

Joc de caràcters en aplicacions Canigó 3

A qui va dirigit

Aquest how-to va dirigit a tots aquells que vulguin desenvolupar aplicacions Canigó 3.

Versió de Canigó

Els passos descrits en aquest document apliquen a aplicacions Canigó 3 i que utilitzin la versió 1.1.0 o posterior del mòdul canigo.web.jsf, i que per tant, utilitzin JSF 2.

Introducció

En aquest document es presenten algunes recomanacions, en relació amb la utilització del joc de caràcters UTF-8 en els diferents elements de l'arquitectura per aplicacions web desenvolupades amb Canigó 3.

Entorn de treball

L'entorn de desenvolupament de Canigó 3 està basat en Eclipse. Des de l'opció de menú de preferències de l'entorn és possible establir per a tots els fitxers de text amb que treballa l'aplicació el joc de caràcters UTF-8. Es recomana fer-ho per els següents tipus de fitxers:

- Java Properties File: *.properties (Configuració i l18n)
- XML: *.xml (Configuració)
- HTML: *.htm, *.html, *.jsf, *.xhtml (Presentació)

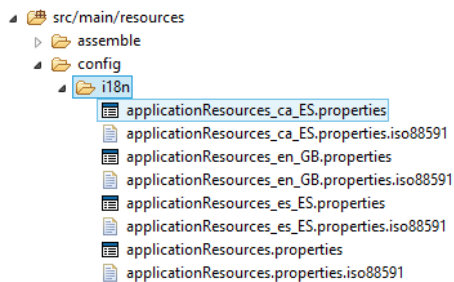
Abans de fer el canvi de joc de caràcters, s'ha de fer la conversió del contingut dels fitxers. Tots els enumerats anteriorment són susceptibles de contenir caràcters que amb el canvi de codificació poden visualitzar-se posteriorment de forma incorrecte:

Ex: Monitorització (ISO-8859-1) → Monitorizaci❖ (UTF-8)

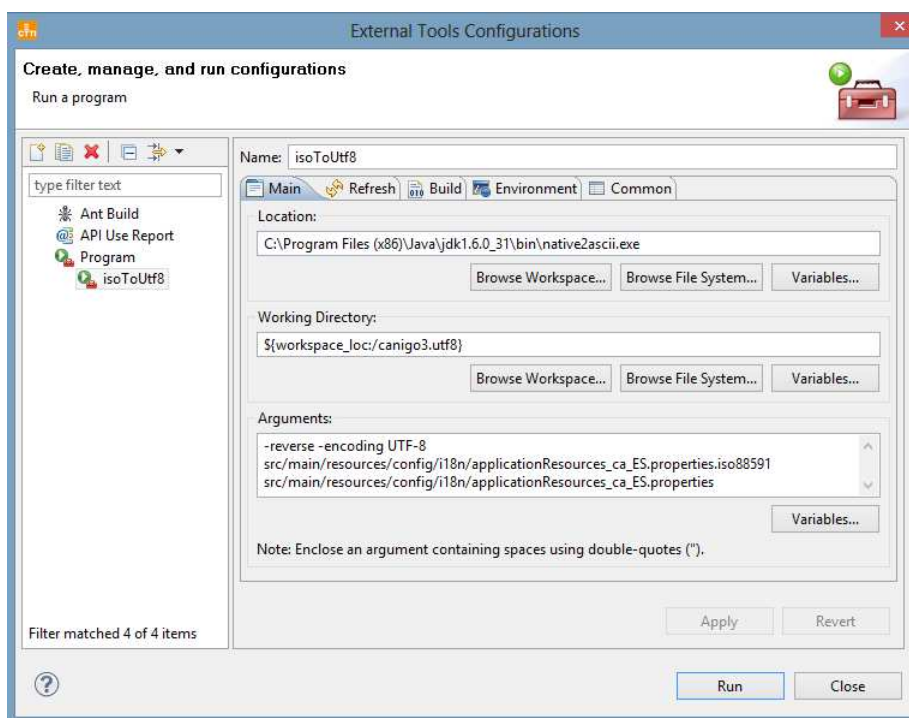
Tot i això, es farà únicament per els fitxers d'internacionalització (l18n) dins la carpeta src/main/resources/config/i18n. Per la resta, si s'ha de codificar de nou algun caràcter es pot fer de forma manual. A priori, no haurien de contenir caràcters problemàtics (accentuats, el·les geminades, etc.).

