAsDefaultMessage" value="true" />
  <property name="fallbackToSystemLocale" value="false" />
</bean>
```

➤ <xxxNombreWAR>

- Modificar en el fichero **mvc-config.xml** el bean *messageSource* añadiendo las líneas destacadas.

```
<bean id="messageSource"
      class="org.springframework.context.support.ReloadableResource
      BundleMessageSource">
  <property name="parentMessageSource" ref="appMessageSource" />
  <property name="basename" value="/WEB-
  INF/resources/x21aMantenimientos.i18n" />
  <property name="defaultEncoding" value="UTF-8" />
  <property name="useCodeAsDefaultMessage" value="true" />
  <property name="fallbackToSystemLocale" value="false" />
</bean>
```

- Modificar en el fichero **mvc-config.xml** para sustituir el antiguo interceptor por el nuevo. Teniendo en cuenta que ahora las propiedades del fichero xxx.properties para la gestión de idioma por defecto y layout se configuran mediante propiedades del interceptor.

```
[Viejo]
<!-- Configures Handler Interceptors -->
<mvc:interceptors>
  <!-- Changes the locale when a 'locale' request parameter
  is sent; e.g. /?locale=en -->
```

```

        <bean
            class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor" />
    </mvc:interceptors>
    <!-- Saves a locale change using a cookie -->
    <bean id="localeResolver"
        class="org.springframework.web.servlet.i18n.CookieLocaleResolver">
        <property name="cookieName">
            <value>language</value>
        </property>
    </bean>

```

```

[Nuevo]
<bean id="localeResolver"
    class="org.springframework.web.servlet.i18n.CookieLocaleResolver">
    <property name="cookieName" value="language" />
</bean>

<bean id="mvcInterceptor"
    class="com.ejie.x38.control.MvcInterceptor">
    <!-- <property name="paramName" value="locale" /> -->
    <property name="defaultLanguage" value="es" />
    <property name="defaultLayout" value="horizontal" />
    <property name="availableLangs" value="es,eu,en,fr" />
    <!-- <property name="portalCookie"
        value="r0leuskadiCookie" /> -->
</bean>

<mvc:interceptors>
    <ref bean="mvcInterceptor" />
</mvc:interceptors>

```

- Modificar en el fichero **mvc-config.xml** para añadir las nuevas clases de resolución de vista incluidas en x38.

```

[Viejo]
<bean id="viewResolver"
    class="org.springframework.web.servlet.view.UrlBasedViewResolver">
    <property name="viewClass"
        value="org.springframework.web.servlet.view.tiles2.TilesView" />
</bean>

```

```

[Nuevo]
<bean id="viewResolver"
    class="com.ejie.x38.control.view.UdaViewResolver">
    <property name="viewClass"
        value="com.ejie.x38.control.view.UdaTilesView" />
    <property name="exposedContextBeanNames" >
        <list>
            <value>localeResolver</value>
            <value>mvcInterceptor</value>
        </list>
    </property>
</bean>

```

```
</property>
</bean>
```

- Modificar en el fichero **base-includes.jsp** para cambiar las variables utilizadas en la configuración del idioma para RUP:

```
[Viejo]
<script type="text/javascript">
    APP_RESOURCES = 'x21a',
    CTX_PATH = '<%= request.getContextPath() %>/',
    RUP = '${staticsUrl}/rup',
    STATICS = '${staticsUrl}',
    DEFAULT_LANGUAGE = "${defaultLanguage}",
    LAYOUT = "${defaultLayout}",
    WAR_NAME = "x21aPilotoPatrones",
    AVAILABLE_LANGS = "es, eu, en, fr";
</script>

[Nuevo]
<script type="text/javascript">
    APP_RESOURCES = 'x21a',
    CTX_PATH = '<%= request.getContextPath() %>/',
    STATICS = '${staticsUrl}',
    RUP = '${staticsUrl}/rup',
    WAR_NAME = "x21aPilotoPatrones",
    //model
    LAYOUT = "${empty defaultLayout ?
mvcInterceptor.defaultLayout : defaultLayout}",
    //mvc-config.xml
    LOCALE_COOKIE_NAME = "${localeResolver.cookieName}",
    LOCALE_PARAM_NAME = "${mvcInterceptor.paramName}",
    AVAILABLE_LANGS = "${mvcInterceptor.availableLangs}",
</script>
```

- DEFAULT_LANGUAGE (deprecated): Ahora se carga desde el mvc-config.xml
- LOCALE_COOKIE_NAME: Nombre de la cookie (definida en mvc-config.xml)
- LOCALE_PARAM_NAME: Nombre del parámetro de cambio de idioma (definida en mvc-config.xml)
- AVAILABLE_LANGS: Posibles lenguajes de la aplicación Nombre (definidos en mvc-config.xml)

➤ <xxxStatics>

- Modificar en el fichero **_layoutLoader.js** (existe uno por cada WAR) para cambiar la invocación del componente rup.language [en caso de que se utilice] utilizando la variable definida.

