Com fer la paginació amb grans volums de dades

A qui va dirigit

Aquest how-to va dirigit a tots aquells desenvolupadors d'aplicacions Canigó que facin us de la paginació Valuelist de Canigó, especialment si el seu model de dades incorpora grans volums d'informació.

En aquest document es presenta el procediment per configurar la nova implementació de la paginació.

Versió de Canigó

Els passos descrits en aquest document són aplicables a partir de la **versió 2.3.5 de Canigó**.

Requeriments

Podeu aplicar aquest procediment per aplicacions web Canigó.

Context

La implementació de la paginació en Canigó exten el projecte de codi obert Valuelist. La classe que permet adaptar la paginació a Hibernate s'anomena *Hibernate30Adapter*. Tant la implementació original d'aquesta classe com l'extensió que en fa Canigó utilitzen el tipus estàndard de JDBC *ScrollableResultset* com a mecanisme de paginació.

L'avantatge d'utilitzar ScrollableResultset és que amb una sola query s'aconsegueix posicionar el resultset al punt de paginació i també obtenir el nombre de registres posicionant-se al final del resultset amb el mètode *last()*.

L'inconvenient d'aquesta implementació sorgeix quan el volum de dades és gran i la implementació de scrollableResultset no és òptima. Aquest és el cas del driver JDBC d'Oracle, que fins i tot en versions recents té un temps d'accés que es proporcional al número de registres.

A partir de la versió 2.3.5 de Canigó s'ha posat a disposició una nova implementació, *Hibernate30AdapterV2*, que permet configurar la paginació per determinar-ne el comportament.

> **Important**: **Abans d'aplicar qualsevol optimització és molt recomanable revisar els requeriments i el disseny de l'aplicació. Consultes que han de paginar volums grans de registres solen ser indicadors de dissenys incorrectes o requeriment poc raonables. Per norma general una llista amb milers o milions de registres no és manejable ni útil per l'usuari i pot comportar temps de resposta no acceptables**.

Objectius

L'objectiu d'aquest howTo es descriure com usar i configurar el nou component de paginació:

Hibernate30AdapterV2

La configuració que es presenta permet adaptar-se a diferents escenaris d'ús, particularment amb grans volums de dades.

Com fer la paginació amb grans volums de dades

Procediment

0. Passos previs

Seguint el descrit a la documentació de Canigó sobre el servei de Llistats, s'ha de disposar d'un fitxer *canigo-services-web-lists.xml* on es defineixen els següents beans:

vlConfig, valueListActionHelper, extendedValueListActionHelper, valueListHelper, valueListHandler

1. Definició de les paginacions de l'aplicació

Per cadascuna de les paginacions de l'aplicació s'ha de crear una entrada en el *bean valueListHandler* (en el mapa *config.adapters*).

Aquestes entrades poden ser especialitzades en diferents fonts de dades (fitxers, fitxers de log, Hibernate, etc).

En aquest document ens centrarem únicament en fonts de dades Hibernate.

a. Definició de l'adaptador per fonts de dades Hibernate:

Per comoditat es crea un *bean* configurat amb els valors per defecte pels adaptadors de Hibernate. Aquest *bean* s'anomena *baseHibernateAdapter* i és del nou tipus :

net.gencat.ctti.canigo.services.web.lists.Hibernate30AdapterV2

Les propietats comunes i que normalment es defineixen són les següents:

```
<bean name="baseHibernateAdapter"
class="net.gencat.ctti.canigo.services.web.lists.Hibernate30AdapterV2">
                    <property name="sessionFactory" ref="sessionFactory"/>
                    <property name="defaultNumberPerPage" value="5"/>
                    <property name="defaultSortColumn" value="id"/>
                    <property name="defaultSortDirection" value="asc"/>
                    <property name="removeEmptyStrings" value="true"/>
                    <property name="focusOptimalization" value="false"/>
          </bean>
```

Aquests valors han de ser modificats a l'aplicació per adaptar-se a les necessitats de la mateixa.

Important: Per poder accedir a les noves funcionalitats de paginació cal assegurar que es fa servir la classe: *net.gencat.ctti.canigo.services.web.lists.Hibernate30AdapterV2* en lloc de *net.gencat.ctti.canigo.services.web.lists.Hibernate30Adapter*

Com fer la paginació amb grans volums de dades

L'ús de la nova classe:

net.gencat.ctti.canigo.services.web.lists.Hibernate30AdapterV2

requereix que el bean *valueListHelper* sigui del tipus:

net.gencat.ctti.canigo.services.web.lists.ValueListHandlerHelper

```
<bean id="valueListHelper"
class="net.gencat.ctti.canigo.services.web.lists.ValueListHandlerHelper">
        <property name="valueListHandler">
            <ref bean="valueListHandler" />
        </property>
    </bean>
```

b. Configuració específica de cada paginació:

Com s'ha dit abans, per cadascuna de les paginacions de l'aplicació basades en Hibernate s'ha de crear una entrada en el bean *valueListHandler* (en el mapa config.adapters).

Aquestes entrades són *beans* que tenen com a pare el *bean baseHibernateAdapter* el qual hem definit en el punt previ.

Aquí es definiran les propietats particulars de cada paginació:

- Consulta d'obtenció de les dades (propietat *hql*)
- Estratègia de paginació

En aquest howTo ens centrarem en les propietats que defineixen l'estratègia de paginació.

