Joc de Caràcters en Aplicacions Canigó

A qui va dirigit

Aquest how-to va dirigit a tots aquells que al desenvolupar una aplicació amb Canigó tinguin necessitat d'utilitzar jocs de caràcters més amplis que l'ASCII.

Versió de Canigó

Els passos descrits en aquest document són d'aplicació a les versions 2.x de Canigó.

Introducció

En aquest document es presenten algunes recomanacions, en relació amb el joc de caràcters utilitzat en els diferents elements de l'arquitectura, per aplicacions web desenvolupades amb Canigó.

Es fa un repàs als problemes més habituals i per cada un s'indiquen els elements a tenir en compte per que el problema no es produeixi.

Finalment, es presenta un problema detectat en el funcionament d'Hibernate i relacionat amb el joc de caràcters en ús.

Joc de caràcters per defecte

En alguns dels apartats que segueixen es fa referència al joc de caràcters per defecte de la màquina virtual Java.

El joc de caràcters per defecte té conseqüències en totes les operacions que impliquin conversions de texts entre la representació interna de la màquina virtual Java (String, etc.) i una representació externa (arxius, transmissions per xarxa, etc.) quan no s'utilitzi un mecanisme de conversió que especifiqui l'ús d'un joc de caràcters explícit (per exemple i com es veurà més endavant, els drivers JDBC d'Oracle incorporen el seu propi mecanisme de conversió, que es configura de forma diferent).

Aquest joc de caràcters es configura automàticament a l'arrancar la màquina virtual Java a partir de factors i amb valors que depenen del sistema operatiu, idioma, variables d'entorn, etc. i per tant és molt probable que canviï entre els diferents entorns en els que es desenvolupa, prova i utilitza una aplicació.

És aconsellable que els desenvolupadors d'una aplicació siguin conscients de quin és el valor del joc de caràcters per defecte en els diferents entorns en els que s'executarà l'aplicació, des de Desenvolupament fins a Producció, per tal de preveure els possibles problemes que se'n puguin derivar.

Si fos necessari canviar-lo, pot definir-se el paràmetre file.encoding a l'arrancar la màquina virtual. Per exemple

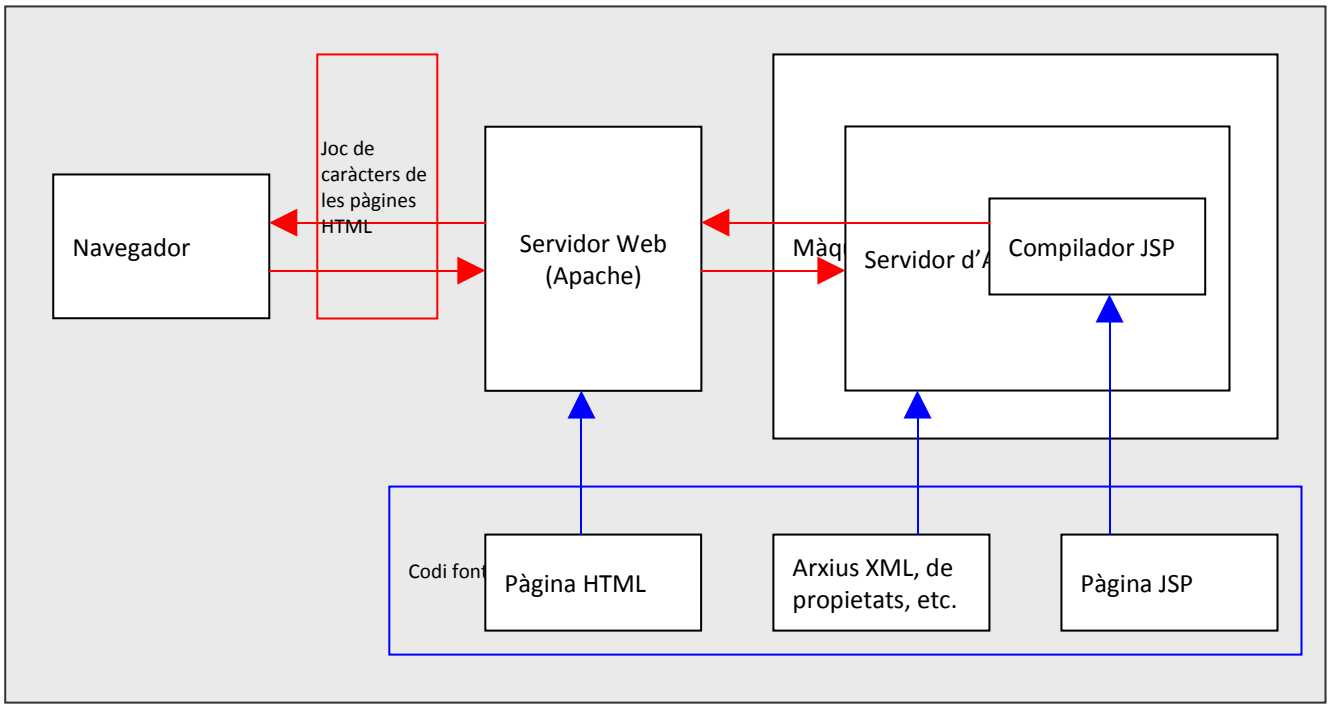
```
-Dfile.encoding=utf-8

-Dfile.encoding=iso-8859-15
```

