Bab 6

Membangun Aplikasi web CRUD

dengan Cimande

6.1 Tujuan

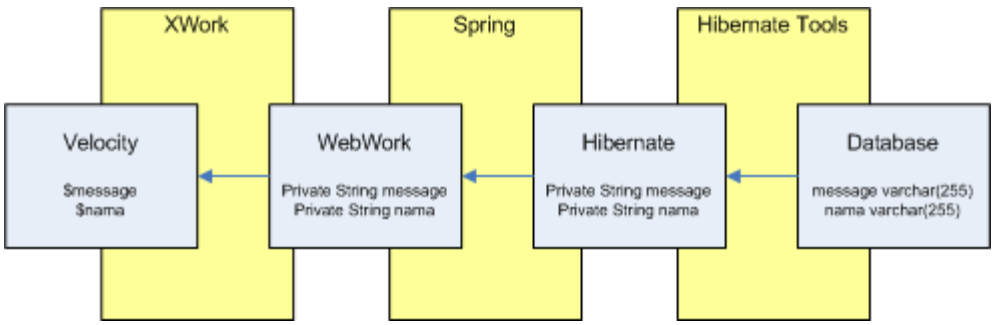
Dalam bab ini, kita akan mengimplementasikan teknologi Model-View-Controller secara lengkap dengan menggunakan project Cimande.

Pada akhir bab ini, pelajar diharapkan dapat:

- Mengenal project Cimande
- Memahami prinsip kerja Model-View-Controll
- Mengimplementasikan CRUD dalam project Cimande

6.2 Pengenalan Cimande

Modul ini akan menjelaskan bagaimana membuat sebuah aplikasi dengan implementasi CRUD (Create, Read, Update and Delete) yang lebih umum dapat dikatakan membuat aplikasi untuk membuat content baru, melakukan search/pencarian, melakukan pengeditan data, serta menghapus data.Adapun mapping standar setiap C, R, U dan D adalah dengan melakukan data flow mapping dari database ke Velocity sesuai dengan gambar berikut.



Arsitektur Projek Cimande dari BlueOxygen

Dimana metode ini adalah mengacu pada projek Cimande Thin dari BlueOxygen. Cimande Thin

adalah contoh kecil dari fondasi solusi, dimana Cimande Thin adalah framework integrasi antara Velocity/JSP, WebWork dengan Hibernate, hubungan WebWork dengan Hibernate menggunakan Spring. Pemetaan WebWork dengan Velocity diurus oleh XWork. Untuk menghubungkan database dengan hibernate dapat menggunakan Hibernate Tools yang terintegrasi dengan Ant atau menggunakan Hibernate Tools yang merupakan bagian dari JbossIDE. Dimana, mekanisme Hibernate dengan database telah dibahas pada bab 4.

Jadi dapat dikatakan bab ini adalah merupakan skema integrasi dari teknologi MVC yang telah dibahas dibab sebelumnya, menjadi satu kesatuan, yang disebut Cimande.

6.3 Sekilas tentang projek Cimande

Bagian ini akan menjelaskan bagimana mengembangkan sebuah aplikasi berbasis MVC, yang menggunakan Velocity sebagai presentation layer, WebWork sebagai controller, serta Hibernate sebagai Modelnya. Sedangkan integrasi antara WebWork dengan Hibernate menggunakan Spring telah disediakan oleh Cimande, termasuk juga integrasi WebWork dengan Velocity telah diintegrasi oleh Xwork, yang mana contoh pengembangan WebWork dengan Velocity telah dijelaskan pada bab 2.

Untuk membuat sebuah aplikasi MVC, sebenarnya sama dengan yang dijelaskan pada bab 2, tetapi yang membedakan adalah adanya implementasi PersistenceAware, yang merupakan sebuah mapping insialisasi objek yang lebih populer disebut dengan IoC, atau singkatan dari Injection of Control.

