


Conception en UML, Architecture n-tiers, par l'exemple

Utilisation de php 5, Mysql, Html, css, ...

Inspiré de UML2 par la pratique

M. Blay-Fornarino

Les codes sont disponibles sur le site web

Bibliographie

- «Why MVC is not an application architecture» Stefan Priebsch, the PHP.cc ZendCon 2010
- Developing Web Applications with PHP, RAD for the World Wide Web,

Plan

- ✓ Rappels sur le «pattern» Observer
- ✓ Approche générale
- ✓ MVC
- ✓ DAO
- ✓ Des classes aux bases de données

«Pattern Observer» : le problème

- ✓ Problème : Mettre en oeuvre une relation de «un vers plusieurs» objets afin que plusieurs objets puissent être notifiés du changement d'état d'un objet et puisse réagir.

Il est très utilisé en IHM, mais peut être appliqué dans bien d'autres cas.

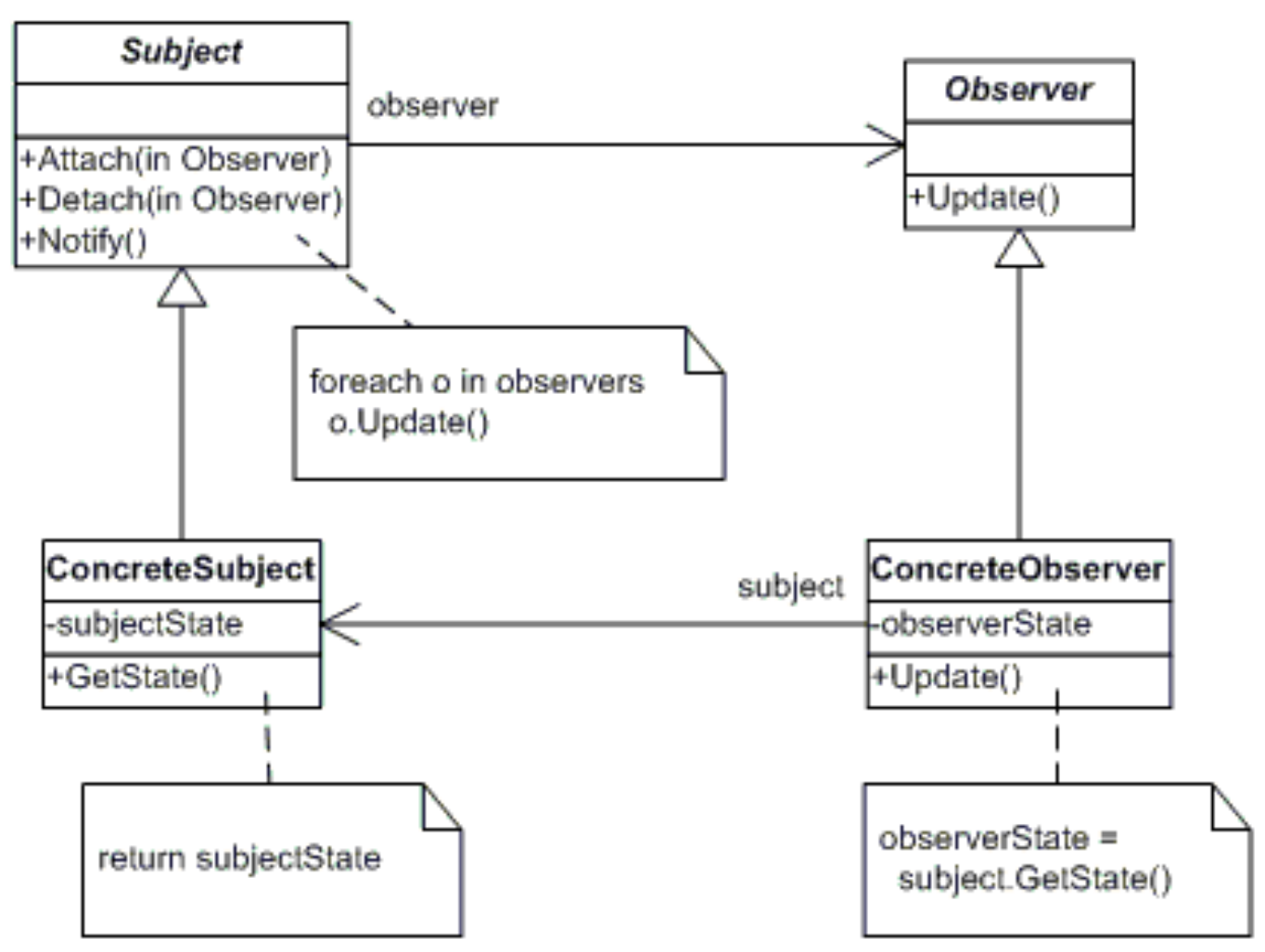
«Pattern Observer» : les rôles

- ✓ Rôles : Un sujet et des «observers» (le sujet doit devenir «observable»)
- ✓ Responsabilités :
  - ➡ Le sujet :
    - ▸ notifie les observeurs quand il «change»
    - ▸ permet aux observeurs de s'(de-)enregistrer.
  - ➡ Les observeurs
    - ▸ acceptent les notifications

Pattern Observer : Solution

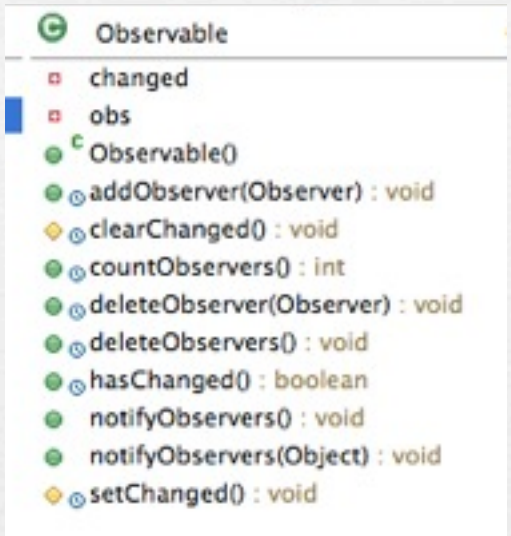
- **Sujet Abstrait** (Observable) :
  - gère les observeurs (AddObserver(Observer))
  - notifie les observeurs (notifyObservers)
- **Sujet (Observable) :**
  - A chaque changement d'état, il «notifie» les observeurs (Notify)
  - Il peut donner son état (getState)
- **Observeur**
  - Se met à jour quand il est notifié (update)
- **Observeur (Abstrait)**
  - Peut être notifié (update)

Pattern Observer : Solution

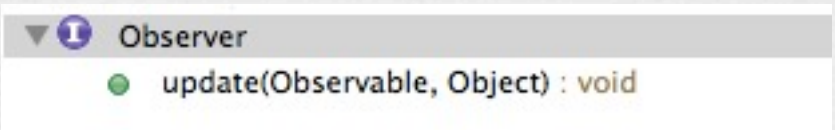


«Pattern Observer» en java

- ✓ Le sujet abstrait : classe abstraite **java.util.observable**
- ✓ Le sujet concret :
  - ➡ Votre classe qui hérite de observable
  - ➡ *C'est à vous d'appeler* **notifyObservers**



- ✓ L'observeur abstrait : Interface java.util.Observer
- ✓ L'observeur concret :
  - ➡ Votre classe qui «implémente» Observer
  - ➡ *doit implémenter* **update**



«Pattern Observer» en java exemple

```
import java.util.Observable;
public class ObservableObject extends Observable
{
   private int n = 0;
   public ObservableObject(int n)
   {
      this.n = n;
   }
   public void setValue(int n)
   {
      this.n = n;
      setChanged();
      notifyObservers();
   }
   public int getValue()
   {
      return n;
   }
}
```

http://www.javaworld.com/article/2077258/learn-java/observer-and-observable.html

«Pattern Observer» en java exemple

```
    import java.util.Observer;
import java.util.Observable;
public class TextObserver implements Observer
{
   private ObservableObject ov = null;
   public TextObserver(ObservableObject ov)
   {
      this.ov = ov;
   }
   public void update(Observable obs, Object obj)
   {
      if (obs == ov)
      {
         System.out.println(ov.getValue());
      }
   }
}
```

http://www.javaworld.com/article/2077258/learn-java/observer-and-observable.html

«Pattern Observer» en java exemple

```
public class Main
{
   public Main()
   {
      ObservableValue ov = new ObservableValue(0);
      TextObserver to = new TextObserver(ov);
      ov.addObserver(to);
   }
   public static void main(String [] args)
   {
      Main m = new Main();
   }
}
```

http://www.javaworld.com/article/2077258/learn-java/observer-and-observable.html

«Pattern Observer» en action

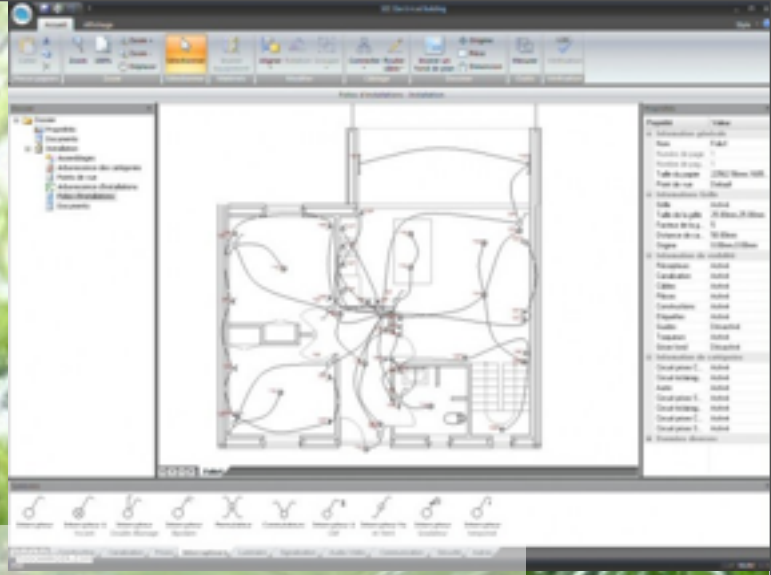
- ✓ Un forum
  - ➡ On peut poster des messages dans le forum : un message à un titre.
- ✓ Des Abonnés
  - ➡ Un abonné peut recevoir des messages dans ses boites de messages.
- ✓ Dès qu'un message est posté sur le forum, tous les abonnés sont notifiés.
  - ➡ Certains abonnés enregistrent le message dans leur boite.
  - ➡ (2) Certains abonnés n'enregistrent que les messages dont le titre contient «IUT» ;-)


Approche

Analyse

L'analyse produit :

- Au moins
  - Les uses cases de plus haut niveau
  - les premières procédure de tests de validation
  - Des diagrammes de séquences de niveau analyse
  - Un diagramme de classes capturant les grandes lignes du domaine
  - un glossaire initial



Conception

vise à produire

-> Architecture

-> Classes

-> Données


Conception
Focus
Architecture

Choix d'Architecture

Présentation

Logique applicative

Gérer les informations

Stockage

```
CREATE TABLE `information` (
 `titre` varchar(20) NOT NULL,
 `date` varchar(22) NOT NULL,
`identifiant` int(11) NOT NULL auto_increment,
 PRIMARY KEY  (`identifiant`))
```

Choix d'Architecture

Présentation

Logique applicative

Gérer les informations

Stockage

```
CREATE TABLE `information` (
 `titre` varchar(20) NOT NULL,
 `date` varchar(22) NOT NULL,
`identifiant` int(11) NOT NULL auto_increment,
 PRIMARY KEY  (`identifiant`))
```