Per fer el canvi a UTF-8, primer de tot s'ha de fer la conversió del contingut. Per això, el que es pot fer es renombrar els fitxers existents afegint el sufix iso88591:

Joc de caràcters en aplicacions Canigó 3



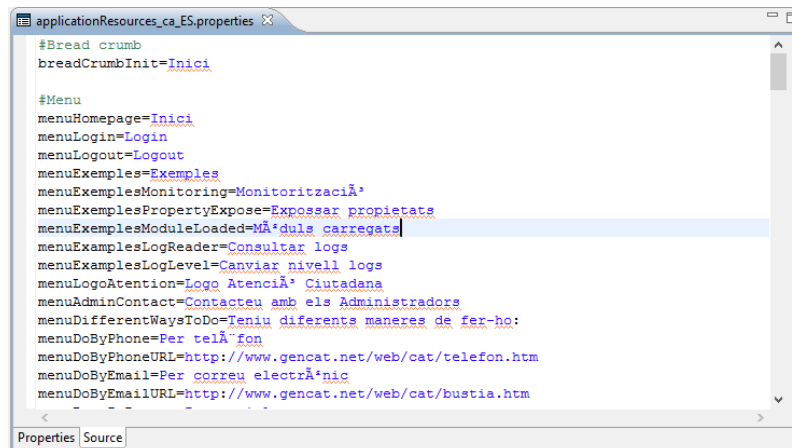
I executar la tasca “native2ascii” que proporciona Java. La configuració ha realitzar és la següent:



Un cop executada s’haurà convertit el contingut del fitxer applicationResources_ca_ES.properties.iso88591 codificat en ISO-8859-1 a UTF-8 en el fitxer applicationResources_ca_ES.properties. Aquesta operació s’ha de conversió s’ha de fer per a cadascun dels fitxers d’internacionalització dins la carpeta src/main/resources/config/i18n. Es recomana crear un fitxer de recursos per defecte applicationResources.properties còpia del applicationResources_ca_ES.properties ja convertit.

Degut a que s’ha fet el canvi de codificació del contingut però no s’ha especificat a l’Eclipse que ha d’interpretar-ho també com a UTF-8 enlloc del sistema de codificació natiu del sistema operatiu (Ex. Cp1252 a Windows), la visualització es farà de forma incorrecte:

Joc de caràcters en aplicacions Canigó 3



```
applicationResources_ca_ES.properties
#Bread crumb
breadCrumbInit=Inici

#Menu
menuHomepage=Inici
menuLogin=Login
menuLogout=Logout
menuExemples=Exemples
menuExemplesMonitoring=MonitoritzaciÃ
menuExemplesPropertyExpose=Expossar propietats
menuExemplesModuleLoaded=MÃduls carregats
menuExemplesLogReader=Consultar logs
menuExemplesLogLevel=Canviar nivell logs
menuLogoAttention=Logo AtenciÃ Ciutadana
menuAdminContact=Contacteu amb els Administradors
menuDifferentWaysToDo=Teniu diferents maneres de fer-ho:
menuDoByPhone=Per telÃfon
menuDoByPhoneURL=http://www.gencat.net/web/cat/telefon.htm
menuDoByEmail=Per correu electrÃnic
menuDoByEmailURL=http://www.gencat.net/web/cat/bustia.htm
```