però cal tenir en compte que aquesta definició afectarà a tota la màquina virtual i, per tant, al servidor d'aplicacions i a totes les aplicacions que hi hagi instal·lades.

Generació de les pàgines de l'aplicació

En el següent diagrama es presenten els elements involucrats en la generació de la pàgina que es presenta a l'usuari mitjançant el navegador i els diferents punts del procés on és important tenir en compte el joc de caràcters que s'està utilitzant



El codi font de l'aplicació, guardat en forma d'arxius de text, ha d'estar codificat amb un joc de caràcters compatible amb el procés de lectura de l'arxiu que en farà el servidor d'aplicacions.

En el cas de les JSPs, aquest codi font serà compilat i executat. Durant la execució s'hi barrejarà text procedent, per exemple, dels arxius de propietats d'internacionalització i dades procedents de sistemes externs i el resultat serà enviat via xarxa cap al navegador, on serà convertit en la representació que veurà l'usuari.

El contingut estàtic (pàgines HTML), guardat també en arxius de text, serà llegit pel servidor Web (Apache) i enviat també via xarxa cap al navegador.

En el cas dels formularis, les dades introduïdes per l'usuari seran enviades pel navegador via xarxa cap al servidor d'aplicacions, on es convertiran al format intern per que l'aplicació les processi.

En tot aquest procés és molt important que es mantingui la coherència en la codificació del text.

Codi font de l'aplicació

Alguns dels elements que constitueixen el codi font de l'aplicació poden contenir caràcters no ASCII i és important garantir que aquests caràcters seran interpretats correctament.

Els principals elements afectats son les pàgines HTML i JSP i els arxius de propietats, sobre tot els utilitzats per la internacionalització de l'aplicació, però això és aplicable a qualsevol altre arxiu de dades de l'aplicació que contingui text amb caràcters no ASCII.

En el cas dels arxius que son llegits pel servidor d'aplicacions (JSP, propietats), al crear-los s'ha de vigilar d'utilitzar un joc de caràcters compatible amb el que utilitzarà posteriorment l'aplicació per llegir-los o interpretar-los.

Si la creació dels arxius es fa amb una barreja d'eines (Eclipse, editors de text, etc.) en una varietat d'entorns (Windows, Linux, etc.) pot ser que el resultat no sigui completament coherent, per lo que es recomana estandarditzar els entorns i/o els procediments de desenvolupament per que tots els arxius es creïn amb el mateix joc de caràcters.

JSPs

En el cas de les JSPs, es pot informar al servidor d'aplicacions / compilador de la codificació utilitzada per crear l'arxiu JSP mitjançant l'atribut pageEncoding de la directiva page i el valor per defecte és iso-8859-1

Per exemple:

```
<%@ page contentType="text/html; charset=ISO-8859-15" pageEncoding="ISO-8859-15"%>
```

XML

Normalment els arxius xml comencen per un tag que indica en quin joc de caràcters ha estat creat l'arxiu. Per exemple

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

Cal que el joc de caràcters indicat en cada cas correspongui amb el que s'ha fer servir realment per crear l'arxiu.