Choix d'Architecture

Présentation

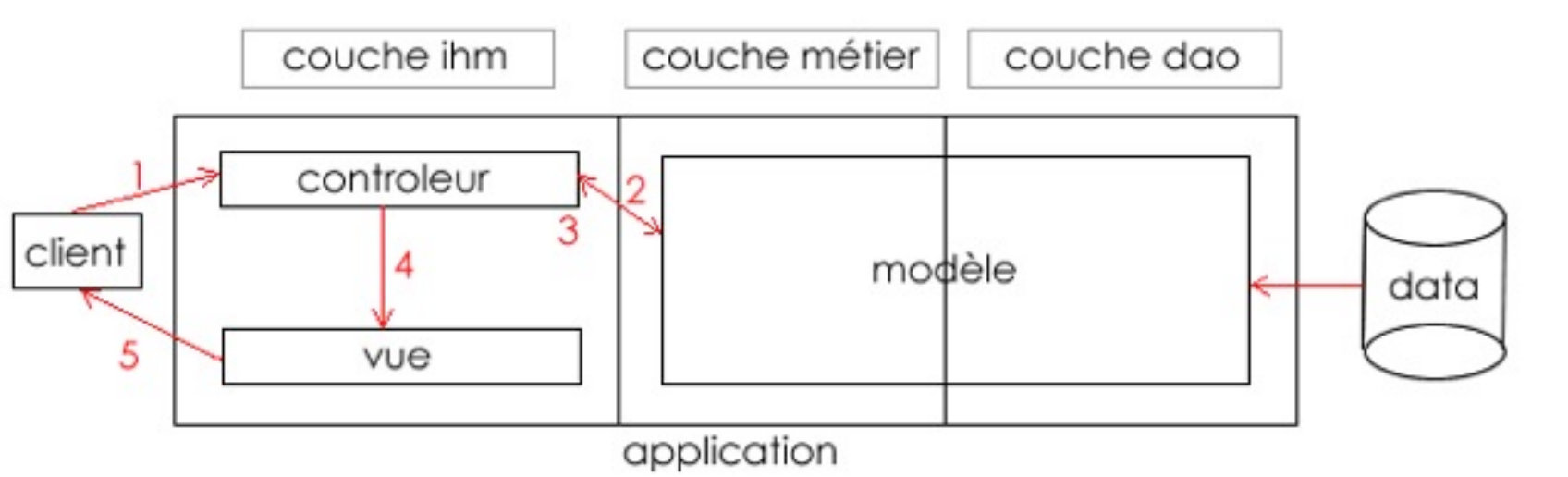
Logique applicative

Gérer les informations

Stockage

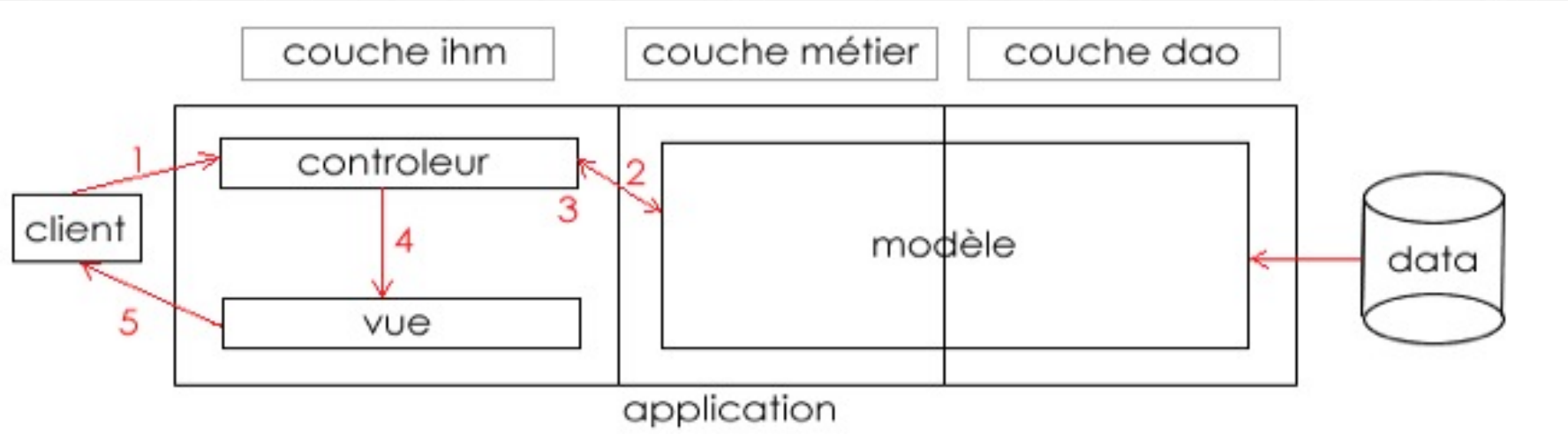
```
CREATE TABLE `information` (
 `titre` varchar(20) NOT NULL,
 `date` varchar(22) NOT NULL,
`identifiant` int(11) NOT NULL auto_increment,
 PRIMARY KEY  (`identifiant`))
```

Couches logicielles

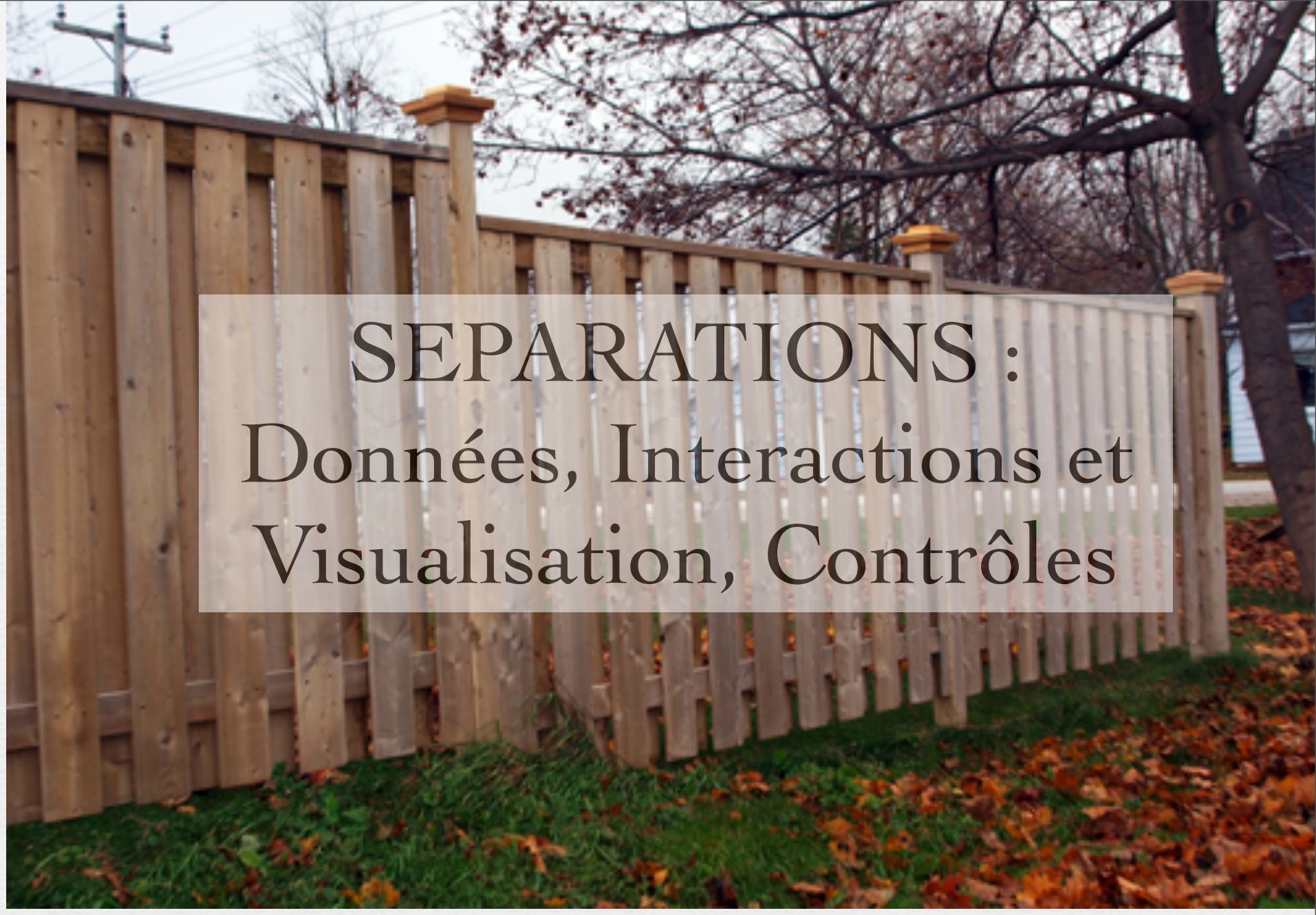


- **couche ihm**: c'est l'interface utilisateur encore appelé interface homme machine
- **couche métier** : c'est le coeur de l'application où réside les objets traités par l'application
- **couche dao** : couche d'accès aux données (data access object). Cette couche permet une indépendance de la logique métier et du stockage des données associées

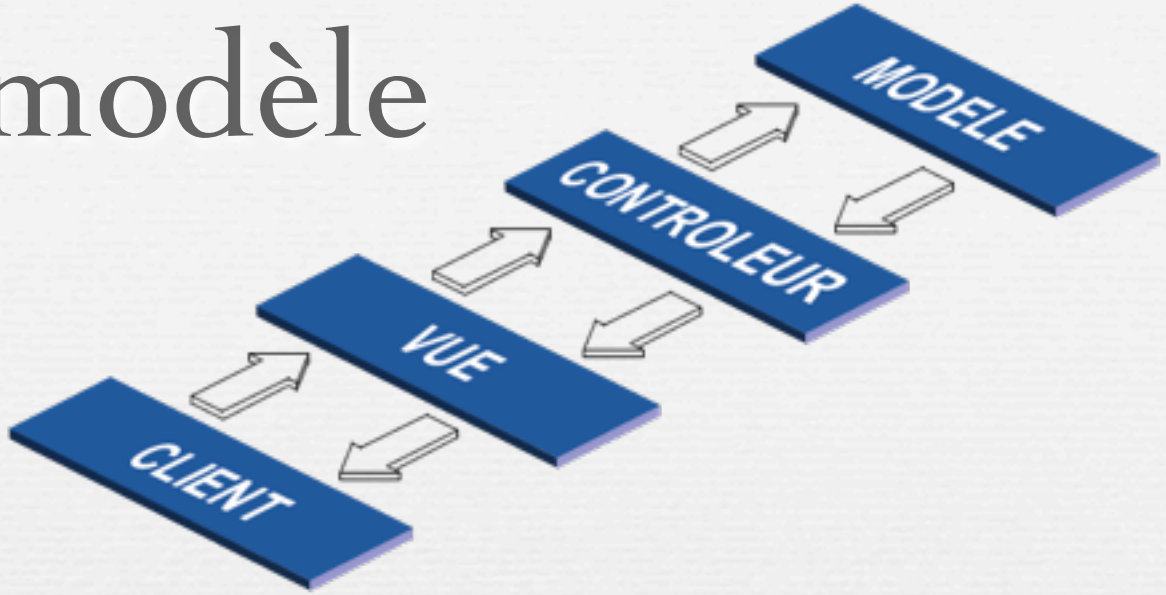
Couches logicielles



1. le client fait une demande au contrôleur. Ce contrôleur voit passer toutes les demandes des clients.
2. le contrôleur doit traiter la demande. Pour ce faire, il peut avoir besoin de la couche métier, cette dernière peut éventuellement accéder aux données (via la couche dao)
3. le contrôleur effectue les traitements nécessaires sur / avec les objets renvoyés par la couche métier
4. le contrôleur sélectionne et nourrit la (les) vue(s) pour présenter les résultats du traitement qui vient d'être effectuée
5. la vue est enfin envoyée au client par le contrôleur

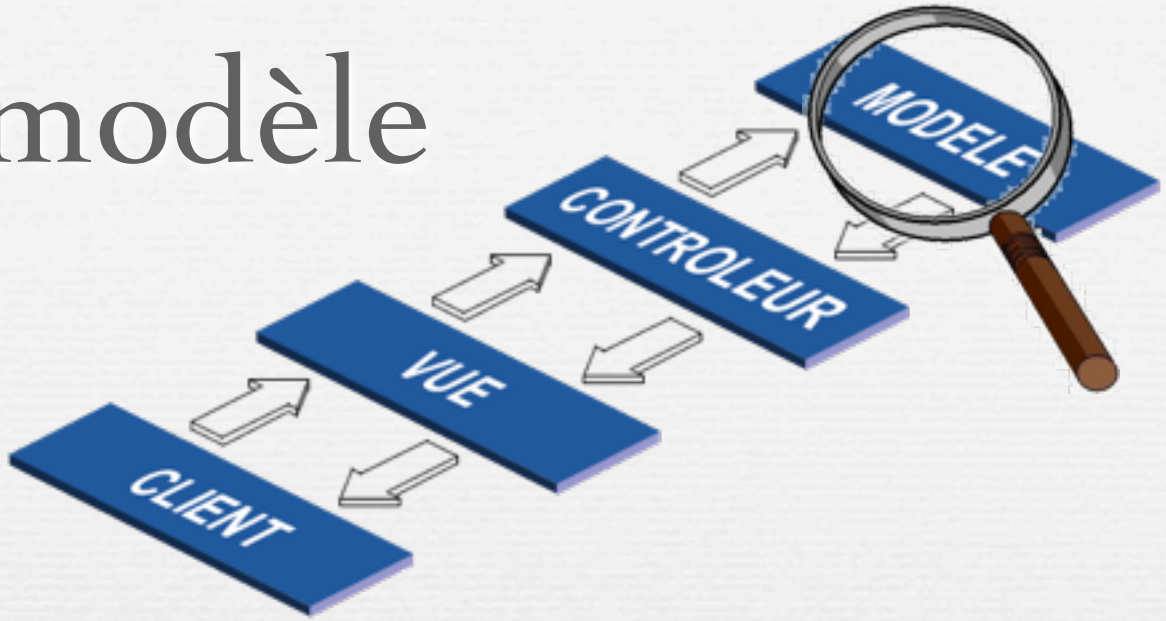

SEPARATIONS :
Données, Interactions et
Visualisation, Contrôles

Le modèle



http://blog.mazenod.fr/2010/01/design-pattern-mvc-zoom-sur-la-couche-modele-dal-dao-orm-crud/

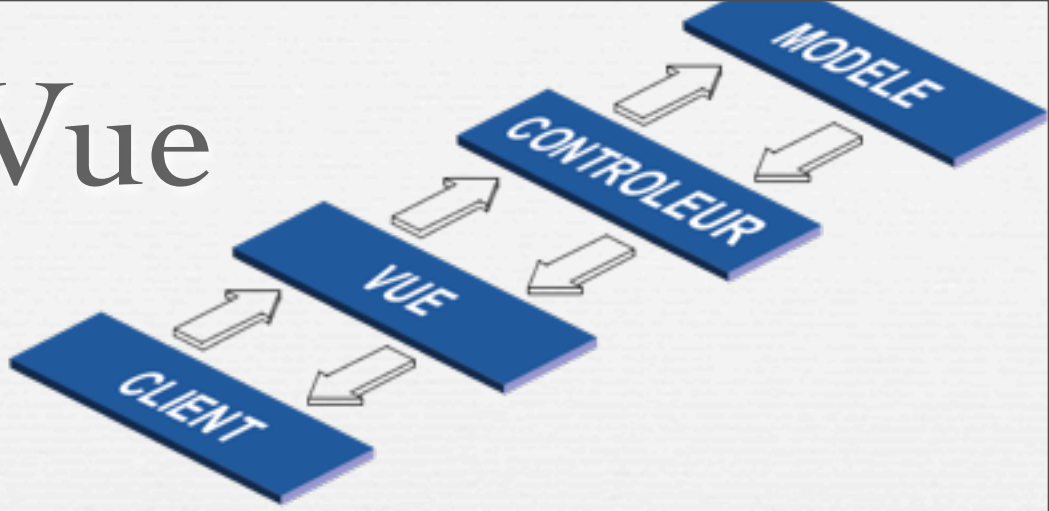
Le modèle



- ✓ décrit et contient les données manipulées par l'application, ainsi que des traitements propres à ces données
- ✓ les résultats renvoyés par le modèle sont dénués de toute présentation
- ✓ le modèle contient la logique métier de l'application

http://blog.mazenod.fr/2010/01/design-pattern-mvc-zoom-sur-la-couche-modele-dal-dao-orm-crud/

