

Llistat paginable a Servidor

A qui va dirigit

Aquest how-to va dirigit a tots aquells usuaris que tinguin la necessitat de fer servir un llistat paginable a Servidor amb Canigó 3 .

Versió de Canigó

Els passos descrits en aquest document apliquen a la darrera versió del Framework Canigó 3. La implementació de l'exemple proposat és compatible per JSF2 / Richfaces 4.0.0.

Introducció

Així com Canigó 2 comptava amb un servei de llistats propi (que ja realitzava la paginació a servidor), a Canigó 3 no existeix cap servei de llistats, es nodreix de JSF - Richfaces 4 per generar llistats. Tot i això, la paginació per defecte que ofereixen els components de Richfaces es realitza al costat client. Quan es treballa amb llistats que han de gestionar gran número de registres es poden patir problemes de rendiment per temps de càrrega elevats. La opció a aplicar per millorar aquesta situació és la de realitzar la paginació dels resultats al costat servidor, de manera que es segmenta la informació en blocs de dades més petits i no afecten negativament al rendiment de l'aplicació. Els usuaris que es trobin amb aquest problema poden fer servir aquest howTo per construir-ne la seva pròpia paginació a servidor.

Llistat paginable a Servidor

Implementació exemple llistat amb paginació a servidor

Característiques de l'exemple seguit al Howto

L'objectiu principal d'aquest exemple és el d'explicar una de les possibles maneres d'implementar un llistat paginable al costat servidor. Com a afegit per aquest exemple, també s'inclou la ordenació i cerca dels resultats.

Implementació de la vista

S'ha fet ús dels següents components:

- `<rich:extendedDataTable>` com a contenidor general de les dades i vinculat amb el llistat del Bean.
- `<rich:dataScroller>` per manegar la navegació dels resultats.
- `<rich:column>` per renderitzar, ordenar i filtrar els valors.

La generació de la vista s'ha separat en 2 arxius: el que conté la taula i el que serveix de plantilla per a les diferents columnes d'aquesta:

llistatArrangable.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:rich="http://richfaces.org/rich"
    xmlns:c="http://java.sun.com/jstl/core"
    xmlns:a4j="http://richfaces.org/a4j"
    template="/views/layouts/template.jsf">
    <ui:define name="body">
        <h:form id="form">
            <rich:extendedDataTable keepSaved="true" id="richTable" var="record"
                value="#{serverPaginationBean.dataModel}" rows="5">
                <ui:include src="/views/jpaColumn.xhtml">
                    <ui:param name="bean" value="#{serverPaginationBean}" />
                    <ui:param name="property" value="nom" />
                </ui:include>
                <ui:include src="/views/jpaColumn.xhtml">
                    <ui:param name="bean" value="#{serverPaginationBean}" />
                    <ui:param name="property" value="cognoms" />
                </ui:include>
                <ui:include src="/views/jpaColumn.xhtml">
                    <ui:param name="bean" value="#{serverPaginationBean}" />
                    <ui:param name="property" value="telefon" />
                </ui:include>
                <ui:include src="/views/jpaColumn.xhtml">
                    <ui:param name="bean" value="#{serverPaginationBean}" />
                    <ui:param name="property" value="email" />
                </ui:include>
                <ui:include src="/views/jpaColumn.xhtml">
                    <ui:param name="bean" value="#{serverPaginationBean}" />
                    <ui:param name="property" value="dataNaixement" />
                </ui:include>
                <f:facet name="footer">
                    <rich:dataScroller id="scroller" />
                </f:facet>
            </rich:extendedDataTable>
        </h:form>
    </ui:define>
</ui:composition>
```

Llistat paginable a Servidor

jpaColumn.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:rich="http://richfaces.org/rich"
xmlns:a4j="http://richfaces.org/a4j"
xmlns:c="http://java.sun.com/jsp/jstl/core">
<ui:composition>
<rich:column sortBy="#{property}"
sortOrder="#{serverPaginationBean.sortOrders[property]}"
filterValue="#{serverPaginationBean.filterValues[property]}"
filterExpression="#{property}">
<f:facet name="header">
<b><h:outputLabel value="#{property}" /></b>
<h:commandLink action="#{serverPaginationBean.toggleSort}">
#{serverPaginationBean.sortOrders[property]}
<a4j:ajax render="richTable" />
<f:setPropertyActionListener target="#{serverPaginationBean.sortProperty}"
value="#{property}" />
</h:commandLink>
<br />
<h:inputText value="#{serverPaginationBean.filterValues[property]}">
<a4j:ajax render="richTable@body scroller" event="keyup" />
</h:inputText>
</f:facet>
<h:outputText value="#{record[property]}" />
</rich:column>
</ui:composition>
</html>
```