Mekanisme ini sebenarnya adalah sebuah cara sehingga kita tidak perlu melakukan inisialisasi dalam membuat session untuk Hibernate. File yang mengurusi semua ini adalah file ApplicationContext.xml, yang merupakan mapper pada Spring.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans >

        <bean id="sessionFactory"
class="org.blueoxygen.cimande.persistence.hibernate.DefaultHibernateSessionFactory"
        init-method="init"
        destroy-method="destroy"
        singleton="true"/>

   <bean      id="persistenceManager"
class="org.blueoxygen.cimande.persistence.hibernate.HibernatePersistenceManager"
                init-method="init"
                destroy-method="dispose"
                singleton="false">
        <property name="hibernateSessionFactory">
        <ref bean="sessionFactory"/>
        </property>
        </bean>

</beans>
```

XML diatas merupakan implementasi dari SpringBean, dimana pada implementasinya setiap

objek yang hendak terhubung dengan SessionFactory dari Hibernate, harus mengimplementasikan PersistenceManager.

Bilamana dibab sebelumnya dibahas mengenai bagaimana mengimplementasikan Action pada WebWork, ini artinya secara otomatis WebWork akan terhubung dengan tipe result, baik itu Velocity, JSP, Jasper ataupun JfreeChart. Sedangkan untuk menghubungkan antara WebWork dengan Hibernate diperlukan implementasi PersistenceAware.

Berikut adalah contohnya:

```
public class DescriptorForm extends ActionSupport implements PersistenceAware
```

Kemudian dalam setiap implementasinya harus mengimplementasi PersistenceManager, seperti berikut

```
public void setPersistenceManager(PersistenceManager persistenceManager) {
        this.manager = persistenceManager;
}
```

6.4 Logika CRUD mengacu pada skema Cimande

Dalam implementasi dengan MVC, terutama menggunakan WebWork, selalu digunakanlah proses CRUD cycle, dan tentu saja dalam implementasi dalam pengembangannya sebaiknya setiap modul yang mengimplementasikan CRUD dibuat matrixnya seperti berikut

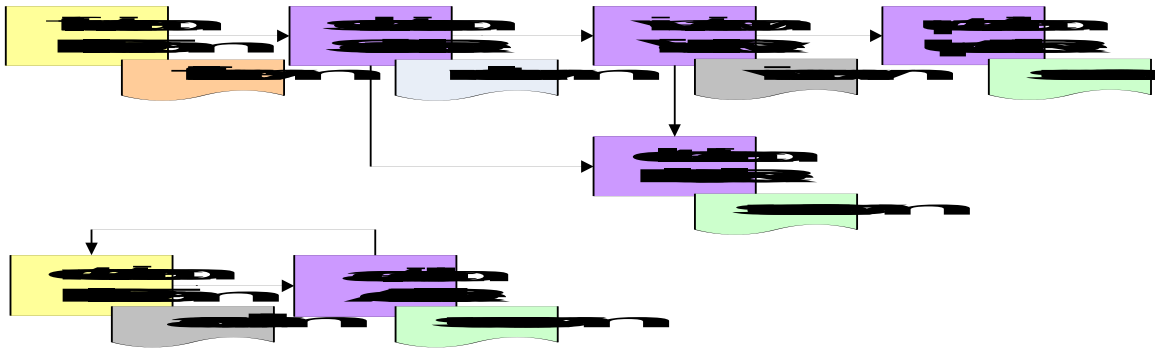
Module	C	R	U	D	J
PageCollection	X	X	X	X	
Business Partner	X	X	X	X	
Contact	X	X	X	X	
Purchase Order	X	X	X		X

Matrix CRUD, umumnya ditambahkan J, artinya Jasper, ini adalah report untuk setiap transaksi CRUD.

Adapun beberapa mapping standard pengembangan CRUD yang dilakukan didalam project BlueOxygen adalah sebagai berikut:

	Naming	Action
C	New	new.action
R	Search	filter.action
U	Edit	edit.action
D	Delete	delete.action

Ada tambahan satu action yaitu view.action yang mana merupakan result dari filter.action. CRUD dan Action mapping bilamana diimplementasikan dengan presentation layer adalah sebagai berikut:



Alur CRUD dan template yang dipakai bersama.

Dalam pengembangan aplikasi CRUD menggunakan Cimande, sebenarnya template yang diperlukan tidak banyak, yaitu filter.vm, result.vm, view.vm, success.vm, dan add.vm. Kedepannya view.vm dan add.vm dapat digabung.

Berikut adalah diagram penghubung untuk kasus pengembangan role dengan MVC.

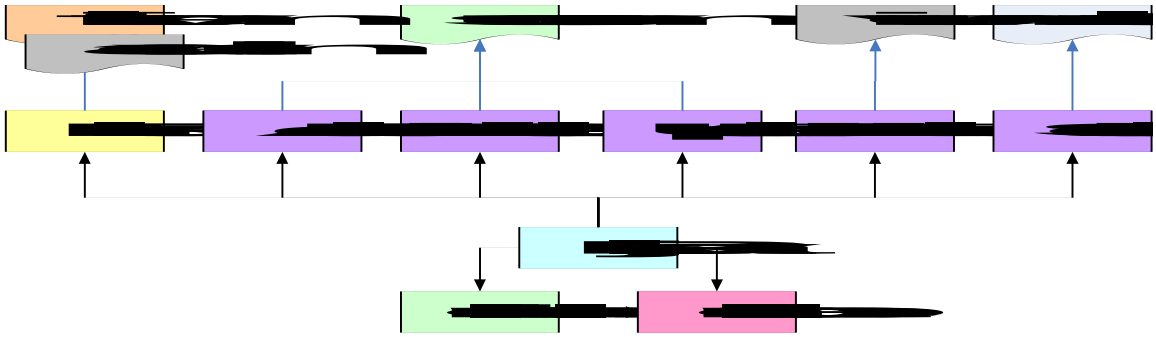


Diagram Implementasi Role antara Hibernate dengan Velocity menggunakan WebWork

Diagram diatas menjelaskan untuk berhubungan dengan table role, diperlukan sebuah file Role.java, dan bilamana mengimplementasikan Hibernate yang core bukan anonasi, diperlukan role.hbm.xml, bilamana menggunakan anonasi, hanya perlu mengimplementasikan JPA.

Sedangkan file RoleForm adalah file yang mengimplementasikan PersistenceAware, sedangkan file yang lain yaitu AddRole, DeleteRole, UpdateRole, ViewRole atau SearchRole semuanya merupakan turunan dari RoleForm.

Yang mana kita hanya perlu mereplace execute dengan kegiatan yang kita inginkan.

Sebagai contoh adalah untuk AddRole, isi dari execute adalah