Arxius de propietats

En el cas dels arxius de propietats o altres arxius de dades utilitzats en l'aplicació, no és possible especificar en el propi arxiu el joc de caràcters utilitzat i la lectura i corresponent conversió es fa assumint que l'arxiu està codificat amb el joc de caràcters per defecte de la màquina virtual.

Contingut estàtic (html, css, javascript,...)

En el cas del contingut estàtic, el text de l'arxiu no és interpretat per l'Apache, que l'envia al navegador tal com estigui codificat en origen.

Joc de caràcters de les pàgines HTML

Les pàgines HTML enviades als navegadors es generen mitjançant struts / tiles a partir de contingut estàtic, JSPs i llibreries de tags i contenen, entre altres coses, la definició del joc de caràcters amb el que s'han de representar.

Aquesta definició és important tant per que el navegador realitzi correctament la presentació com per que els valors introduïts en els diferents camps dels formularis siguin correctament retornats a l'aplicació.

Per evitar dependre de les configuracions per defecte dels sistemes operatius, màquina virtual Java, el servidor d'aplicacions, el servidor web i els navegadors, és convenient declarar explícitament en cada pàgina quin és el joc de caràcters a utilitzar.

Per fer-ho, es pot incloure en cada JSP la declaració del joc de caràcters, utilitzant l'atribut charset de la directiva page. Per exemple

```
<%@ page contentType="text/html; charset=ISO-8859-15" pageEncoding="ISO-8859-15"%>
```

o bé incorporar a l'aplicació un filtre, que forma part de la llibreria Spring, i que afegeix aquesta definició "al vol" en totes les pàgines:

```
<filter>
    <filter-name>SetCharacterEncodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>ISO-8859-15</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>
```

En el cas dels arxius HTML estàtics servits des del servidor Apache i que per tant no es veurien afectats pel filtre anterior, és convenient indicar al navegador la codificació amb la que s'ha creat l'arxiu utilitzant l'atribut charset del tag meta

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
</head>
```

i el valor indicat ha de correspondre a la codificació utilitzada en el moment de crear l'arxiu de text.

Si no s'inclou aquest tag, és recomanable que el servidor Apache afegeixi en la capçalera de la resposta HTTP el joc de caràcters utilitzat en el contingut estàtic i que aquest coincideixi amb el joc de caràcters real amb que s'ha creat l'arxiu.

Finalment, **és molt recomanable que tots els elements que composen la interfase d'usuari comparteixin el mateix joc de caràcters**, per tal d'evitar problemes en la representació o en la codificació de les dades de retorn dels formularis.

Selecció del joc de caràcters

El joc de caràcters escollit ha de ser suficient per als diferents idiomes pels que s'hagi de donar suport en l'aplicació.

Normalment el iso-8859-1 és una bona elecció per aplicacions que donin suport al català, espanyol i altres idiomes basats en variants de l'alfabet llatí i té l'avantatge de ser el joc de caràcters per defecte especificat en el protocol HTTP. L'inconvenient és que no inclou el caràcter €, que requereix de passar a un joc de caràcters posterior (iso-8859-15).

Com alternativa, el utf-8 és un joc de caràcters que suporta pràcticament qualsevol idioma i funciona correctament en el context dels servidors d'aplicacions i les pàgines HTML, però pot resultar problemàtic per utilitzar-lo fora d'aquest context, per exemple per arxius de text o dades que s'han de tractar amb programes desenvolupats en altres llenguatges.

Configuració del servidor de base de dades

En la majoria d'aplicacions desenvolupades amb Canigó s'utilitza Oracle com a servidor de base de dades.

Al crear la base de dades, cal tenir en compte un parell de paràmetres que tenen relació amb el joc de caràcters i l'idioma. Aquesta configuració afecta a tota la base de dades i per tant és única i comú a tots els esquemes que s'hi creïn.

En primer lloc, el joc de caràcters de la base de dades ha de ser suficient per emmagatzemar les dades de l'aplicació. Pels cassos més habituals es recomana utilitzar el WE8ISO8859P15 que equival al iso-8859-15, tot i que s'han d'analitzar els requeriments de l'aplicació i veure si afecten a la elecció del joc de caràcters.

El segon punt important és l'idioma de la base de dades. L'idioma determina, entre altres coses, el criteri d'ordenació i, per tant, afecta a la forma com son creats els índexs. Si la BD es crea amb AMERICAN_AMERICA la ordenació deixa les vocals accentuades al final, mentre que si l'idioma és, per exemple, SPANISH_SPAIN les vocals accentuades queden junt amb les corresponents no accentuades. A més, això pot tenir impacte sobre la utilització o no d'un índex en la recuperació de resultats ordenats, per lo que es recomana fer un estudi detallat dels requeriments i del disseny de la BD.

Finalment, cal tenir en compte que el Client utilitzat per l'accés a la BD imposa el seu propi joc de caràcters i el seu propi idioma sobre els interns de la BD. Si no coincideixen , Oracle fa la conversió "al vol". Els valors per defecte de l'idioma i joc de caràcters del client oracle, que en aquest cas és l'aplicació desenvolupada amb Canigó, els determina el driver JDBC i la forma de fer-ho és diferent pels drivers Thin i els drivers OCI.

Com a conclusió, es recomana tenir en compte els diferents elements que composen l'arquitectura i la configuració de cada un d'ells, per verificar que és possible emmagatzemar i recuperar les dades segons els requeriments de l'aplicació i que els accessos a la BD son suficientment òptims.

Problema amb algunes queries Hibernate

S'han reportat alguns problemes a l'executar queries Hibernate que utilitzen caràcters no ASCII.

Un cas típic d'aquest problema es produeix quan la query es construeix utilitzant els marcadors [] per indicar la posició d'algun dels paràmetres i el valor d'aquests paràmetres conté caràcters no ASCII, com per exemple vocals accentuades o el símbol de l'euro (€).

Un exemple d'una d'aquestes queries, utilitzant la plantilla de Canigó, podria ser a canigo-services-web-lists.xml :