Vue



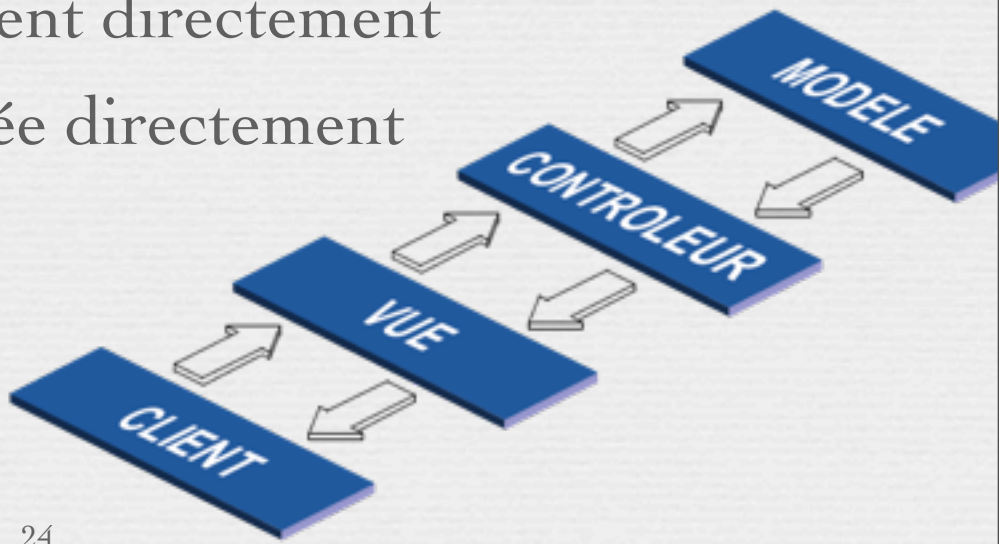
- ✓ Interface avec laquelle l'utilisateur interagit
  - ➡ reçoit toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, soumission de formulaire ...)
  - ➡ envoie les événements au contrôleur
- ✓ Présentation des résultats renvoyés par la couche modèle, après le traitement du contrôleur
- ✓ La vue n'effectue aucun traitement

Vue

- ✓ Interface avec laquelle l'utilisateur interagit
  - ➡ reçoit toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, soumission de formulaire ...)
  - ➡ envoie les événements au contrôleur
- ✓ Présentation des résultats renvoyés par la couche modèle, après le traitement du contrôleur
- ✓ La vue n'effectue aucun traitement

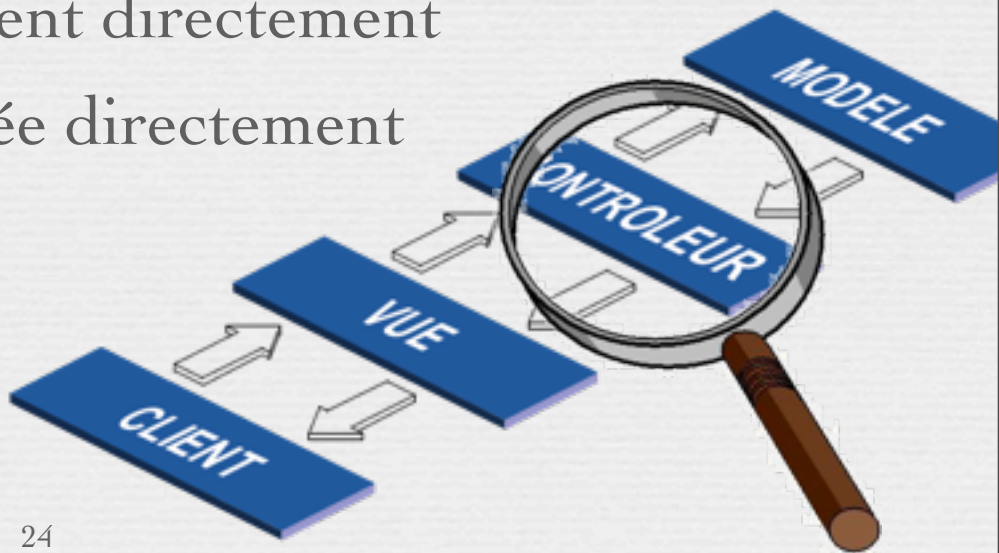
Contrôleur

- Gestion des événements de synchronisation entre modèle et vue
- Détermine l'action à réaliser
  - Si une action nécessite un changement des données
    - demande la modification des données au modèle
- Ne fait qu'appeler des méthodes
  - n'effectue aucun traitement directement
  - ne modifie aucune donnée directement

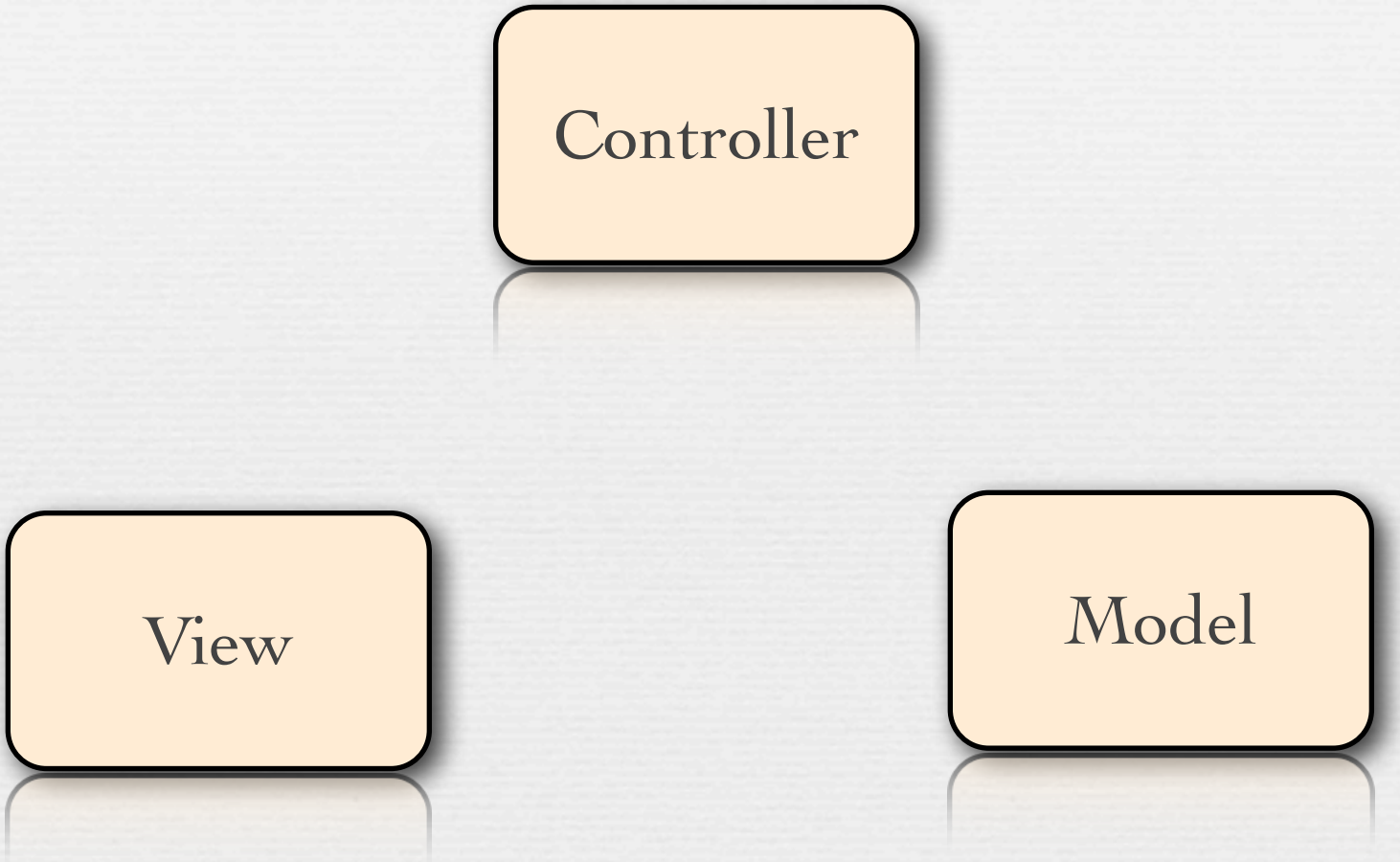


Contrôleur

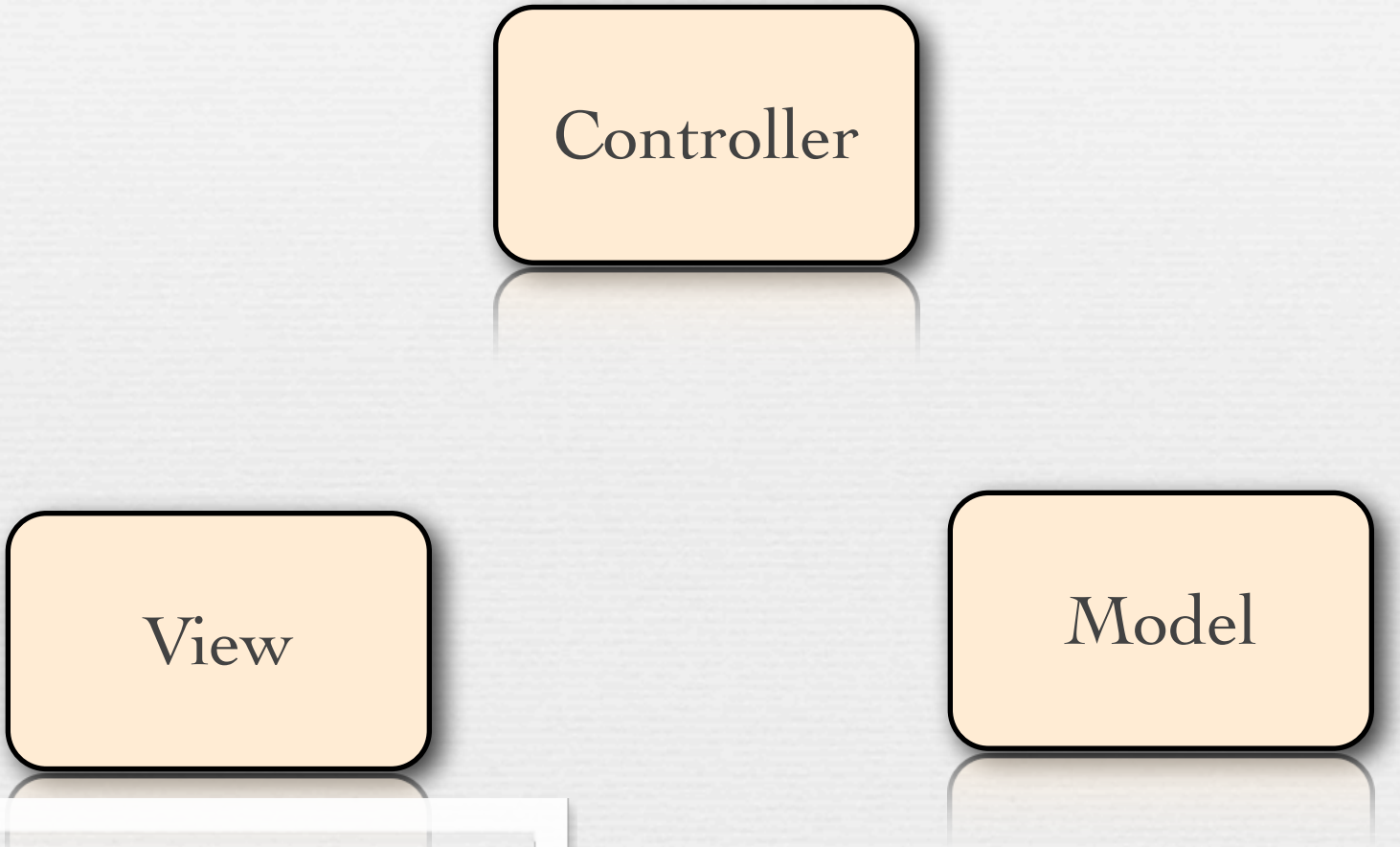
- Gestion des événements de synchronisation entre modèle et vue
- Détermine l'action à réaliser
  - Si une action nécessite un changement des données
    - demande la modification des données au modèle
- Ne fait qu'appeler des méthodes
  - n'effectue aucun traitement directement
  - ne modifie aucune donnée directement