```
public String execute() {
        Role role = new Role();

        if (getName().equalsIgnoreCase(""))
                addActionError("Name can't be empty.");
        if (getDescription().equalsIgnoreCase(""))
                addActionError("Description can't be empty.");

        if (hasErrors()) {
                return INPUT;
         } else {
```

```
                role.setName(getName());
                role.setDescription(getDescription());

                // logging information
                LogInformation log = new LogInformation();

                if (sessionCredentials.getCurrentUser() != null ) {

                        log.setCreateBy(sessionCredentials
                                .getCurrentUser()
                                .getId());
                        log.setLastUpdateBy(sessionCredentials
                                .getCurrentUser()
                                .getId());

                }

                log.setCreateDate(new
                        Timestamp(System.currentTimeMillis()));
                log.setLastUpdateDate(new
                        Timestamp(System.currentTimeMillis()));
                log.setActiveFlag(getActiveFlag());

                if (getActiveFlag() == -1) {
                        log.setActiveFlag(LogInformation.INACTIVE);
                } else {
                        log.setActiveFlag(getActiveFlag());
                }

                role.setLogInformation(log);
                pm.save(role);
                return SUCCESS;
        }
}
```

AddRole diatas kalau dilihat lebih dekat terdiri dari 3 area pengembangan, yang mana area pertama adalah validasi, area berikutnya adalah memasukan informasi user yang aktif ke dalam objek role, kemudian melakukan penyimpanan ke database.

Mekanisme pertama yaitu validasi, sebenarnya dapat diganti dengan validasi yang ada di WebWork dengan melakukan implementasi interceptor validasi pada setiap package action yang hendak dilakukan validasi.

Sedangkan ViewRole adalah sebuah action untuk menampilkan isi dari sebuah record, untuk dilakukan proses update.

ViewRole dan SearchRole merupakan sebuah kegiatan yang melempar data dari Hibenate ke Velocity/JSP, sedangkan yang lainnya hanyalah sebuah kegiatan untuk berinteraksi dengan Hiberante.

Adapun code execute ViewRole adalah sebagai berikut:

```
public String execute() {
        role = (Role) pm.getById(Role.class, id);
```

```
            if (role == null) {
                addActionError("Such role couldn't be found");
                return ERROR;
            } else {
                return SUCCESS;
            }
        }
```

Sedangkan untuk SearchRole yang digunakan untuk filter.action adalah.