```
<entry key="categoriesList">
    <bean parent="baseHibernateAdapter">
        <property name="hql">
            <value>
                FROM
                net.gencat.ctti.canigo.samples.prototip.model.Category
                AS vo WHERE 1=1
                /~descn: AND vo.descn LIKE '[descn]' ~/
                /~sortColumn: ORDER BY vo.[sortColumn] [sortDirection]~/
            </value>
        </property>
    </bean>
</entry>
```

En aquests cassos, s'observa que la instrucció SQL que s'acaba enviant al gestor de base de dades pot tenir valors erronis pels caràcters no ASCII.

Per exemple, si en el cas anterior descn = "€" llavors la query que s'envia a Oracle és

```
select category0_.CATID as CATID, category0_.NAME as NAME0_, category0_.DESCN as DESCN0_
from CATEGORY category0_ where 1=1 and (category0_.DESCN like '?') order by category0_.CATID
asc
```

Diagnòstic del problema

En les queries construïdes utilitzant [ ] com a marcador dels paràmetres, es realitza una substitució de strings per canviar el nom del paràmetre pel seu valor i construir una query HQL sense paràmetres, que llavors és passada a Hibernate per la seva execució. Al fer aquesta operació, la query HQL que s'acaba executant pot contenir caràcters no ASCII.

Amb l'exemple anterior, si el valor del paràmetre descn és "€" la query HQL que s'executarà realment és

```
FROM
net.gencat.ctti.canigo.samples.prototip.model.Category
AS vo WHERE 1=1
 AND vo.descn LIKE '€'
 ORDER BY vo.id asc
```

i s'observa que el parser HQL de Hibernate no tracta correctament els caràcters no ASCII.

Per poder traçar aquest comportament del parser HQL és suficient activar en els logs de l'aplicació

```
<category name="org.hibernate.hql.ast.QueryTranslatorImpl">
    <level value="debug"/>
</category>
```

i per veure el SQL que s'envia al gestor de BD es pot activar el log