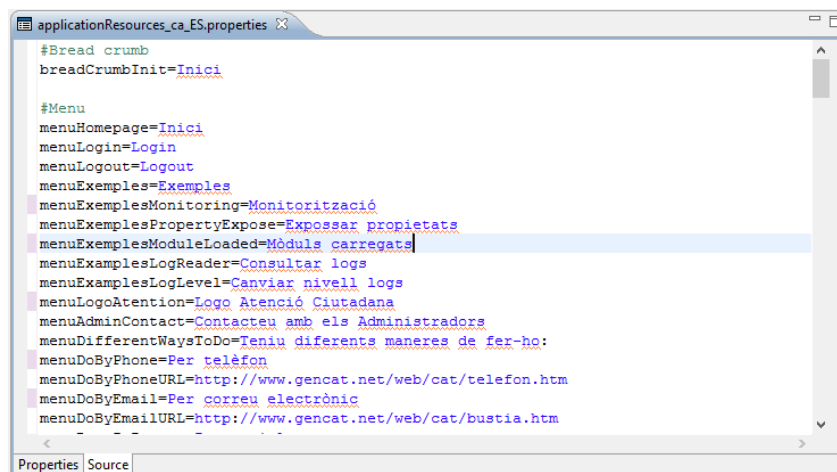
Per a solucionar-ho, cal especificar a l'Eclipse que els fitxers de propietats (Java Properties File) estan codificats amb UTF-8. També per d'altres tipologies de fitxers susceptibles de tenir caràcters accentuats i d'altres caràcters que puguin presentar problemes amb el canvi de codificació:

Window → Preferences → General → Content types → Text

- Java Properties File → *.properties → Default Encoding: **UTF-8** [Update]
- XML → *.xml → Default Encoding: **UTF-8** [Update]
- HTML → *.htm, *.html, *.jsf, *.xhtml → Default Encoding: **UTF-8** [Update]

No farem la conversió amb “native2ascii” per a tots ells degut a que no haurien, a priori, de contenir cap caràcter problemàtic. Tot i així, durant el desenvolupament es podrien introduir-ne i d'aquesta manera evitem possibles problemes de codificació/descodificació.

Un cop aplicat el canvi de “Content types” dins l'Eclipse, el fitxer applicationResources_ca_ES.properties ja hauria de visualitzar-se correctament:

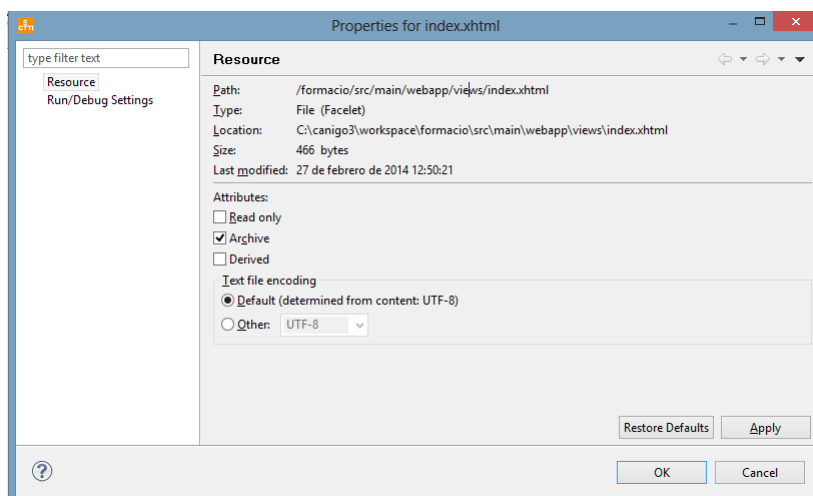


```
applicationResources_ca_ES.properties
#Bread crumb
breadCrumbInit=Inici

#Menu
menuHomepage=Inici
menuLogin=Login
menuLogout=Logout
menuExemples=Exemples
menuExemplesMonitoring=Monitorització
menuExemplesPropertyExpose=Expossar propietats
menuExemplesModuleLoaded=Mòduls carregats
menuExemplesLogReader=Consultar logs
menuExemplesLogLevel=Canviar nivell logs
menuLogoAttention=Logo Atenció Ciutadana
menuAdminContact=Contacteu amb els Administradors
menuDifferentWaysToDo=Teniu diferents maneres de fer-ho:
menuDoByPhone=Per telèfon
menuDoByPhoneURL=http://www.gencat.net/web/cat/telefon.htm
menuDoByEmail=Per correu electrònic
menuDoByEmailURL=http://www.gencat.net/web/cat/bustia.htm
```