L'aspecte de la renderització d'aquesta vista quedaria semblant a la següent captura:

nom unsorted	cognoms unsorted	telefon unsorted	email unsorted	dataNaixement unsorted
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

test2	test	99545511	tttest@test.cat	2013-11-22
test3	test	test	test@test.es	2013-11-07
test3	test	test	test@test.es	2013-11-07
Joan	Gracia	93453784	email1@gencat.cat	2013-11-25
Nuria	Martin	95443784	email2@gencat.cat	2013-11-25

<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
----------------------	----------------------	----------------------	----------------------	----------------------

Caldria adaptar els estils per tal que compleixin els definits a la guia d'us d'estils corporatius:
<http://www.gencat.cat/web/guies/aplicacions/ca/taules.html>

Llistat paginable a Servidor

Configuració de Bean i dataModel del llistat

ServerPaginationBean.java

```
@Component("serverPaginationBean")
@Scope(value="session")
public class ServerPaginationBean implements Serializable {

    private static final long serialVersionUID = 3612067009208008814L;

    @Autowired
    private TbUsuariService service;
    private Map<String, SortOrder> sortOrders = new HashMap<String, SortOrder>();
    private Map<String, String> filterValues = new HashMap<String, String>();
    private String sortProperty;
    private Object dataModel;

    public ServerPaginationBean(){
        //De sèrie definim les columnes de la taula com a no-ordenades
        sortOrders.put("nom", SortOrder.unsorted);
        sortOrders.put("cognoms", SortOrder.unsorted);
        sortOrders.put("telefon", SortOrder.unsorted);
        sortOrders.put("email", SortOrder.unsorted);
        sortOrders.put("dataNaixement", SortOrder.unsorted);
    }

    public void toggleSort() {
        for (Entry<String, SortOrder> entry : sortOrders.entrySet()) {
            SortOrder newOrder;
            if (entry.getKey().equals(sortProperty)) {
                if (entry.getValue() == SortOrder.ascending) {
                    newOrder = SortOrder.descending;
                } else {
                    newOrder = SortOrder.ascending;
                }
            } else {
                newOrder = SortOrder.unsorted;
            }
            entry.setValue(newOrder);
        }
    }

    public Object getDataModel() {
        this.dataModel=new ServerPaginationDataModel<TbUsuaris>(service, filterValues,
sortOrders);
        return this.dataModel;
    }

    public Map<String, SortOrder> getSortOrders() {return sortOrders;}
    public Map<String, String> getFilterValues() {return filterValues;}
    public String getSortProperty() {return sortProperty;}
    public void setSortProperty(String sortPropety) {
        this.sortProperty = sortPropety;
    }
}
```

A aquest Bean trobem:

- **service** : Bean de servei encarregat de gestionar la comunicació amb els DAO, per accedir-hi a realitzar les operacions contra BBDD.
- **sortOrders**: Emmagatzema l'estat d'ordenació de les columnes de cara a realitzar la cerca.
- **filterValues**: Emmagatzema els filtres aplicats per a cada columna de cara a realitzar la cerca.
- **sortProperty**: Variable que fa servir la vista per comunicar al Bean quina columna és l'objecte del canvi d'ordenació que s'ha efectuat.

Llistat paginable a Servidor

- **dataModel**: Model de dades customitzat que conté i gestiona les dades a renderitzar a la taula.
- El mètode **toggleSort** es crida des de les capçaleres de les columnes de la taula mitjançant un enllaç i serveix per informar del canvi de ordenació de columna.
- El mètode **getDataModel** és qui s'encarrega de re-carregar les dades del llistat, fent servir la informació recollida sobre ordenació, filtrat i paginació. És cridat quan s'informa un filtre, s'ordena o es pagina a la taula.