```
        public String execute() {

            try {
                sess = hsf.createSession();
                Criteria crit = sess.createCriteria(Role.class);
                if (!getName().equalsIgnoreCase("")) {
                    crit.add(Expression.like("name", "%" +
                    getName() + "%"));
                }
                if (!getDescription().equalsIgnoreCase("")) {
                    crit.add(Expression.like("description", "%" +
                    getDescription()
                                + "%"));
                }
                if (getActiveFlag() != -1) {
                crit.add(Expression.eq("logInformation.activeFlag",
                    new Integer(getActiveFlag())));
                }
                roles = crit.list();
                hsf.endSession(sess);
                return SUCCESS;
            } catch (HibernateException e) {
                return ERROR;
            } catch (SQLException e) {
                return ERROR;
            } finally {
                try {
                    hsf.closeSession(sess);
                } catch (HibernateException e1) {
                    return ERROR;
                } catch (SQLException e1) {
                    return ERROR;
                }
            }
        }
```

Adapun isi dari xwork.xml adalah sebagai berikut:

Untuk membuat form isian atau create.action

```
    <action name="create" class="org.blueoxygen.cimande.role.actions.RoleForm" >
       <result name="success" type="velocity">add.vm</result>
    </action>
```

Untuk mengisi setiap hasil submit dari create.action

```
<action name="add" class="org.blueoxygen.cimande.role.actions.AddRole">
  <result name="success" type="velocity">addSuccess.vm</result>
  <result name="input" type="velocity">add.vm</result>
</action>
```

Untuk menampilkan form search atau filter.action. Perhatikan perbedaannya dengan create.action, objeknya sama, tetapi templatenya berbeda.

```
<action name="filter" class="org.blueoxygen.cimande.role.actions.RoleForm">
   <result name="success" type="velocity">filter.vm</result>
   <result name="error">/errors/errors.vm</result>
</action>
```

Untuk menampilkan hasil pemcarian

```
        <action name="search"
class="org.blueoxygen.cimande.role.actions.SearchRole">
   <result name="success" type="velocity">result.vm</result>
   <result name="error">/errors/errors.vm</result>
</action>
```

Untuk menampilkan sebuah record yang terpilih:

```
<action name="edit" class="org.blueoxygen.cimande.role.actions.EditRoleLoad">
   <result name="success" type="velocity">edit.vm</result>
   <result name="error">/errors/errors.vm</result>
</action>
```

Untuk mengupdate perubahaan terhadap data yang telah ditampilkan atau setelah submit dari edit.action

```
<action name="update" class="org.blueoxygen.cimande.role.actions.EditRole">
   <result name="success" type="redirect">view.action?id=${id}</result>
   <result name="input" type="velocity">edit.vm</result>
   <result name="error">/errors/errors.vm</result>
</action>
```

Untuk menghapus record (dapat diakses langsung saat result dari pencarian)

```
<action name="confirmDelete" class="org.blueoxygen.cimande.role.actions.ViewRole">
   <result name="success" type="velocity">confirmDelete.vm</result>
   <result name="error">/errors/notfound.vm</result>
</action>
<action name="delete" class="org.blueoxygen.cimande.role.actions.DeleteRole">
   <result name="success" type="velocity">deleteSuccess.vm</result>
   <result name="error">/errors/notfound.vm</result>
</action>
```

Dari semua ini sebenarnya ada satu link flow yang hilang, yaitu bagaimana sebuah create.action dapat mengeksekusi add.action, dan filter.action dapat menghasilkan search.action, demikian juga dengan edit.action ke update.action, atau confirmDelete.action ke delete.action.

Flow ini sebenarnya terdapat pada form template, yang mana pada kasus diatas menggunakan velocity semuanya.

Coba buka file add.vm, ini adalah file yang digunakan saat melakukan create.action, ada dua area yang dapat diperhatikan yaitu menampilkan error validasi dan form isian.

Berikut ini adalah cara menampilkan validasi.