També podem validar que, per exemple, els fitxers *.xhtml (Facelet) han canviat de codificació. Per fer aquesta validació fer botó dret sobre un fitxer .xhtml com /src/main/webapp/views/index.xhtml:

Joc de caràcters en aplicacions Canigó 3



Servidor d'aplicacions

Ja tenim configurat l'entorn de treball preparat per a treballar amb el joc de caràcters UTF-8. El següent pas és fer les configuracions necessàries a l'aplicació (codi Java) i al servidor d'aplicacions per a que en temps d'execució s'interpretin correctament aquests fitxers font.

SERVEI DE PRESENTACIÓ (JSF)

Missatges

Si executem una aplicació plantilla Canigó 3 únicament amb el canvi realitzat a nivell d'entorn de desenvolupament i compilada posteriorment, la plana de benvinguda es visualitzarà de la següent manera:



JSF, com a tecnologia utilitzada per a la presentació, treballa per defecte amb ISO-8859-1. Per tal de forçar el canvi a UTF-8 en la presentació de missatges a les planes *.jsf i *.xml a través del bundle "msg", cal fer el següent canvi en el fitxer de configuració src/main/webapp/WEB-INF/faces-config.xml:

faces-config.xml (ISO-8859-1)

```
<resource-bundle>
  <base-name>config.i18n.applicationResources</base-name>
  <var>msg</var>
</resource-bundle>
```

Joc de caràcters en aplicacions Canigó 3

faces-config.xml (UTF-8)

```
<resource-bundle>
  <base-name> cat.gencat.canigo3.utf8.il8n.ApplicationResources</base-name>
  <var>msg</var>
</resource-bundle>
```

On la classe `cat.gencat.canigo3.utf8.il8n.ApplicationResources` és una implementació de la classe abstracta `java.util.ResourceBundle`.

ApplicationResources.java

```
package cat.gencat.canigo3.utf8.il8n;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.util.Enumeration;
import java.util.Locale;
import java.util.PropertyResourceBundle;
import java.util.ResourceBundle;

import javax.faces.context.FacesContext;

public class ApplicationResources extends ResourceBundle {

    protected static final String BUNDLE_NAME = "config.il8n.applicationResources";
    protected static final String BUNDLE_EXTENSION = "properties";
    protected static final String CHARSET = "UTF-8";
    protected static final Control UTF8_CONTROL = new UTF8Control();

    public ApplicationResources() {
        setParent(ResourceBundle.getBundle(BUNDLE_NAME,
            FacesContext.getCurrentInstance().getViewRoot().getLocale(), UTF8_CONTROL));
    }

    @Override
    protected Object handleGetObject(String key) {
        return parent.getObject(key);
    }

    @Override
    public Enumeration<String> getKeys() {
        return parent.getKeys();
    }

    protected static class UTF8Control extends Control {
        public ResourceBundle newBundle
            (String baseName, Locale locale, String format, ClassLoader loader, boolean
reload)
            throws IllegalAccessException, InstantiationException, IOException
        {
            // The below code is copied from default Control#newBundle() implementation.
            // Only the PropertyResourceBundle line is changed to read the file as UTF-
8.

            String bundleName = toBundleName(baseName, locale);
            String resourceName = toResourceName(bundleName, BUNDLE_EXTENSION);
            ResourceBundle bundle = null;
            InputStream stream = null;
            if (reload) {
                URL url = loader.getResource(resourceName);
                if (url != null) {
                    URLConnection connection = url.openConnection();
                    if (connection != null) {
                        connection.setUseCaches(false);
                        stream = connection.getInputStream();
                    }
                }
            }
        }
    }
}
```