ServerPaginationDataModel.java

```
public class ServerPaginationDataModel<T> extends ExtendedDataModel<T> implements
Arrangeable{

    private static final long serialVersionUID = -1956179896877538628L;
    private ArrangeableState arrangeableState;

    private Integer currentPk;
    private Integer rowCount;
    private Map<Integer, TbUsuaris> wrappedData = new HashMap<Integer, TbUsuaris>();
    private List<Integer> wrappedKeys = null;
    private TbUsuariService service;
    private Map<String, String> filterValues;
    private Map<String, SortOrder> sortOrders;

    public ServerPaginationDataModel(TbUsuariService service, Map<String, String>
filterValues, Map<String, SortOrder> sortOrders) {
        this.service=service;
        this.filterValues=filterValues;
        this.sortOrders=sortOrders;
    }

    @Override
    public Object getRowKey() {return currentPk; }

    @Override
    public void setRowKey(Object key) {this.currentPk = (Integer) key; }

    @Override
    public void walk(FacesContext context, DataVisitor visitor, Range range, Object
argument) {
        int firstRow = ((SequenceRange) range).getFirstRow();
        int numberOfRows = ((SequenceRange) range).getRows();
        wrappedKeys = new ArrayList<Integer>();
        for (TbUsuaris item : this.service.getItemsByrange(new Integer(firstRow),
numberOfRows, filterValues, sortOrders)) {
            wrappedKeys.add(item.getIdUsuari());
            wrappedData.put(item.getIdUsuari(), item);
            visitor.process(context, item.getIdUsuari(), argument);
        }
    }

    @Override
    public int getRowCount() {
        if (rowCount == null) {
            rowCount = new Integer(this.service.getRowCount(filterValues));
            return rowCount.intValue();
        } else {
            return rowCount.intValue();
        }
    }

    //Continua a la pàgina següent ...
}
```

Llistat paginable a Servidor

```
// Continua des de la pàgina anterior ...

@SuppressWarnings("unchecked")
@Override
public T getRowData() {
    if (currentPk == null) {
        return null;
    } else {
        TbUsuaris ret = wrappedData.get(currentPk);
        if (ret == null) {
            ret = this.service.getTbUsuariItemByPk(currentPk);
            wrappedData.put(currentPk, ret);
        }
        return (T) ret;
    }
}

// Unused rudiment from old JSF staff.
@Override
public int getRowIndex() {return 0;}

// Unused rudiment from old JSF staff.
@Override
public Object getWrappedData() {throw new UnsupportedOperationException();}

// Never called by framework.
@Override
public boolean isRowAvailable() {
    if (currentPk == null) {
        return false;
    } else {
        return this.service.hasTbUsuariItemByPk(currentPk);
    }
}

// Unused rudiment from old JSF staff.
@Override
public void setRowIndex(int rowIndex) {/* ignore */}

// Unused rudiment from old JSF staff.
@Override
public void setWrappedData(Object data) {throw new UnsupportedOperationException();}

protected ArrangeableState getArrangeableState() {return arrangeableState;}

@Override
public void arrange(FacesContext context, ArrangeableState state) {
    arrangeableState = state;
}
}
```

A aquest dataModel customitzat trobem:

- **currentPK**: clau de referència per obtenir un objecte.
- **rowCount**: emmagatzema el total de registres (filtrats si aplica) de la cerca, sense tenir en compte la paginació. Es fa servir sobretot de cara a saber el total de pàgines del llistat.
- **wrappedData**: Map al que es van desant els objectes obtinguts a partir de les cerques paginades. La clau que fa servir és l'identificador de l'objecte (PK a BBDD).
- **wrappedKeys**: Llistat amb els identificadors dels objectes obtinguts de la cerca paginada (PK a BBDD).
- mètode **walk**: Cridat cada vegada que es filtra, pagina, ordena o renderitza la taula a la vista. El mètode segueix la següent mecànica:

Llistat paginable a Servidor

- Fa un càlcul del rang de valors que ha d'obtenir sobre la cerca.
- Realitza la cerca a la BBDD mitjançant el servei definit per a aquest ús, *service*. En aquesta crida també s'informa la ordenació i filtres de valors a aplicar.
- Afegeix els objectes obtinguts a *wrappedData* i els identificadors d'aquests objectes a *wrappedKeys*.
- mètode **getRowCount**: Fa el càlcul del total de registres (filtrats si aplica) de la cerca, sense tenir en compte la paginació.
- mètode **getRowData**: A partir d'un identificador obté l'objecte.