```
#if (!$actionErrors.isEmpty())
<div class="errorMessage">Errors</div>
<ul class="errorMessage">
#foreach( $error in $actionErrors )
<li>$error</li>
 #end
</ul>
#end
```

Sedangkan berikut ini adalah form untuk melakukan action berikutnya,

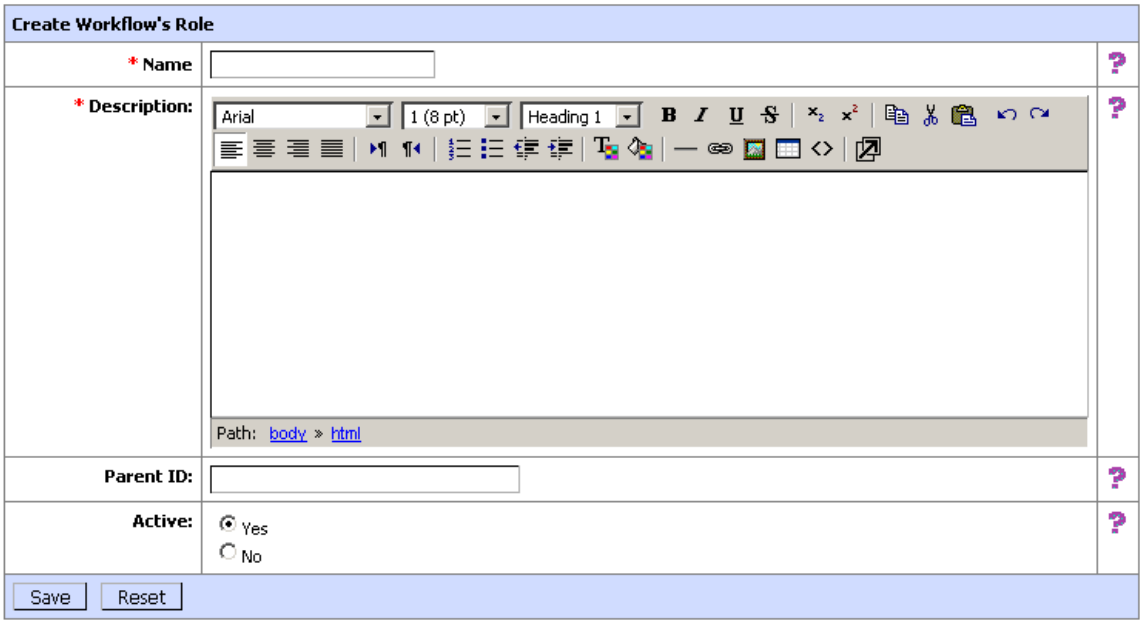
```
<form method="post" action="add.action">
<table border="0" cellspacing="5" cellpadding="5">
  <tr id="tableHeader1">
    <td colspan="2">Role</td>
  </tr>
  <tr>
    <td>Name</td>
    <td><input type="text" name="name" value="$!name">
    </td>
  </tr>
...
</table>
</form>
```

Perhatikan baris <form method="post" action="add.action">, ini adalah link flow yang hilang, form ini artinya bilamana tombol submit diakses dilanjutkan dengan add.action.

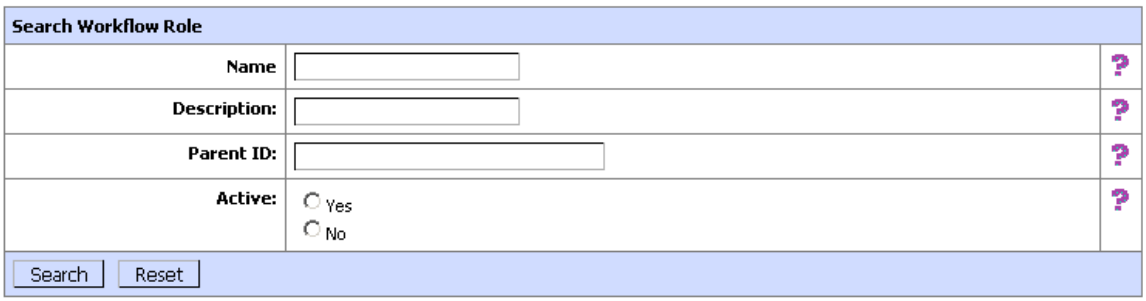
Sedangkan form untuk edit.vm akan berisikan update.action, sedangkan filter.vm yang digunakan oleh filter.action akan memiliki instruksi <form method="post" action="search.action">.