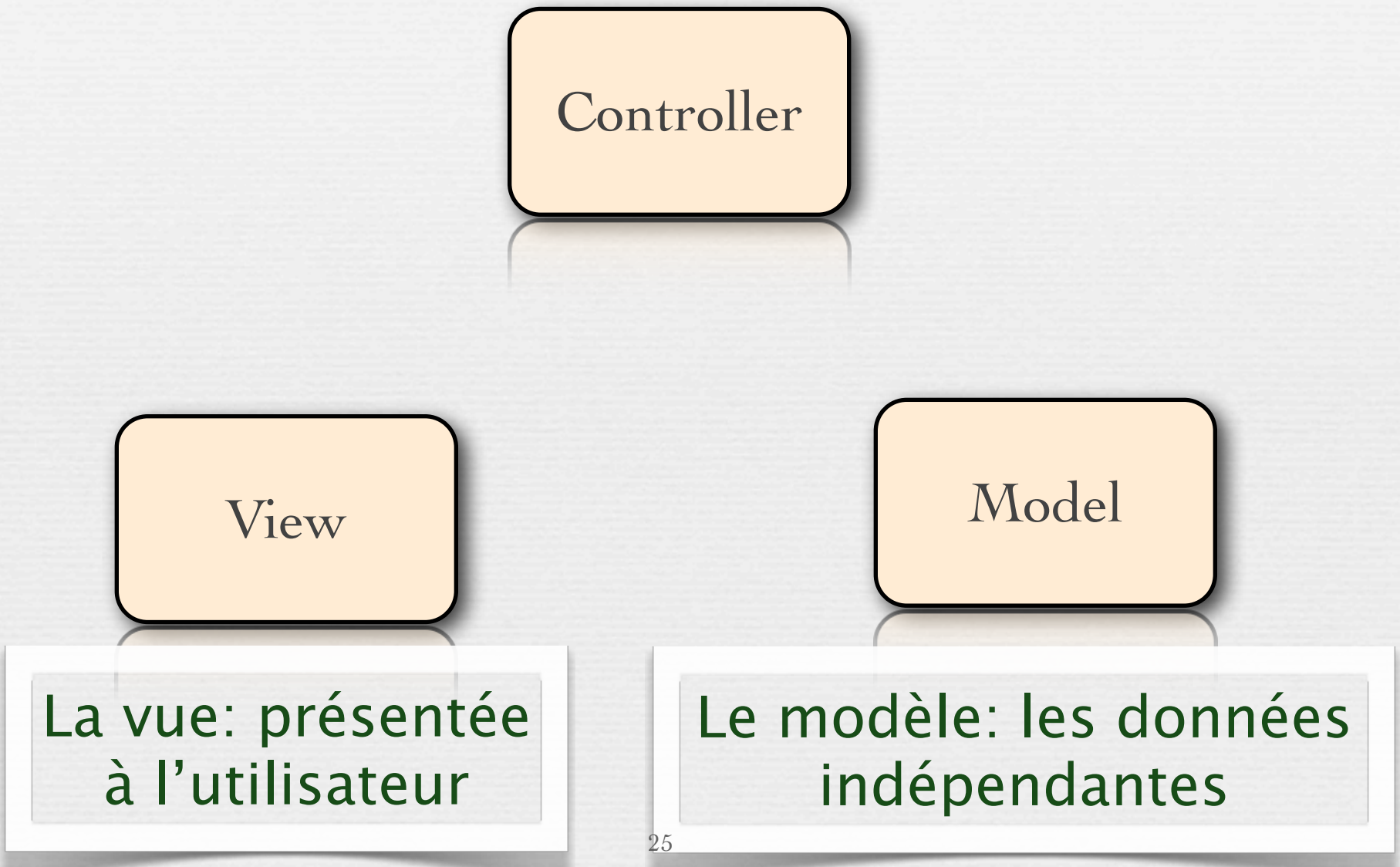
Modèle-Vue-Controleur (MVC)



Modèle-Vue-Controleur (MVC)



Modèle-Vue-Controleur (MVC)



Modèle-Vue-Controleur (MVC)