```
[Viejo]
$("#x21aPilotoPatronesWar_language").rup_language({languages:
["es", "eu", "en", "fr"]});
```

```
[Nuevo]
$("#x21aMantenimientosWar_language").rup_language({languages:
$.rup.AVAILABLE_LANGS_ARRAY});
```

5 Nuevas funcionalidades

La nueva gestión idiomática de UDA, además de resolver los problemas indicados aportar robustez y nuevas funcionalidades que se comentan a continuación:

- ✚ Inclusión de la clase 'MvcInterceptor' (x38) en el mvc-config.xml
 - Centralizar la configuración del layout, idioma por defecto e idiomas disponibles en la aplicación a nivel de WAR evitando la declaración de dichas variables en el properties de aplicación (dependientes del entorno).
 - Facilitar el acceso a dichas variables mediante EL (\${...}) desde la parte cliente (JavaScript).
 - Evitar errores ya que en caso de estar mal configurada alguna de las propiedades (nombre de la propiedad incorrecta o no declarada) se produce una excepción, imposibilitando el despliegue avisando en el fichero de incidencias la causa del problema (detalla la propiedad(es) que ha(n) fallado):


```
Caused By: java.lang.IllegalStateException: No se ha definido correctamente el bean 'MvcInterceptor' en el fichero mvc-config.xml del proyecto <x21aPilotoPatronesWar>. Revisar propiedad(es):[defaultLanguage]
```
 - Simplificar la codificación de los controladores evitando código duplicado en todos ellos. Los métodos encargados de devolver una vista (acceder a una página) ya no requieren de dejar las variables de idioma y layout por defecto en el modelo (en caso de necesidad se permite sobrescribir el layout por defecto incluyéndolo en el modelo:


```
model.addAttribute("layout", "mixto");
```

[Viejo]

```
//Autocomplete
@RequestMapping(value = "autocomplete", method = RequestMethod.GET)
public ModelAndView getAutocomplete(Model model) {
    model.addAttribute("defaultLanguage",
        appConfiguration.get("x21aPilotoPatronesWar.default.language"));
    model.addAttribute("defaultLayout",
        appConfiguration.get("x21aPilotoPatronesWar.default.layout"));
    return new ModelAndView("autocomplete", "model", model);
}
```

[Nuevo]

```
@RequestMapping(value = "autocomplete", method = RequestMethod.GET)
public String getAutocomplete(Model model) {
    return "autocomplete";
}
```
- ✚ Cambio de clases para la resolución de las vistas (UdaViewResolver y UdaTiesView) en el mvc-config.xml
 - Acceso mediante EL (\${...}) a los beans cargados en el contexto de Spring requeridos por RUP (localeResolver y localeInterceptor). Desde el fichero *base-includes.jsp* se podrá acceder a las variables AVAILABLE_LANGS, LOCALE_COOKIE_NAME y LOCALE_PARAM_NAME que gestiona el interceptor (automáticamente) y necesarias para los componentes de RUP. Por compatibilidad hacia atrás en caso de no declararse se tomarán valores por defecto (cookie='language' y param='locale').

```
<bean id="viewResolver" class="com.ejje.x38.view.UdaViewResolver">
  <property name="viewClass" value="com.ejje.x38.view.UdaTilesView" />
  <property name="exposeContextBeansAsAttributes" value="true" />
</bean>
```

Mediante la propiedad **exposeContextBeansAsAttributes** se exponen todos los beans del contexto. Para acceder bastará con indicar el nombre del bean y la propiedad deseada:

```
${localeResolver.cookieName}
```

En caso de querer exponer solo algunos de los beans existe la propiedad **exposedContextBeanNames**.

```
<property name="exposedContextBeanNames">
  <list>
    <value>beanA</value>
    <value>beanB</value>
  </list>
</property>
```

- ✚ Cambio en la declaración de `appMessageSource` (EARClasses – `service-config.xml`) y `messageSource` (WAR – `mvc-config.xml`).

- Evitar el uso de la Locale del servidor de aplicaciones en caso de no encontrar el fichero de recursos (i18n).

```
<property name="fallbackToSystemLocale" value="false" />
```

- Evitar errores en el acceso a la aplicación ya que en caso de no encontrar una propiedad se mostrará su clave en lugar de generar una excepción. Ahora se mostrará la página a la que se estaba accediendo en lugar de redirigirse a la página de error.

```
<property name="useCodeAsDefaultMessage" value="true" />
```

Ejemplo de traza de error:

```
org.springframework.web.servlet.tags.MessageTag ~~ ERROR
~~ No message found under code 'usuario' for locale 'zh'.
~~ javax.servlet.jsp.JspTagException: No message found
under code 'usuario' for locale 'zh'.
```

NOTA: Se deben modificar ambos ficheros ya que en caso de que no encontrarse un literal pedido desde el WAR se accede a los recursos del EARClasses.

La versión 2.0.0 de los componentes de RUP gestiona los literales de la misma manera que la parte servidora. En caso de existir algún problema en la carga de los ficheros de recursos (i18n), los componentes y la aplicación seguirán funcionando correctamente pero en pantalla se mostrará como literal la clave con la que se accede a los properties.

Los componentes de RUP acceden a las variables definidas en el fichero **base-includes.jsp** (propio de la aplicación) para configurar las propiedades relativas a la gestión del idioma definidas en el interceptor en el fichero del WAR (`mvc-config.xml`). Estas variables se comentan en el apartado 3.1 del presente documento.