Joc de caràcters en aplicacions Canigó 3

```
    }  
  }  
  } else {  
    stream = loader.getResourceAsStream(resourceName);  
  }  
  if (stream != null) {  
    try {  
      bundle = new PropertyResourceBundle(new InputStreamReader(stream,  
CHARSET));  
    } finally {  
      stream.close();  
    }  
  }  
  return bundle;  
}  
}
```

Aplicats aquests canvis a l'aplicació ja s'haurien de visualitzar correctament els missatges a les planes JSF:



Codificació pàgines

En totes les planes JSF (*.jsf) i Facelets (*.xhtml) s'ha de substituir la capçalera:

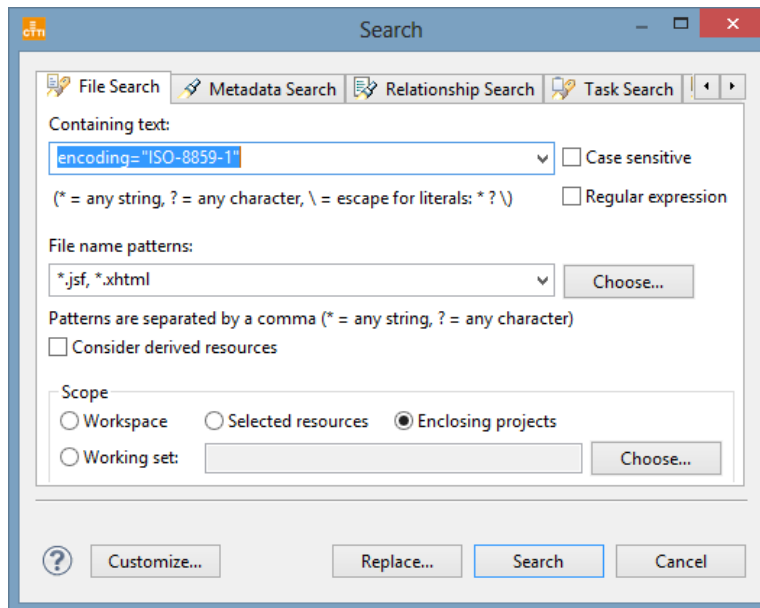
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
```

Per:

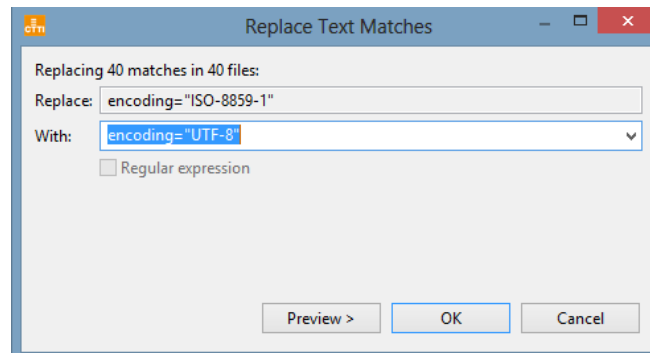
```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

Aquest canvi es pot fàcilment amb l'opció Search → File dins l'Eclipse:

Joc de caràcters en aplicacions Canigó 3



I executant l'acció de Replace:



També en la plana `src/main/webapp/views/layouts/template.jsf` substituir el charset:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

Per:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Codificació peticions HTTP

Per tal que en la descodificació de la petició HTTP s'utilitzi un encoding determinat, Spring ofereix el filtre `org.springframework.web.filter.CharacterEncodingFilter`. Per la utilització de UTF-8 configurar el filtre al descriptor de l'aplicació web `src/main/webapp/WEB-INF/web.xml` com s'especifica a continuació:

web.xml

```
<filter>  
<filter-name>SetCharacterEncodingFilter</filter-name>
```