Controller

contrôleur: chef d'orchestre

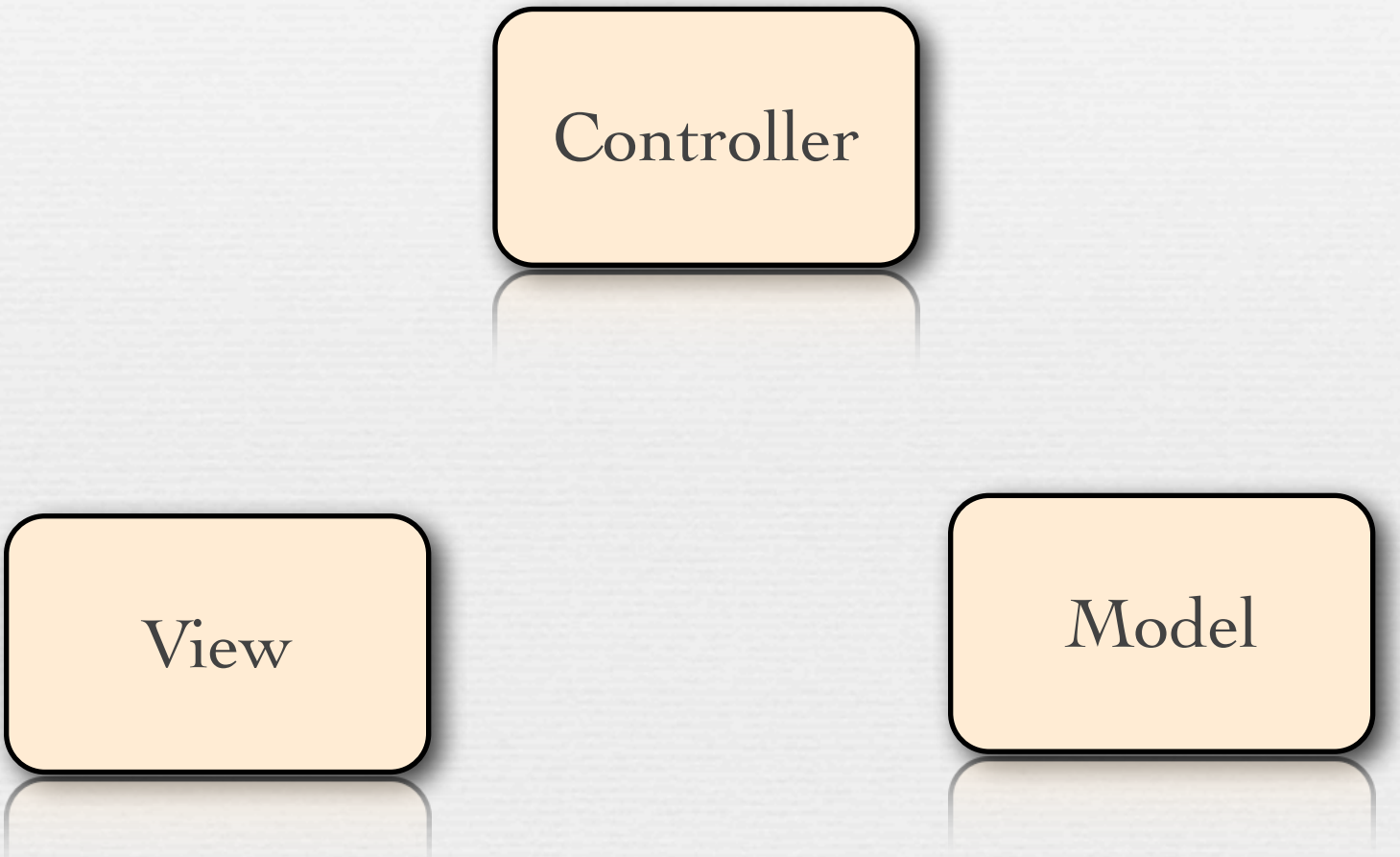
View

La vue: présentée à l'utilisateur

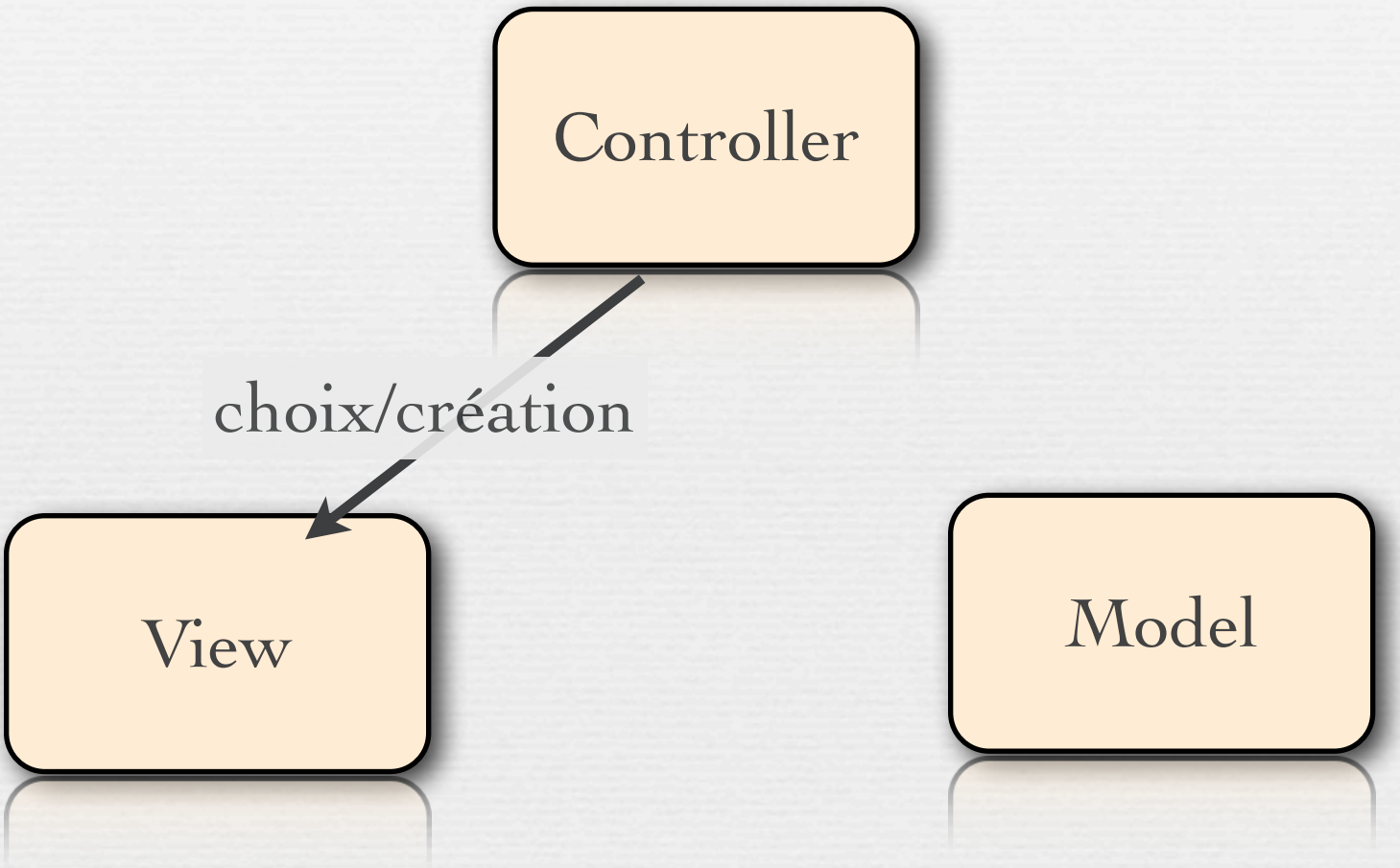
Model

Le modèle: les données indépendantes

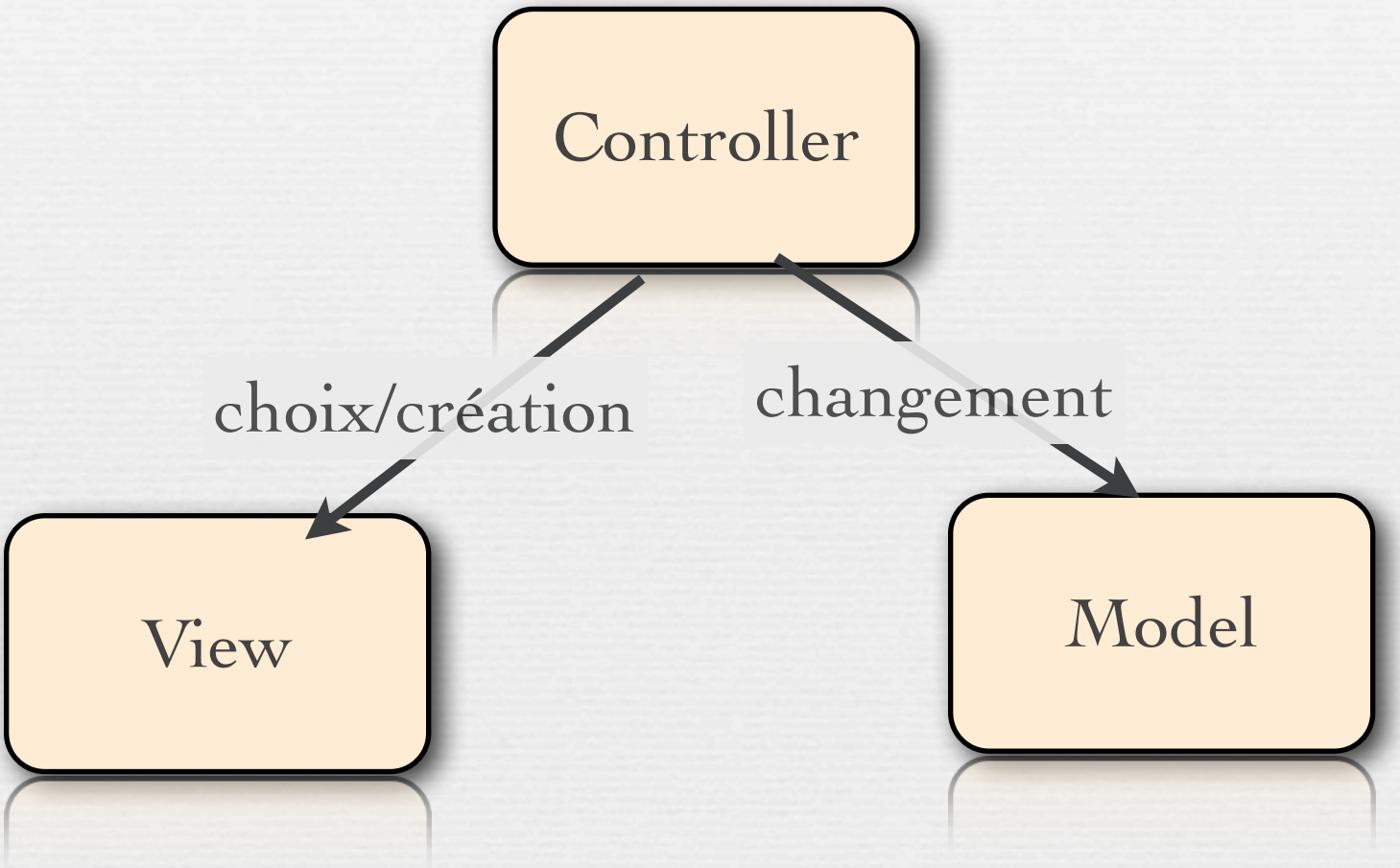
MVC



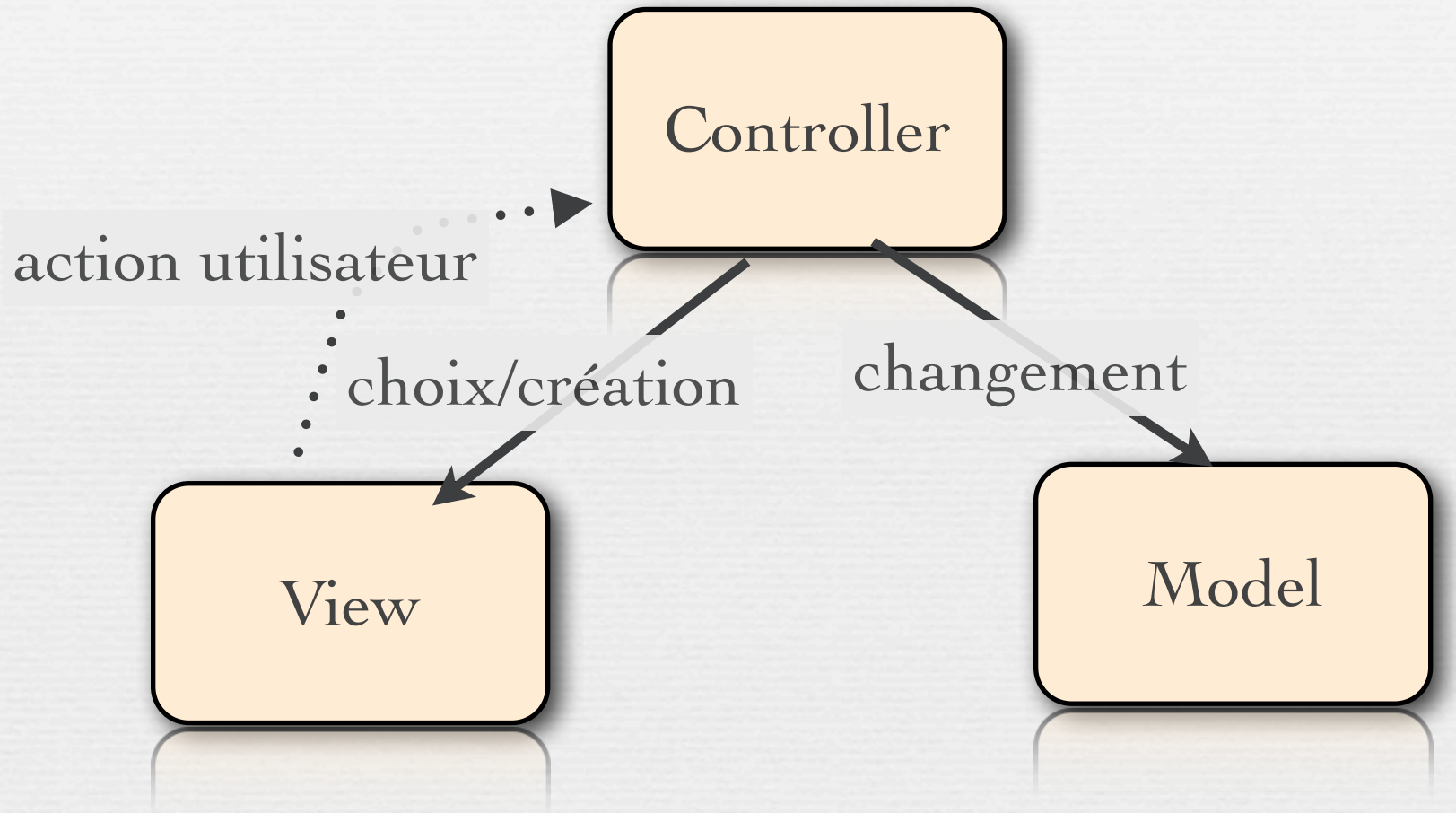
MVC



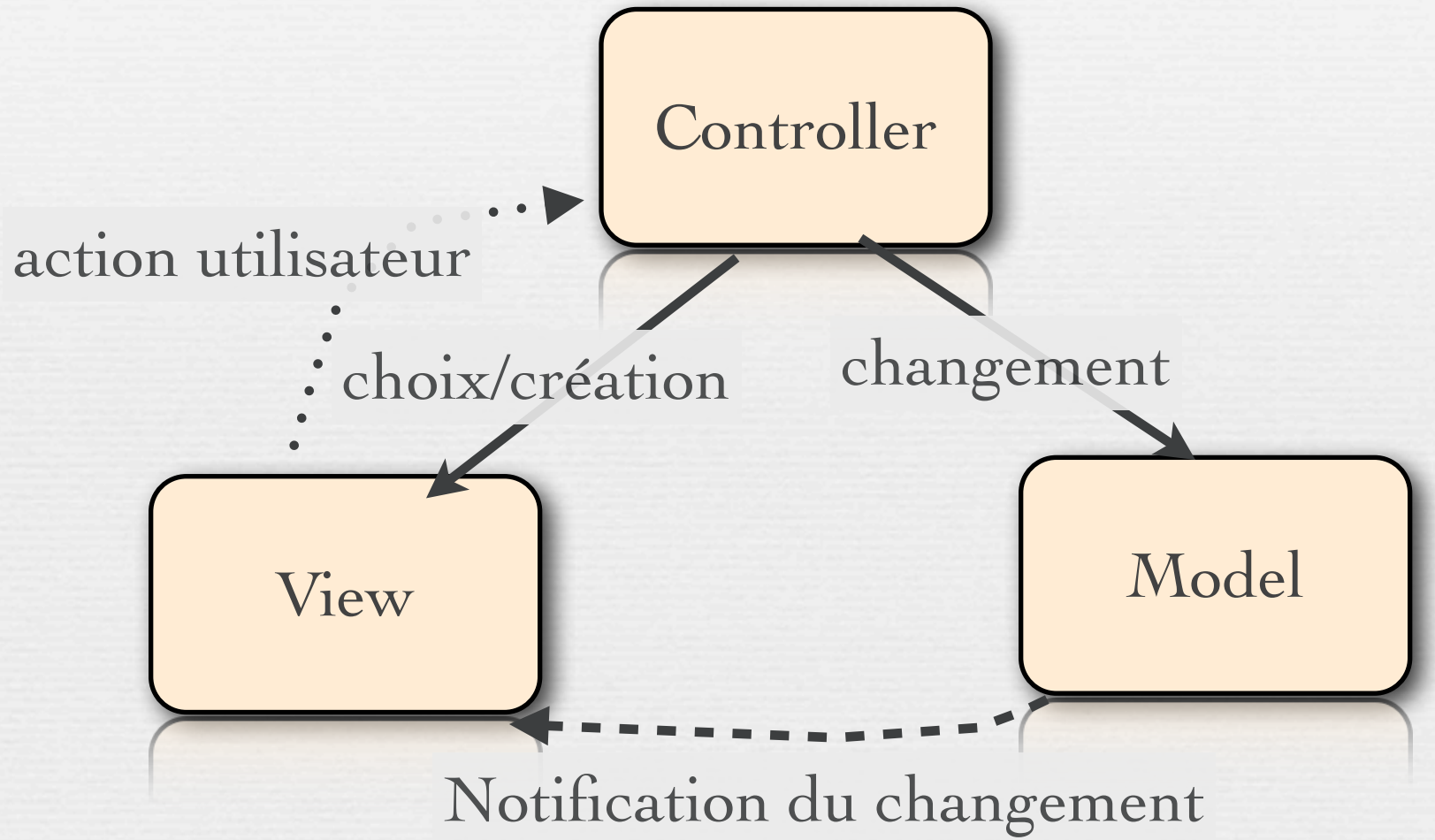
MVC



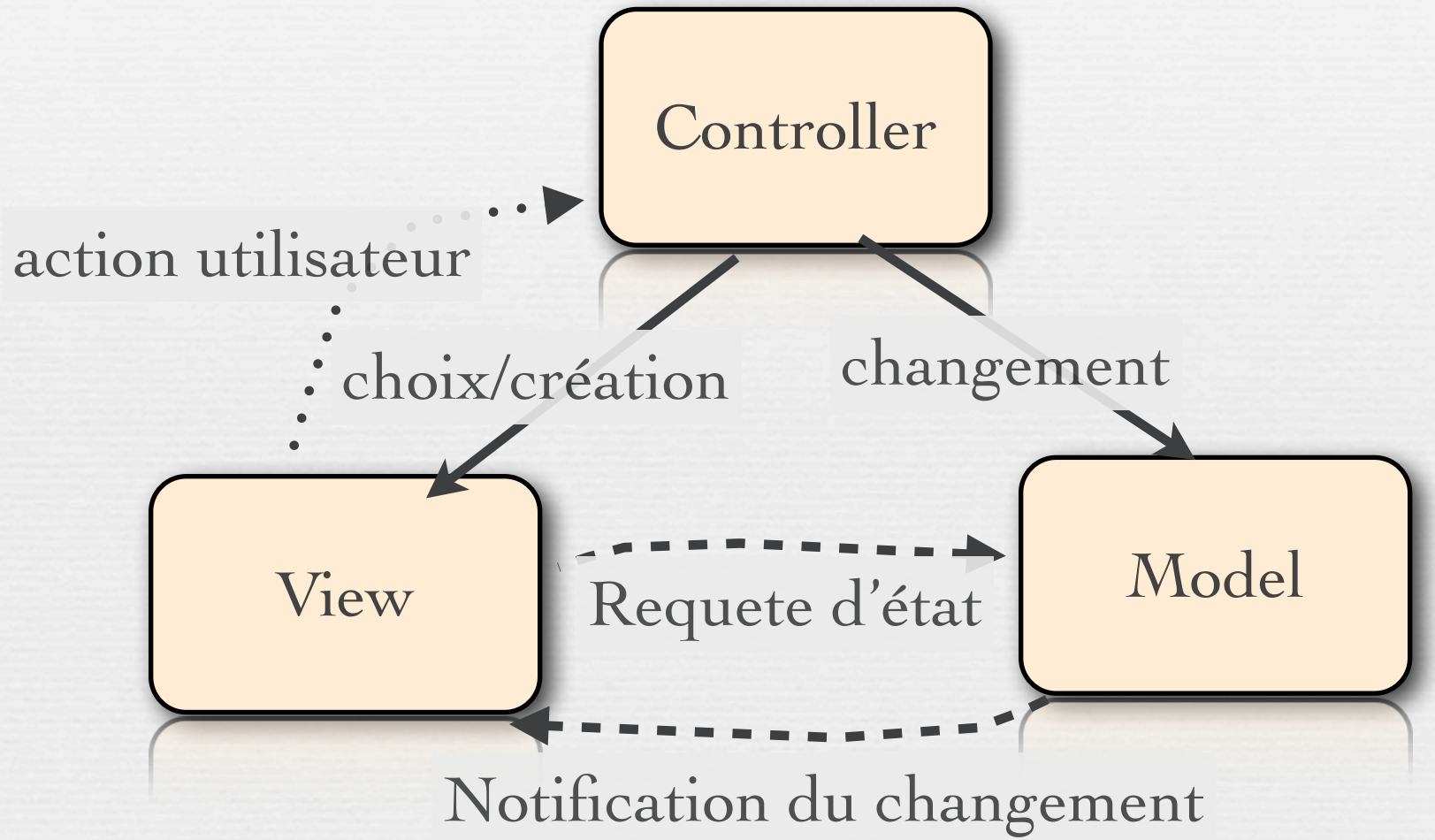
MVC

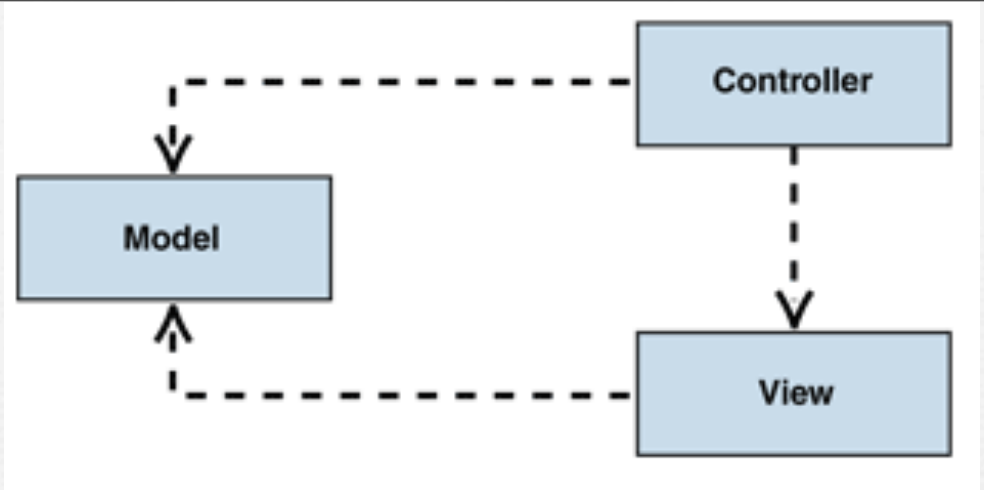

Controller
action utilisateur
choix/création
changement
View
Model

MVC

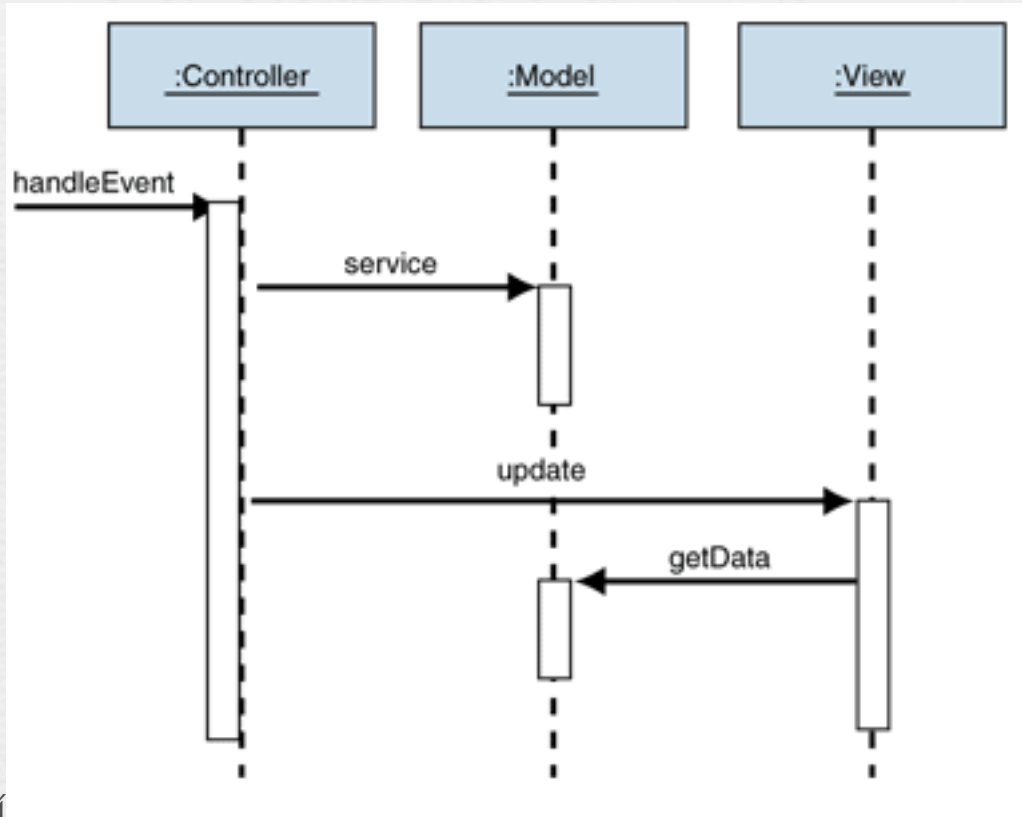

Controller
action utilisateur
choix/création
changement
View
Model
Notification du changement

MVC


Controller
action utilisateur
choix/création
changement
View
Requete d'état
Model
Notification du changement

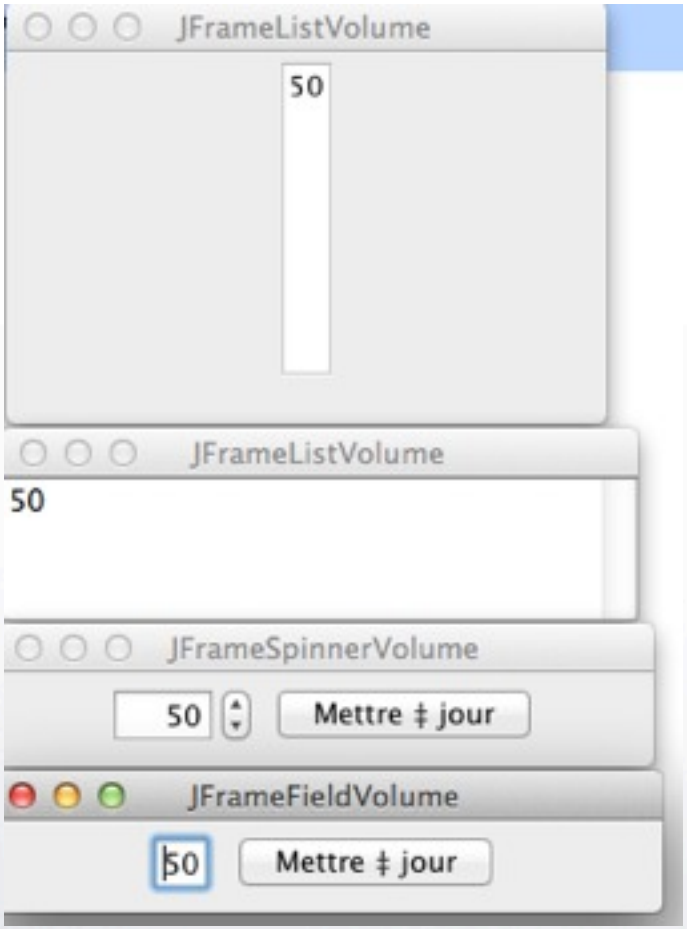


MVC : vision MS



http://msdn.microsoft.com/en-us/library/ff649643.aspx

Exemple en java



- Une vue listant les différents volumes et qui ajoutera chaque nouveau volume dans une liste déroulante : JFrameListVolume
- Une vue permettant de modifier le volume à l'aide d'un spinner avec un bouton permettant de valider le nouveau volume : JFrameSpinnerVolume
- Une vue permettant de modifier le volume avec un champ texte avec un bouton permettant de valider le nouveau volume : JFrameFieldVolume

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Modèle

```
public class VolumeModel {
    private int volume;

    public VolumeModel(){
        super();
        volume = 0;
    }

    public int getVolume() {
        return volume;
    }

    public void setVolume(int volume)
{
        this.volume = volume;
    }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Modèle «observable*»

```
import
javax.swing.event.EventListenerList;

public class VolumeModel {
  private int volume;
  private EventListenerList listeners;

  public VolumeModel(){
    this(0);
  }
  public VolumeModel(int volume){
    super();
    this.volume = volume;
    listeners = new EventListenerList();
  }

  public void setVolume(int volume) {
        this.volume = volume;
        fireVolumeChanged();
    }
```

```
public void
addVolumeListener(VolumeListener
listener){
 listeners.add(
     VolumeListener.class,listener);
}
```

```
public void fireVolumeChanged(){
  VolumeListener[] listenerList =
(VolumeListener[])listeners.getListener
s(VolumeListener.class);
  for(VolumeListener listener :
listenerList){
    listener.volumeChanged(new
VolumeChangedEvent(this, getVolume()));
        }
    }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Evénements ...

```
import java.util.EventListener;

public interface VolumeListener extends EventListener {
    public void volumeChanged(VolumeChangedEvent event);
}

import java.util.EventObject;

public class VolumeChangedEvent extends EventObject{
    private int newVolume;

    public VolumeChangedEvent(Object source, int newVolume){
        super(source);

        this.newVolume = newVolume;
    }

    public int getNewVolume(){
        return newVolume;
    }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Vue «abstraite» à l'écoute*

```
public abstract class VolumeView implements VolumeListener{
    private VolumeController controller = null;

    public VolumeView(VolumeController controller){
        super();
        this.controller = controller;
    }

    public final VolumeController getController(){
        return controller;
    }