A continuació es descriuen les dues estratègies implementades i finalment els paràmetres addicionals que permeten ajustar el seu comportament.

b1. Estratègia basada en ScrollableResultSets

Aquesta es l'estratègia original del projecte ValueList. Els paràmetres de configuració són els següents:

```
<property name="usaScrollableResultSet"  value="true"/>
<property name="usaSelectCount" value="false"/>
```

El desavantatge d'aquesta estratègia és que el rendiment es degrada(*) linealment segons creix el nombre de registres a paginar, independentment de la pàgina que s'estigui presentant.

Aquesta es l'estratègia per defecte del nou adaptador, per tant quan no s'especifica aquests propietats el comportament serà el mateix que si les especifiquem així.

(*) Aquest comportament s'ha observat amb Oracle

b2. Estratègia basada en setFirstResult / setMaxResults

Aquesta estratègia fa servir l'api de paginació estàndard de Hibernate

```
<property name="usaScrollableResultSet"  value="false"/>
<property name="usaSelectCount"value="true"/>
```

D'aquesta forma Hibernate pot traduir a sentències SQL més òptimes (basades en ROWNUM).

Com fer la paginació amb grans volums de dades

Per aflorar aquestes optimitzacions heu de verificar que es fa servir el dialecte Hibernate apropiat.

El desavantatge d'aquesta estratègia és que Canigó ha de construir de forma programàtica una variació de la query original per generar un "select count(*)". Per tant només podreu aplicar aquesta estratègia si substituir els camps i expressions del Select per count(*) porta a una query vàlida i que donarà el número de registres.

Normalment es fan queries que projecten objectes d'una única classe i per tant aquesta estratègia serà aplicable.

Per exemple, aquesta estratègia no seria aplicable quan hi hagi un DISTINCT en la SELECT. En aquest cas el SELECT COUNT(*) retornaria un nombre erroni de registres. Per mirar de corregir aquesta situació vegeu el paràmetre *selectCountPart* en el següent apartat.

b3. Paràmetres addicionals

A més del paràmetres que defineixen l'estratègia s'han de considerar les següents possibilitats de configuració.

- **tamanyMaximResultset**.

Limita el nombre de registres de la consulta. El major guany s'obté en el cas de usaScrollableResultSet=true ja que era el més penalitzat en resultsets grans. En l'altre cas el guany es que evitarà que l'usuari pugui anar als casos més ineficients (últimes pàgines).

En aquest cas és molt important aflorar a l'usuari el fet de que la consulta té més registres dels que es presenten. Veure el apartat *c* d'aquest document.

- **selectCountPart.**

Aquest paràmetre es fa servir en els casos en que usaSelectCount=true i la part SELECT de la query fa servir DISTINCT o altres modificadors que puguin dur a COUNT(*) erroni. (veure apartat b2).

Per exemple si la query original és:

SELECT DISTINCT VO

FROM net.gencat.canigo.exemple.model.persona as VO,

net.gencat.canigo.exemple.model.animaldecompanyia as VO2

Aquí caldria assignar el següent valor a *selectCountPart:*

```
<property name="selectCountPart"value="SELECT COUNT(DISTINCT VO) "/>
```

- **cacheable** (*)

Indica si les queries de la cerca i les dades de la pàgina són susceptibles de ser incloses a la caché.

- **cacheRegion** (*)

Ha de contenir el nom de la zona de regió que es farà servir. Aquesta s'ha d'haver configurat a Hibernate

(*) Una possibilitat per millorar el temps de resposta de la paginació pot ser configurar la caché de segon nivell de Hibernate i fer-la servir per desar els resultats de les queries. Per exemple, això permetria que les primeres pàgines de les consultes (les més frequents) no requerissin accedir al backend. La configuració de les caches de Hibernate excedeix l'abast d'aquest document, però s'ha volgut oferir la possibilitat de configurar mitjançant els paràmetres "cacheable" i "cacheRegion" l'ús de cache.

c. Limitar el número màxim de pàgines

Quan es fa servir la propietat *tamanyMaximResultset* és important fer aflorar a l'usuari el fet de que el nombre de registres retornats ha quedat limitat a aquest valor, i que per tant no es presenten tots els registres que complien el filtre.

Per tal de notificar a l'usuari s'haurà d'afegir una construcció similar a la que es presenta en aquest exemple.

Si es fa servir la propietat *tamanyMaximResultset* pot ser interessant que la vista JSP avisi d'aquest fet. Podeu incorporar aquest codi a la JSP, que comprova l'atribut "limitedRows" de la informació del Valuelist per saber si la consulta ha estat restringida.

Si s'ha fet ús de SelectCount llavors la propietat actualRowCount de la informació del valueList estarà informada i contindrà el número de registres real.

```
<c:if test=
        "${(list.valueListInfo.class.name==
         'net.gencat.ctti.canigo.services.web.lists.ExtendedValueListInfo'
         )
        &&
        list.valueListInfo.limitedRows}">
        <fmt:message key="valuelist.limit.paginacio">
        <fmt:param
                        value="${list.valueListInfo.totalNumberOfPages}">
                </fmt:param>
        </fmt:message>
</c:if>
```

Els tag fmt:message fa servir el missatge següents, que caldrà afegir a i18n.properties, i que podeu adequar al vostre gust.

```
valuelist.limit.paginacio=La seva consulta ha retornat més de {0} registres. El resultat presentat s'ha limitat a {0} registres
```