Joc de caràcters en aplicacions Canigó 3

```
<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF8</param-value>
  </init-param>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>>true</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>SetCharacterEncodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Nota: es recomana que el filtre SetCharacterEncodingFilter estigui com a primer filter-mapping

D'aquesta manera els inputs dels formularis es descodificaran amb UTF-8. Això pel que fa l'ús del mètode HTTP POST. Per el mètode HTTP GET, si es vol que els paràmetres enviats a la URL es descodifiquin amb UTF-8, depenent del servidor d'aplicacions és possible que s'hagi de fer una configuració especial. Per el cas de Tomcat, s'ha de configurar el paràmetre URIEncoding de la següent manera:

server.xml

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000"
  redirectPort="8443" URIEncoding="UTF-8" />
```

Per el servidor d'aplicacions Weblogic no cal fer cap configuració ja que per defecte utilitza UTF-8.

SERVEI D'INTERNACIONALITZACIÓ (I18N)

El paquet d'internacionalització dins el core de Canigó, de la mateixa manera que amb JSF, cal adaptar-lo per a que treballi amb UTF-8. Per defecte, l'API java.util.Properties de Java utilitzada internament en la internacionalització de missatges empra ISO-8859-1, d'aquí la necessitat d'aquest canvi. Per a fer-ho, cal sobrescriure la definició del bean de Spring amb id "messageSource" configurada al core de Canigó:

app-custom-i18n.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

  <bean id="messageSource"
    class="cat.gencat.canigo3.utf8.i18n.CustomReloadableResourceBundleMessageSource">
    <description>
      Message Resource que s'encarrega de carregar i proporcionar
      internacionalització
      al framework.

      Per recuperar el locale necessari per a recuperar els literals:
      - Validem si al threadlocal de la petició el trobem
      informat
      - Validem si te un locale per defecte, veure: defaultLocale
      - Locale de la instància de la VM

    </description>
    <property name="basenames">
      <list>
        <value>cat.gencat.ctti.canigo.arch.config.i18n.</value>
```


Joc de caràcters en aplicacions Canigó 3

```
        <value>config.i18n.</value>
      </list>
    </property>
    <property name="defaultLocale" value="\${application.defaultLanguage}"/>
    <property name="fileEncodings" value="UTF-8" />
    <property name="defaultEncoding" value="UTF-8" />
  </bean>
</beans>
```

Creem aquest fitxer i l'ubiquem en el path `src/main/resources/spring` dins l'aplicació. Aquest nou bean "messageSource" instància la classe `cat.gencat.canigo3.utf8.i18n.CustomReloadableResourceBundleMessageSource` que estén `org.springframework.context.support.ReloadableResourceBundleMessageSource` de Spring, la qual permet informar les propietats `fileEncodings` i `defaultEncoding`. El codi d'aquesta classe pròpia és la següent:

CustomReloadableResourceBundleMessageSource.java

```
package cat.gencat.canigo3.utf8.i18n;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.ResourceBundle;
import java.util.Set;
import java.util.TreeSet;

import org.springframework.beans.factory.InitializingBean;
import org.springframework.core.io.Resource;
import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
import org.springframework.core.io.support.ResourcePatternResolver;

import cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource;
import cat.gencat.ctti.canigo.arch.core.threadlocal.ThreadLocalHolder;
import cat.gencat.ctti.canigo.arch.core.utils.CoreUtils;

/**
 * Implementation that provides access to resource bundles without
 * need to specify the current locale
 *
 * @author cscanigo
 *
 */
public class CustomReloadableResourceBundleMessageSource extends
org.springframework.context.support.ReloadableResourceBundleMessageSource implements
InitializingBean, I18nResourceBundleMessageSource {

    /** default application locale */
    private Locale defaultLocale = null;