    public abstract void display();
    public abstract void close();
}
```

Contrôleur

```
public class VolumeController {
  public VolumeView fieldView = null;
  public VolumeView spinnerView = null;
  public VolumeView listView = null;

  private VolumeModel model = null;

  public VolumeController (VolumeModel
model){
    this.model = model;

    fieldView = new
JFrameFieldVolume(this,model.getVolume());
    spinnerView = new
JFrameSpinnerVolume(this,model.getVolume()
);
    listView = new JFrameListVolume(this,
model.getVolume());

    addListenersToModel();
  }
```

```
  private void addListenersToModel() {
    model.addVolumeListener(fieldView);
    model.addVolumeListener(spinnerView);
    model.addVolumeListener(listView);
  }

  public void displayViews(){
    fieldView.display();
    spinnerView.display();
    listView.display();
  }

  public void closeViews(){
    fieldView.close();
    spinnerView.close();
    listView.close();
  }

  public void notifyVolumeChanged(int
volume){
    model.setVolume(volume);
  }
```

Vue «réactive et active»

```
public class JFrameFieldVolume extends
VolumeView impl  ements ActionListener{
  ........

  public
JFrameFieldVolume(VolumeController
controller) {
    this(controller, 0);
  }

  public
JFrameFieldVolume(VolumeController
controller, int volume){
    super(controller);
    buildFrame(volume);
  }

  private void buildFrame(int volume) {
    frame = new JFrame();
    .....
    field.setValue(volume);
    .....
  }
```

```
  @Override
  public void close() {
    frame.dispose();
  }

  @Override
  public void display() {
    frame.setVisible(true);
  }

    public void
  volumeChanged(VolumeChangedEvent event)
  {
  field.setValue(event.getNewVolume());
    }

    public void
  actionPerformed(ActionEvent arg0) {
     getController().
       notifyVolumeChanged(
  Integer.parseInt(field.getValue().toStr
  ing()));
    }
```

Vue (réactive)

```
public class JFrameListVolume
extends VolumeView {
     ....
  public
JFrameListVolume(VolumeController
controller) {
    this(controller, 0);
  }

  public
JFrameListVolume(VolumeController
controller, int volume){
    super(controller);
    buildFrame(volume);
  }

  private void buildFrame(int
volume) {
    ...
    jListModel.addElement(volume);
        list....
  }
```

```
@Override
public void close() {
  frame.dispose();
}

@Override
public void display() {
  frame.setVisible(true);
}

  public void
volumeChanged(VolumeChangedEvent
event) {
jListModel.addElement(event.getNewVo
lume());
  }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Lanceur

```
package volume;

public class JVolume {
  public static void main(String[] args) {
    VolumeModel model = new VolumeModel(50);
    VolumeController controller = new VolumeController(model);
    controller.displayViews();
  }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Vue «réactive et active»

```
public class JFrameFieldVolume extends
VolumeView impl  ements ActionListener{
  ........

  public
JFrameFieldVolume(VolumeController
controller) {
    this(controller, 0);
  }

  public
JFrameFieldVolume(VolumeController
controller, int volume){
    super(controller);
    buildFrame(volume);
  }

  private void buildFrame(int volume) {
    frame = new JFrame();
    .....
    field.setValue(volume);
    .....
  }
```

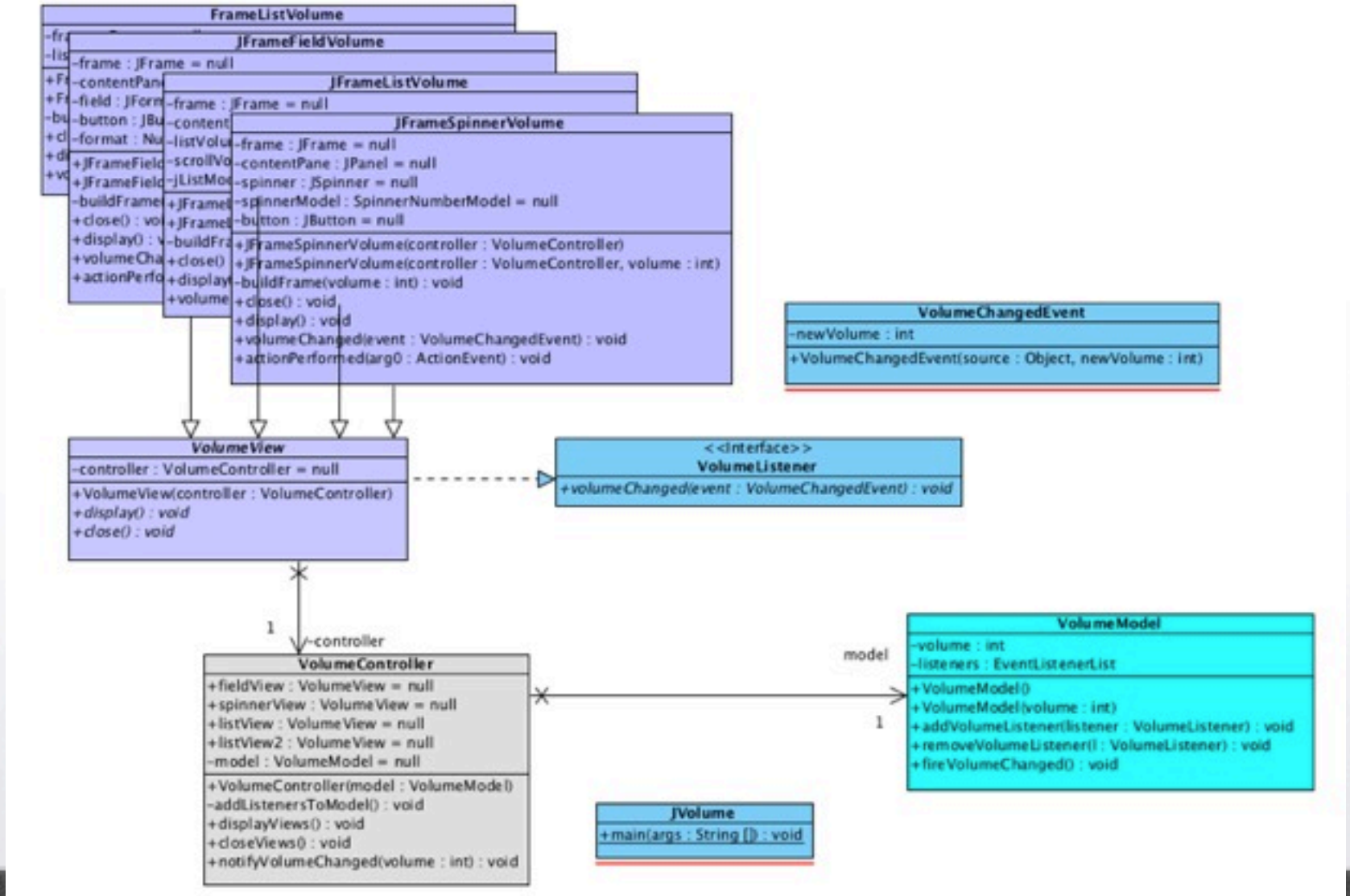
```
  @Override
  public void close() {
    frame.dispose();
  }