Configuració de la Persistència

TbUsuariService.java

```
@Service("usuariService")
@Lazy
public class TbUsuariService {

    @Autowired
    private TbUsuariDAO dao;

    public int getRowCount(Map<String, String> filterValues) {
        return dao.getRowCount(filterValues);
    }

    public List<TbUsuaris> getItemsByrange(Integer startPk, int numberOfRows,
Map<String, String> filterValues, Map<String, SortOrder> sortOrders) {
        return dao.getItemsByrange(startPk, numberOfRows, filterValues, sortOrders);
    }

    public TbUsuaris getTbUsuariItemByPk(Integer pk) {
        return dao.getTbUsuariItemByPk(pk);
    }

    public boolean hasTbUsuariItemByPk(Integer pk) {
        return dao.hasTbUsuariItemByPk(pk);
    }
}
```

TbUsuariDao.java

```
public interface TbUsuariDAO extends GenericDAO<TbUsuaris, Integer> {

    public TbUsuaris getTbUsuariItemByPk(Integer pk);
    public boolean hasTbUsuariItemByPk(Integer pk);
    public int getRowCount(Map<String, String> filterValues);
    public List<TbUsuaris> getItemsByrange(Integer startPk, int numberOfRows,
Map<String, String> filterValues, Map<String, SortOrder> sortOrders);
}
```

Llistat paginable a Servidor

TbUsuariDaoImpl.java

```
public class TbUsuariDaoImpl extends JPAGenericDaoImpl<TbUsuaris, Integer> implements
TbUsuariDAO {

    @Override
    public TbUsuaris getTbUsuariItemByPk(Integer pk) {
        Session session = (Session) this.getEntityManager().getDelegate();
        Criteria crit = session.createCriteria(TbUsuaris.class);
        crit.add(Restrictions.eq("idUsuari", pk));
        return (TbUsuaris) crit.uniqueResult();
    }

    @Override
    public boolean hasTbUsuariItemByPk(Integer pk) {
        TbUsuaris usuari=this.getTbUsuariItemByPk(pk);
        return (usuari!=null);
    }

    @SuppressWarnings("unchecked")
    @Override
    public List<TbUsuaris> getItemsByrange(Integer startPk, int numberOfRows, Map<String,
String> filterValues, Map<String, SortOrder> sortOrders) {
        Session session = (Session) this.getEntityManager().getDelegate();
        Criteria crit = session.createCriteria(TbUsuaris.class);
        //Afegim filtres
        if (filterValues!=null && filterValues.size()>0){
            for (Map.Entry<String, String> entry : filterValues.entrySet()){
                crit.add(Restrictions.like(entry.getKey(), "%"+entry.getValue()+"%"));
            }
        }
        //Afegim la ordenació
        if (sortOrders!=null && sortOrders.size()>0){
            for (Map.Entry<String, SortOrder> entry : sortOrders.entrySet()){
                SortOrder so=entry.getValue();
                if (so.name().equals("ascending")){
                    crit.addOrder(Order.asc(entry.getKey()));
                }else if (so.name().equals("descending")){
                    crit.addOrder(Order.desc(entry.getKey()));
                }
            }
        }
        //Afegim el rang a consultar (paginació)
        crit.setFirstResult(startPk);
        crit.setMaxResults(numberOfRows);
        return (List<TbUsuaris>) crit.list();
    }

    @Override
    public int getRowCount(Map<String, String> filterValues) {
        Session session = (Session) this.getEntityManager().getDelegate();
        Criteria crit = session.createCriteria(TbUsuaris.class);
        //Afegim filtres donat es necessiten per al rowCount
        if (filterValues!=null && filterValues.size()>0){
            for (Map.Entry<String, String> entry : filterValues.entrySet()){
                crit.add(Restrictions.like(entry.getKey(), "%"+entry.getValue()+"%"));
            }
        }
        crit.setProjection(Projections.rowCount());
        Integer rowCount= (Integer) crit.uniqueResult();
        return Integer.valueOf(rowCount);
    }
}
```

A aquesta implementació del DAO s'han inclòs els mètodes que fa servir la paginació ordenada de l'exemple:

- Mètode **getTbUsuariItemByPk**: Donat un identificador, retorna l'objecte al que es refereix.