    /** (non-Javadoc)
     * @see
     cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource#getMessage(java.la
ng.String, java.lang.Object[], java.lang.String)
     */
    public final String getMessage(String code, Object[] args, String defaultMessage)
    {
        return getMessage(code, args, defaultMessage, getCurrentLocale());
    }

    /** (non-Javadoc)
     * @see
     cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource#getMessage(java.la
ng.String, java.lang.Object[])

```

Joc de caràcters en aplicacions Canigó 3

```
    */
    public final String getMessage(String code, Object[] args) {
        return getMessage(code, args, null, getCurrentLocale());
    }

    /* (non-Javadoc)
     * @see
cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource#getMessage(java.la
ng.String)
     */
    public final String getMessage(String code) {
        return getMessage(code, null, null, getCurrentLocale());
    }

    /* (non-Javadoc)
     * @see
cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource#getCurrentLocale()
     */
    public Locale getCurrentLocale() {
        Locale locale = ThreadLocalHolder.get(LOCALE_PROPERTY_NAME,
Locale.class);

        if (locale == null) {
            locale = (defaultLocale==null)?Locale.getDefault():defaultLocale;
        }

        return locale;
    }

    /*
     * (non-Javadoc)
     * @see
cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource#setDefaultLocale(j
ava.lang.String)
     */
    public void setDefaultLocale(String defaultLocale) {
        this.defaultLocale = CoreUtils.getLocale(defaultLocale);
    }

    /*
     * (non-Javadoc)
     * @see org.springframework.beans.factory.InitializingBean#afterPropertiesSet()
     */
    public void afterPropertiesSet() throws Exception {
        logger.info("Internationalization module loaded...");
    }

    /*
     * (non-Javadoc)
     * @see
cat.gencat.ctti.canigo.arch.core.i18n.I18nResourceBundleMessageSource#getResourceBundle(
java.lang.String)
     */
    public ResourceBundle getResourceBundle(String basename) {
        return null;
        //return this.getResourceBundle(basename, getCurrentLocale());
    }

    /**
     * Return filename without extension and language suffix
     *
     * @param filename
     * @return String
     */
    private String getCleanFileName(final String filename){
        int index = filename.lastIndexOf('.');
        String fileNameWOEx = (index>0&& index <= filename.length() - 2) ?
filename.substring(0, index): filename;
        return fileNameWOEx.split("_")[0];
    }
}
```

Joc de caràcters en aplicacions Canigó 3

```
}

/**
 * Return a resource's list contained in the specified folder
 *
 * @param basename
 * @return
 */
private List<String> getNestedResources(final String basename){
    StringBuffer sb = new StringBuffer();
    sb.append("classpath:");
    sb.append(basename.replaceAll("\\.", "/"));
    sb.append("**.properties");

    Set<String> resourceList = new TreeSet<String>();
    ResourcePatternResolver resolver = new
PathMatchingResourcePatternResolver();
    Resource[] resources;
    try {
        resources = resolver.getResources(sb.toString());
        if(resources != null){
            for(Resource resource : resources){
                try{
                    String resourceName = basename +
getCleanFileName(resource.getFilename());
                    resourceList.add(resourceName);
                    logger.info("Resolved resource: " +
resourceName);
                }catch(Exception e){
                    logger.error("There was a problem during
basename resolution", e);
                }
            }
        }
    } catch (IOException e1) {
        logger.error("There was a problem during basename resolution",
e1);
    }

    return new ArrayList<String>(resourceList);
}

/**
 * Set an array of basenames, each following {@link java.util.ResourceBundle}
 * conventions: essentially, a fully-qualified classpath location. If it
 * doesn't contain a package qualifier (such as org.mypackage),
 * it will be resolved from the classpath root.
 *
 * Override parent behavior to allow folders
 * @see #setBasename
 * @see java.util.ResourceBundle#getBundle(String)
 */
@Override
public void setBasenames(String[] basenames) {