```
<category name="org.hibernate.SQL">
    <level value="debug"/>
</category>
```

Cas de prova

S'ha preparat un petit cas de prova per reproduir el problema.

El cas de prova consisteix en una JSP que reprodueix una part del codi de Hibernate. Concretament del mètode parse de la classe org.hibernate.hql.ast.QueryTranslatorImpl:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-15"
pageEncoding="ISO-8859-15"%>
<%@ page import="java.nio.charset.Charset" %>
<%@ page import="org.hibernate.hql.antlr.HqlTokenTypes" %>
<%@ page import="org.hibernate.hql.ast.ASTPrinter" %>
<%@ page import="org.hibernate.hql.ast.HqlParser" %>
<%@ page import="antlr.RecognitionException" %>
<%@ page import="antlr.TokenStreamException" %>
<%@ page import="antlr.collections.AST" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-15">
<title>Prova Parser Hibernate</title>
</head>
<body>
<h1>Prova Parser Hibernate</h1>
<%
String HQL = "from a where a.b = 'aeiouáéíóúàèïòüñçAEIOUÁÉÍÓÚÀÈÏÒÜÑÇ€'";
%>
<hr>
defaultCharset = <tt><%=Charset.defaultCharset().displayName()%></tt><br>
HQL = <tt><%=HQL%></tt><br>
<hr>
<%
HqlParser parser = HqlParser.getInstance(HQL);
parser.setFilter(true);
parser.statement();
AST hqlAst = parser.getAST();
ASTPrinter printer = new ASTPrinter( HqlTokenTypes.class );
printer.setShowClassNames(false);
String resultatParser = printer.showAsString( hqlAst, "--- resultatParser ---" );
%>
<pre>
<%=resultatParser%>
</pre>
<hr>
```

```
</body>
</html>
```

Aquesta JSP es pot incloure i executar en una aplicació que contingui les llibreries d'Hibernate i les seves dependències.

Aquesta prova posa de manifest una característica important del problema i és que la gravetat del mateix depèn del joc de caràcters per defecte de la màquina virtual Java.

El joc de caràcters per defecte s'estableix a partir de paràmetres del sistema operatiu i depèn, òbviament, del sistema operatiu mateix però es pot forcar definint el paràmetre file.encoding a l'arrancar la màquina virtual. Per exemple

```
-Dfile.encoding=utf-8

-Dfile.encoding=iso-8859-15
```

El cas de prova mostra quin és el joc de caràcters que s'està utilitzant i a l'executar-lo s'observa que si el joc de caràcters és, per exemple, iso-9959-15 l'únic caràcter que es corromp durant la interpretació del HQL és el de l'euro (€) mentre que si el joc de caràcters és utf-8 el parser HQL corromp tots els caràcters no-ascii del cas de prova.

Examinant amb més detall el codi intern d'Hibernate sembla que el problema deriva de varies conversions entre Strings i arrays de bytes, que es veuen afectades pel joc de caràcters definit.

Possibles solucions

El problema sembla estar en el comportament intern del parser HQL, generat amb ANTLR a partir de la gramàtica HQL i, per tant, està completament fora de l'abast de Canigó el arreglar-lo.

Donat que, com s'ha observat, el joc de caràcters per defecte de la màquina virtual té un impacte important sobre l'abast del problema, es recomana verificar quin és el valor actualment en us, quin és el seu efecte sobre el parser HQL i si és possible canviar-lo per un altre que minimitzi el problema. Per això es pot utilitzar la JSP amb el cas de prova.

Entre els diferents jocs de caràcters que s'han provat, sembla que els iso-8859-1 i iso-8859-15 redueixen el problema al caràcter "€" mentre que el utf-8 l'amplia a totes les vocals accentuades (i probablement a la majoria dels caràcters no ASCII).

El problema desapareix si la expressió HQL està formada únicament per caràcters ASCII. Normalment els caràcters no-ascii procedeixen de la substitució dels paràmetres marcats amb [ ]. En canvi, pels paràmetres marcats amb { }, el valor s'estableix mitjançant la substitució JDBC, un cop construïda i parsejada la query.

Per tant, es recomana que tots els paràmetres de la query es marquin amb { }. Això té, a més, altres efectes beneficiosos:

- elimina la possibilitat de realitzar injeccions de SQL en els paràmetres
- afavoreix els mecanismes de cache de queries del servidor de base de dades ja que el text de la query no canvia entre execucions, tot i que ho facin els valors dels paràmetres