Llistat paginable a Servidor

- Mètode **hasTbUsuariltemByPk**: Donat un identificador, indica si l'objecte al que representa existeix o no a la BBDD.
- Mètode **getItemsByrange**: Fa la consulta paginada, ordenada i filtrada a la BBDD,
- Mètode **getRowCount**: Obté el número de registres de la cerca.

app-custom-persistence-jpa.xml

S'ha mantingut la configuració per defecte que es fixa al afegir el mòdul de persistència des de el plugin de Canigó 3 ,afegir-ne la referència al DAO del servei i configurar la transaccionabilitat a nivell de servei. Per a la generació d'aquest exemple la configuració ha quedat de la següent manera:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:jdbc="http://www.springframework.org/schema/jdbc"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
    http://www.springframework.org/schema/jdbc
    http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">
  <aop:aspectj-autoproxy/>
  <bean id="jpaVendorAdapter"
    class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
    <description>
      Fem servir Hibernate com a motor de persistència per sota de JPA.
    </description>
    <property name="showSql" value="${persistence.showSQL:true}" />
    <property name="generateDdl" value="${persistence.generateDdl:false}" />
    <property name="database" value="${persistence.database:HSQl}" />
    <property name="databasePlatform"
      value="${persistence.dialect:org.hibernate.dialect.HSQLDialect}" />
  </bean>

  <tx:advice id="txAdvice">
    <tx:attributes>
      <tx:method name="get*" propagation="REQUIRED" read-only="true" />
      <tx:method name="filter*" propagation="REQUIRED" read-only="true" />
      <tx:method name="find*" propagation="REQUIRED" read-only="true" />
      <tx:method name="load*" propagation="SUPPORTS" read-only="true" />
      <tx:method name="save*" propagation="REQUIRED" />
      <tx:method name="update*" propagation="REQUIRED" />
      <tx:method name="delete*" propagation="REQUIRED" />
      <tx:method name="insert*" propagation="REQUIRED" />
      <!--Control de mètode Afegit per al howto, per defecte no ho incorpora -->
      <tx:method name="has*" propagation="REQUIRED" read-only="true" />
    </tx:attributes>
  </tx:advice>

  <!-- Continua a la pàgina següent ... -->
```

Llistat paginable a Servidor

```
<!-- ... Continua des de la pàgina anterior -->

<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName" value="${jndi.name}"/>
  <property name="lookupOnStartup" value="${jndi.lookupOnStartup:true}"/>
  <property name="cache" value="${jndi.cache:true}"/>
  <property name="proxyInterface" value="javax.sql.DataSource"/>
</bean>

<aop:config >
  <aop:pointcut id="boOperation" expression="execution(*
cat.gencat.howtoserverpagination.service..*.*(..))" />
  <aop:advisor pointcut-ref="boOperation" advice-ref="txAdvice" />
</aop:config>

<bean id="usuariDAO"
class="cat.gencat.howtoserverpagination.dao.impl.TbUsuariDAOImpl"
parent="genericDAO"/>

</beans>
```

Configuracions al pom.xml

Només les requerides per al servei de persistència que s'afegeixen mitjançant el plugin de Canigó 3:

```
<canigo.persistence.jpa>[1.0.0,1.1.0)</canigo.persistence.jpa>

<dependency>
  <groupId>cat.gencat.ctti</groupId>
  <artifactId>canigo.persistence.jpa</artifactId>
  <version>${canigo.persistence.jpa}</version>
  <exclusions>
    <exclusion>
      <artifactId>commons-dbcp</artifactId>
      <groupId>commons-dbcp</groupId>
    </exclusion>
  </exclusions>
</dependency>
```