    List<String> basenameList = new ArrayList<String>();
    boolean isFolder = false;
    if (basenames != null){
        for(String basename : basenames){
            try{
                isFolder = basename.endsWith(".");
                if(!isFolder){
                    basenameList.add(basename);
                }else{
                    basenameList.addAll(
getNestedResources(basename));
                }
            }
        }
    }
}
```

Joc de caràcters en aplicacions Canigó 3

```
    }  
    } catch (Exception e) {  
        logger.error("There was a problem during basename  
resolution", e);  
    }  
    }  
    String ia[] = new String[basenameList.size()];  
    ia = basenameList.toArray(ia);  
    basenames = ia;  
    }  
    super.setBasenames(basenames);  
    }  
    @Override  
    protected PropertiesHolder refreshProperties(String filename,  
        PropertiesHolder propHolder) {  
        StringBuffer sb = new StringBuffer();  
        sb.append("classpath:");  
        sb.append(filename.replaceAll("\\.", "/"));  
        return super.refreshProperties(sb.toString(), propHolder);  
    }  
}
```

L'obtenció de missatges internacionalitzats es farà de la mateixa manera que amb el bean `messageSource` original que incorpora el core de Canigó gràcies a que tots dos implementen la mateixa interfície

`cat.gencat.ctti.canigo.arch.core.il18n.I18nResourceBundleMessageSource`:

```
@Autowired  
I18nResourceBundleMessageSource messageSource;  
  
public void test() throws Exception {  
    String message = messageSource.getMessage("test.message");  
}
```

SERVEI DE CONFIGURACIÓ

Tot i que les propietats definides en la configuració de serveis (URLs, datasources, smtp, etc.) no haurien de contenir caràcters susceptibles de presentar problemes de codificació (vocals accentuades, el·les geminades, etc.), és possible sobre escriure els beans de Spring que manegen aquesta configuració. Per fer-ho crear el fitxer `src/main/resources/spring/app-custom-props.xml` amb el següent contingut:

app-custom-props.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://www.springframework.org/schema/beans  
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">  
    <bean id="propertyPlaceholderConfigurer"  
    <description>  
entorn  
        Placeholder de les propietats de l'aplicació Canigó depenents de  
        - *.nomProperty = value (property global, per a tots els entorns)
```

Joc de caràcters en aplicacions Canigó 3

```
        - loc.nomProperty = value (property especifica d'entorn,
sobrescriu la global)
    </description>
    <property name="locations">
        <list>
            <value>classpath:/config/props/*.properties</value>
        </list>
    </property>
    <property name="fileEncoding" value="UTF-8" />
</bean>

<bean id="propertiesConfiguration"
class="cat.gencat.ctti.canigo.arch.core.config.PropertiesConfiguration">
    <description>
        Classe que permet a les aplicacions accedir a les propietats de
configuració
        carregades pel propertyPlaceholderConfigurer
    </description>
    <constructor-arg index="0" ref="propertyPlaceholderConfigurer"/>
</bean>
</beans>
```

Amb aquesta configuració sobreescrivim els beans `propertiesConfiguration` i `propertyPlaceholderConfigurer` que es defineixen en el core de Canigó establint la propietat `fileEncoding="UTF-8"` en aquest últim.

Servidor de Base de dades

El charset que es configurarà en els servidors Oracle corporatius és **AL32UTF8**. Es recomana que en entorns de desenvolupament s'utilitzi aquest mateix joc de caràcters. Per defecte, Oracle utilitza aquest charset en la creació d'una instància de BBDD. Es pot consultar de la següent manera:

```
SELECT value$ FROM sys.props$ WHERE name = 'NLS_CHARACTERSET' ;
```

Aquesta consulta ha de retornar el valor AL32UTF8.

Com a conclusió, si es segueixen les indicacions d'aquest HowTo les diferents peces que formen part de l'arquitectura de Canigó utilitzaran UTF-8 sense produir problemes derivats d'incoherències en la utilització de diferents jocs de caràcters.