Ini sesuai dengan diagram alur flow yang telah dijelaskan diawal bab ini bukan.

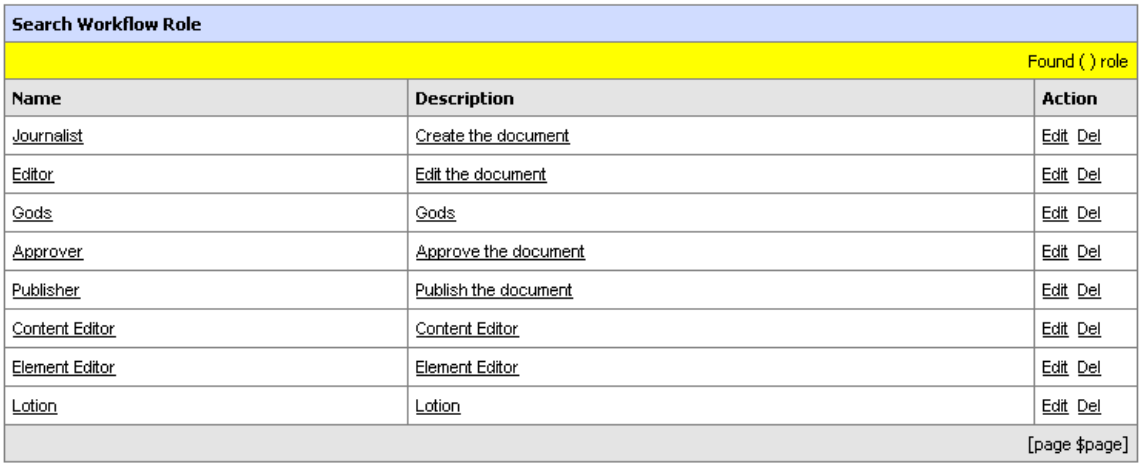
Berikut adalah bentuk hasil dari new.action



Berikut adalah bentuk hasil dari implementasi search.action.

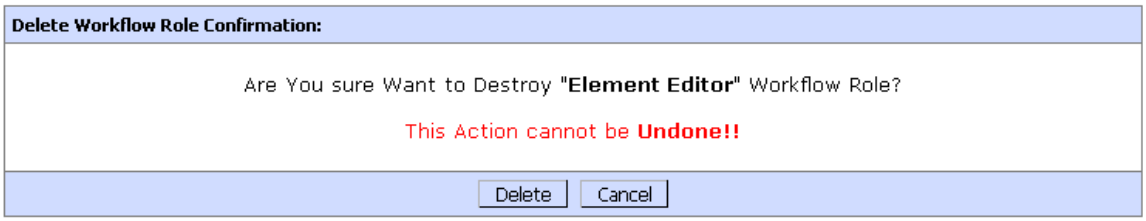


Sedangkan bilamana tombol Search ditekan, akan masuk ke result.action yang merupakan sebuah hasil dari pencarian.



Setiap hasil dari pencarian ada dua link yang dapat dilakukan mengikuti flow CRUDnya Cimande yaitu Edit untuk ke view.action dan Del untuk ke delete.action.

Bilamana Edit ditekan akan keluar form seperti create.action, tetapi tentu saja akan menampilkan isi dari link yang dipilih, sedangkan bilamana Del dipilih, akan muncul page konfirmasi yang akan memberikan konfirmasi untuk penghapusan. Konfirmasi ini dapat disebut sebagai delete.action.



Proses CRUD diatas adalah fondasi yang selalu dilakukan oleh tim penulis untuk mengerjakan sebuah projek berbasis Web menggunakan Cimande.