  @Override
  public void display() {
    frame.setVisible(true);
  }

  public void
volumeChanged(VolumeChangedEvent event)
{
field.setValue(event.getNewVolume());
  }

  public void
actionPerformed(ActionEvent arg0) {
   getController().
     notifyVolumeChanged(
Integer.parseInt(field.getValue().toStr
ing()));
  }
```

Contrôleur

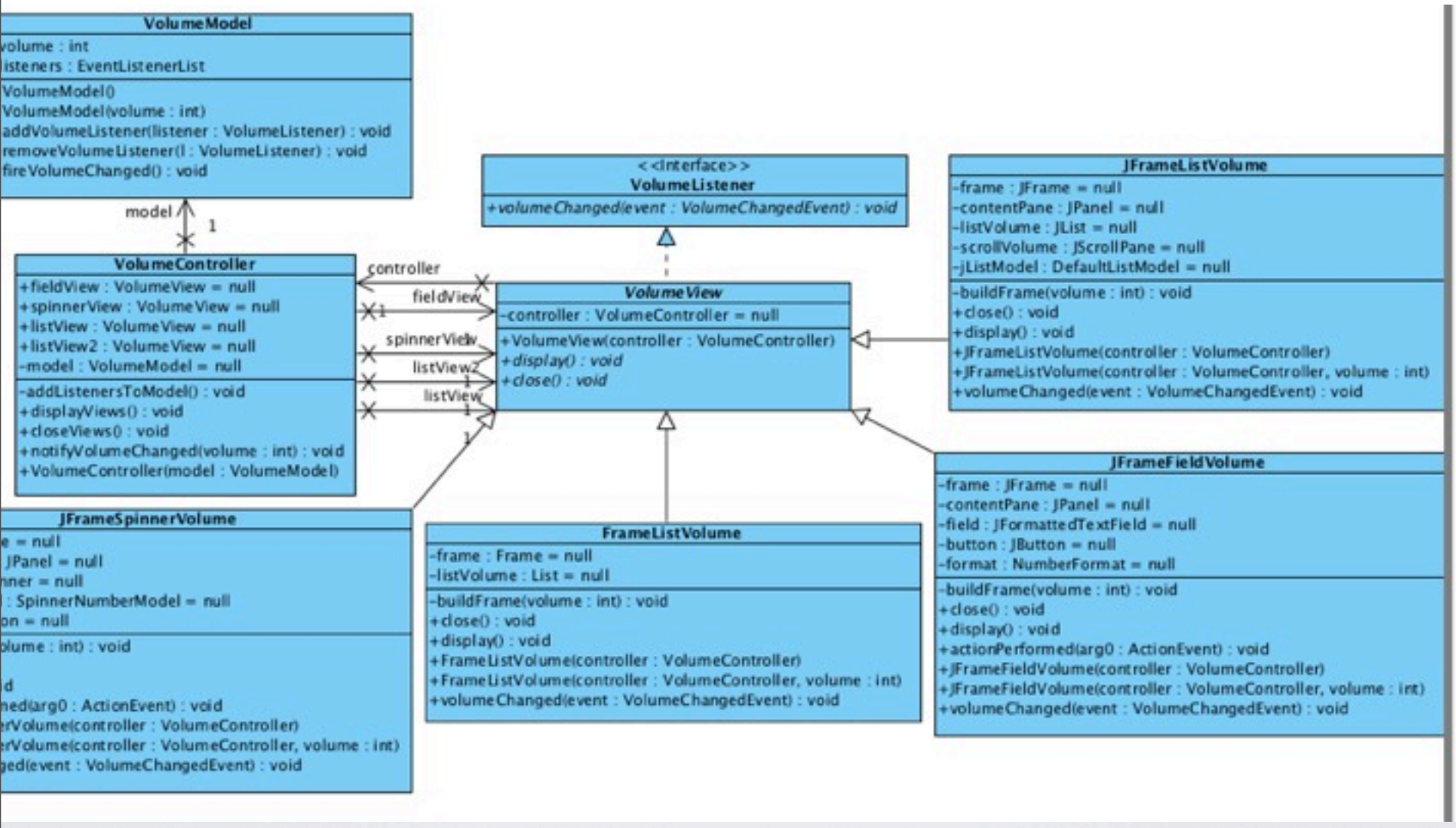
```
public class VolumeController {
  public VolumeView fieldView = null;
  public VolumeView spinnerView = null;
  public VolumeView listView = null;

  private VolumeModel model = null;

  public VolumeController (VolumeModel
model){
    this.model = model;

    fieldView = new
JFrameFieldVolume(this,model.getVolume());
    spinnerView = new
JFrameSpinnerVolume(this,model.getVolume()
);
    listView = new JFrameListVolume(this,
model.getVolume());

    addListenersToModel();
  }
```

```
  private void addListenersToModel() {
    model.addVolumeListener(fieldView);
    model.addVolumeListener(spinnerView);
    model.addVolumeListener(listView);
  }

  public void displayViews(){
    fieldView.display();
    spinnerView.display();
    listView.display();
  }

  public void closeViews(){
    fieldView.close();
    spinnerView.close();
    listView.close();
  }

  public void notifyVolumeChanged(int
volume){
    model.setVolume(volume);
  }
```

Vue UML des classes



Reverse-Engineering



http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

SEPARATIONS : Persistance et accès aux données : DAO (Data Access Object Pattern)

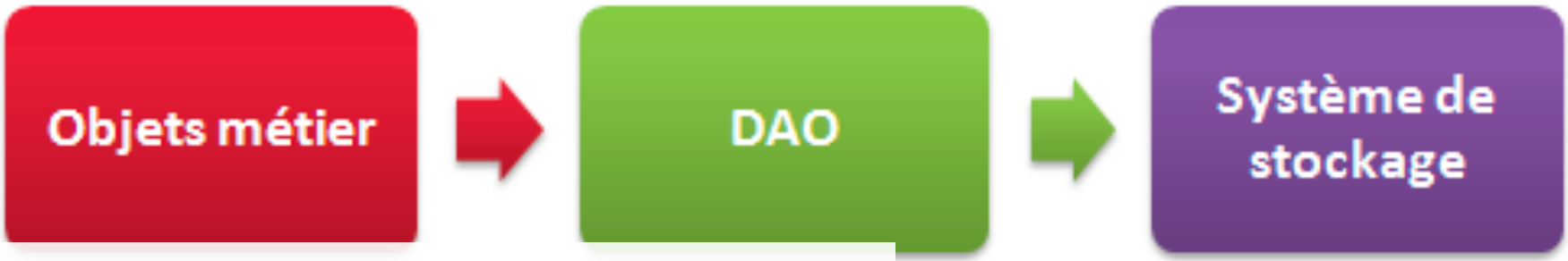
http://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm

Séparer objets métier et stockage des données : Data Access Object

Séparer les données et la manière dont elles sont stockées

http://openclassrooms.com/courses/creez-votre-application-web-avec-java-ee/le-modele-dao

Séparer objets métier et stockage des données : DAO



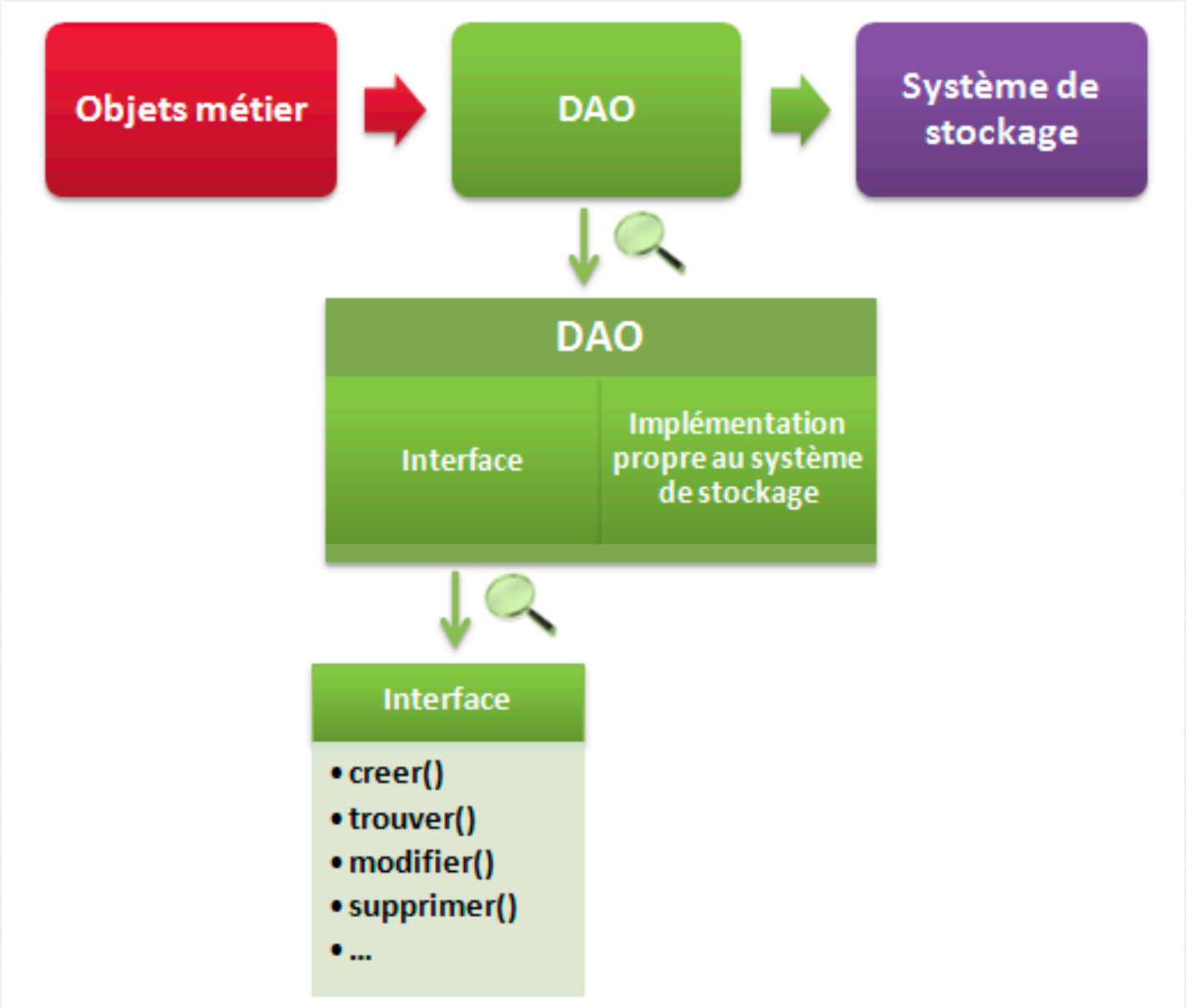
Gérer le stockage :

- Create : créer la donnée,
- Read : lire les données,
- Update : mettre à jour les données,
- Delete : effacer les données

Séparer objets métier et stockage des données : DAO

Indépendance de la mise en oeuvre du stockage ?

DAO


Objets métier
DAO
Système de stockage
DAO
Interface
Implémentation propre au système de stockage
Interface
• creer()
• trouver()
• modifier()
• supprimer()
• ...

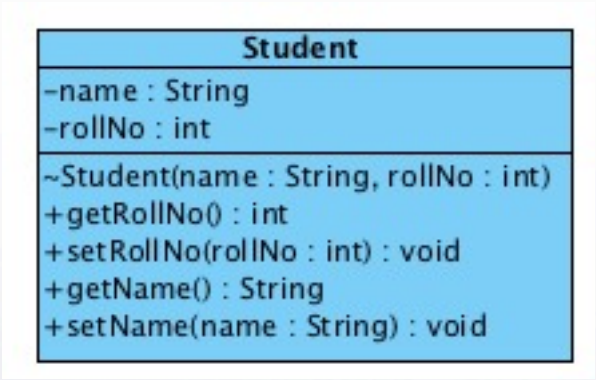
DAO

- ✓ Soit une classe Utilisateur : un nom et un email
- ✓ On veut pouvoir les sauvegarder dans un fichier.
- ✓ On veut pouvoir les sauvegarder dans une base de données

Voir codes au tableau -- les slides suivants présentent un exemple identique mais avec un étudiant qui a un nom et un numéro.

Séparer la logique métier de l'accès aux données

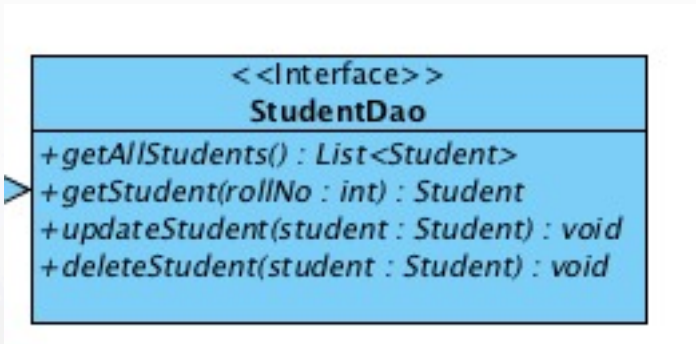
Un étudiant




```
public class Student {
   private String name;
   private int rollNo;

   Student(String name, int rollNo){
      this.name = name;
      this.rollNo = rollNo;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getRollNo() {
      return rollNo;
   }

   public void setRollNo(int rollNo) {
      this.rollNo = rollNo;
   }
}
```

Séparer la logique métier de l'accès aux données

Des étudiants




```java
import java.util.List;

public interface StudentDao {
   public List<Student> getAllStudents();
   public Student getStudent(int rollNo);
   public void updateStudent(Student student);
   public void deleteStudent(Student student);
}
```

Séparer la logique métier de l'accès aux données

USAGE

```
StudentDao studentDao = ....

//print all students
for (Student student : studentDao.getAllStudents()) {
   System.out.println("Student: [RollNo : "
      +student.getRollNo()+", Name : "+student.getName()+" ]");
}

//update student
Student student =studentDao.getAllStudents().get(0);
student.setName("Michael");
studentDao.updateStudent(student);

//get the student
studentDao.getStudent(0);
System.out.println("Student: [RollNo : "
   +student.getRollNo()+", Name : "+student.getName()+" ]");
   }
}
```

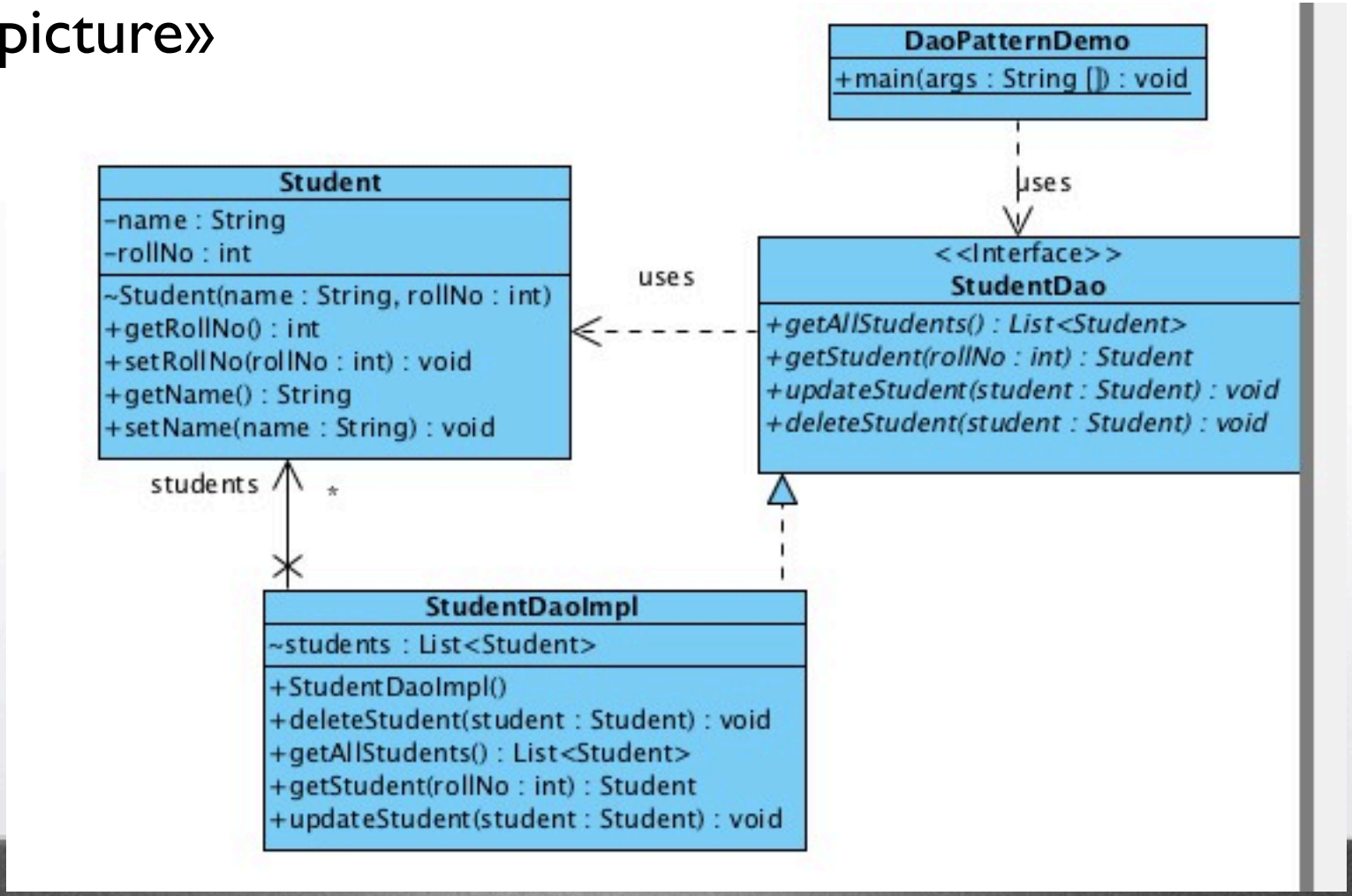
Séparer la logique métier de l'accès aux données

Connexion à la BD

```
import java.util.ArrayList;
import java.util.List;

public class StudentDaoImpl implements
StudentDao {

   //list is working as a database
   List<Student> students;

   public StudentDaoImpl(){
      students = new ArrayList<Student>();
      Student student1 = new
Student("Robert",0);
      Student student2 = new Student("John",
1);
      students.add(student1);
      students.add(student2);
   }
   @Override
   public void deleteStudent(Student
student) {
      students.remove(student.getRollNo());
      System.out.println("Student: Roll No "
+ student.getRollNo()
         +", deleted from database");
   }

   //retrive list of students from the
database
   @Override
   public List<Student> getAllStudents() {
      return students;
   }

   @Override
   public Student getStudent(int rollNo) {
      return students.get(rollNo);
   }

   @Override
   public void updateStudent(Student
student) {
```

Séparer la logique métier de l'accès aux données

«Big picture»



SEPARATIONS : Indépendantes du langage **un exemple en php**

Exemple simple en php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//FR"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en_US" xml:lang=
"en_US">
  <head>
    <title>
      Exemple
    </title>
  </head>
  <body>
  <?php
    $db=mysql_connect($dbhost,$dbuser,$dbpass);
    mysql_select_db($dbname,$db);
    $query="SELECT product.name, product.price FROM product where product.type=42";
   $req=mysql_query($query)or die('Erreur SQL!<br>'.$sql.'<br>'.mysql_error());
   while ($row = mysql_fetch_array($req)) {
   echo "Nom :".$row[0]." Prix :".$row[1];
}
    mysql_close();
  ?>
  </body>
</html>
```

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Exemple simple en php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//FR"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en_US" xml:lang=
"en_US">
  <head>
    <title>
      Exemple
    </title>
  </head>
  <body>
  <?php
    $db=mysql_connect($dbhost,$dbuser,$dbpass);
    mysql_select_db($dbname,$db);
    $query="SELECT product.name, product.price FROM product where product.type=42";
   $req=mysql_query($query)or die('Erreur SQL!<br>'.$sql.'<br>'.mysql_error());
   while ($row = mysql_fetch_array($req)) {
   echo "Nom :".$row[0]." Prix :".$row[1];
}
    mysql_close();
  ?>
  </body>
</html>
```

Mise en page

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Exemple simple en php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//FR"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en_US" xml:lang=
"en_US">
  <head>
    <title>
      Exemple
    </title>
  </head>
  <body>
  <?php
    $db=mysql_connect($dbhost,$dbuser,$dbpass);
    mysql_select_db($dbname,$db);
    $query="SELECT product.name, product.price FROM product where product.type=42";
   $req=mysql_query($query)or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());
   while ($row = mysql_fetch_array($req)) {
   echo "Nom :".$row[0]." Prix :".$row[1];
}
    mysql_close();
  ?>
  </body>
</html>
```

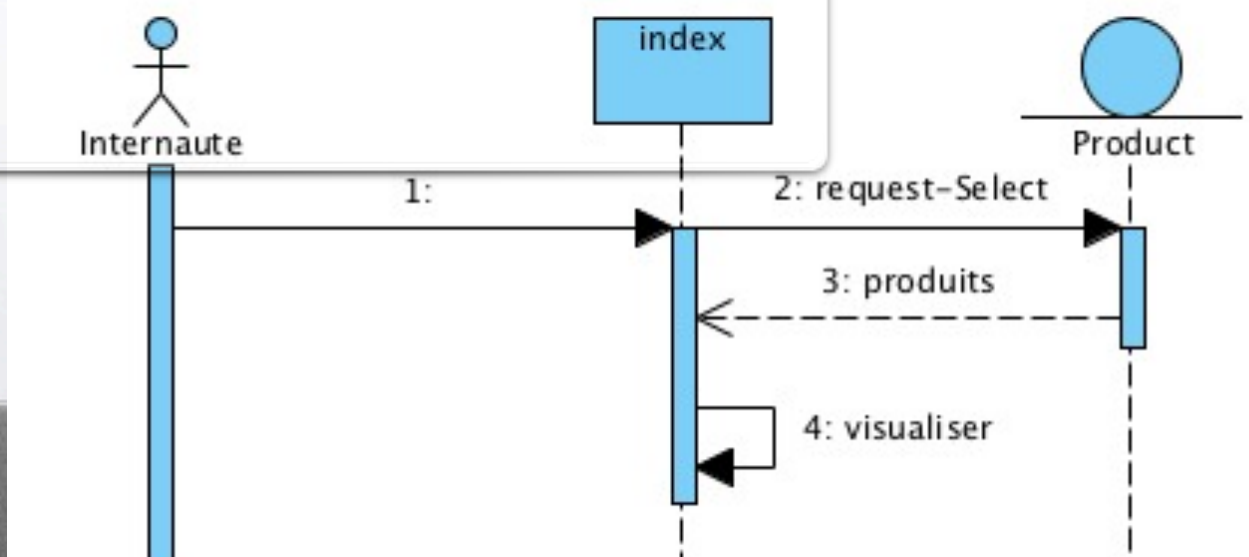
Mise en page

Accès aux données

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

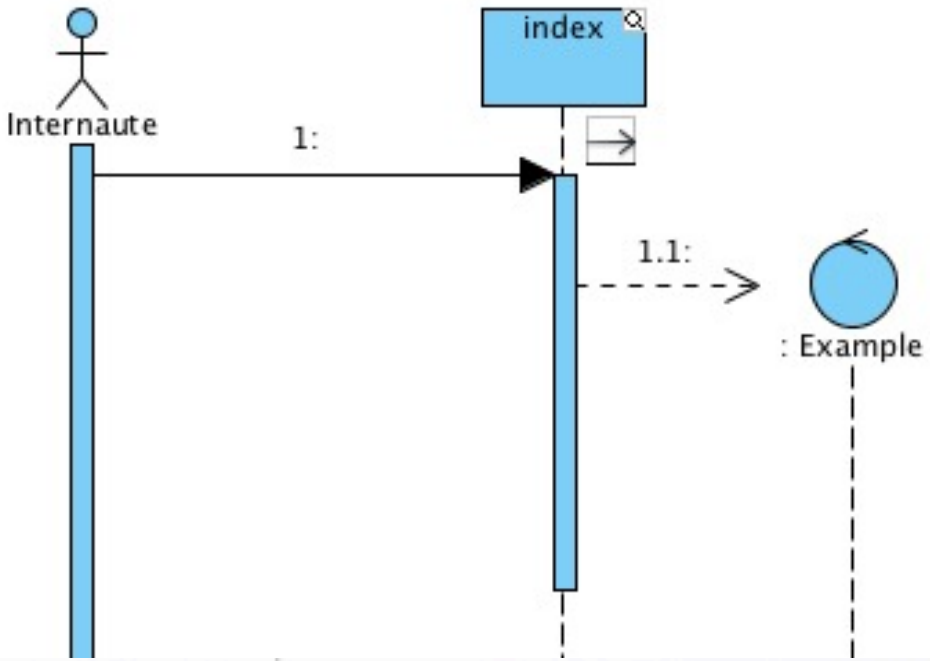
Exemple simple en php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//FR"
...
  <head>
...
  </head>
  <body>
  <?php
    $db=mysql_connect($dbhost,$dbuser,$dbpass);
    mysql_select_db($dbname,$db);
    $query="SELECT product.name, product.price FROM product where
product.type=42";
   $req=mysql_query($query)or die('Erreur SQL!<br>'.
$sql.'<br>'.mysql_error());
   while ($row = mysql_fetch_array($req)) {
   echo "Nom :".$row[0]." Prix :".$row[1];
}
    mysql_close();
  ?>
  </body>
</html>
```



Version MVC : entrée

index.php:



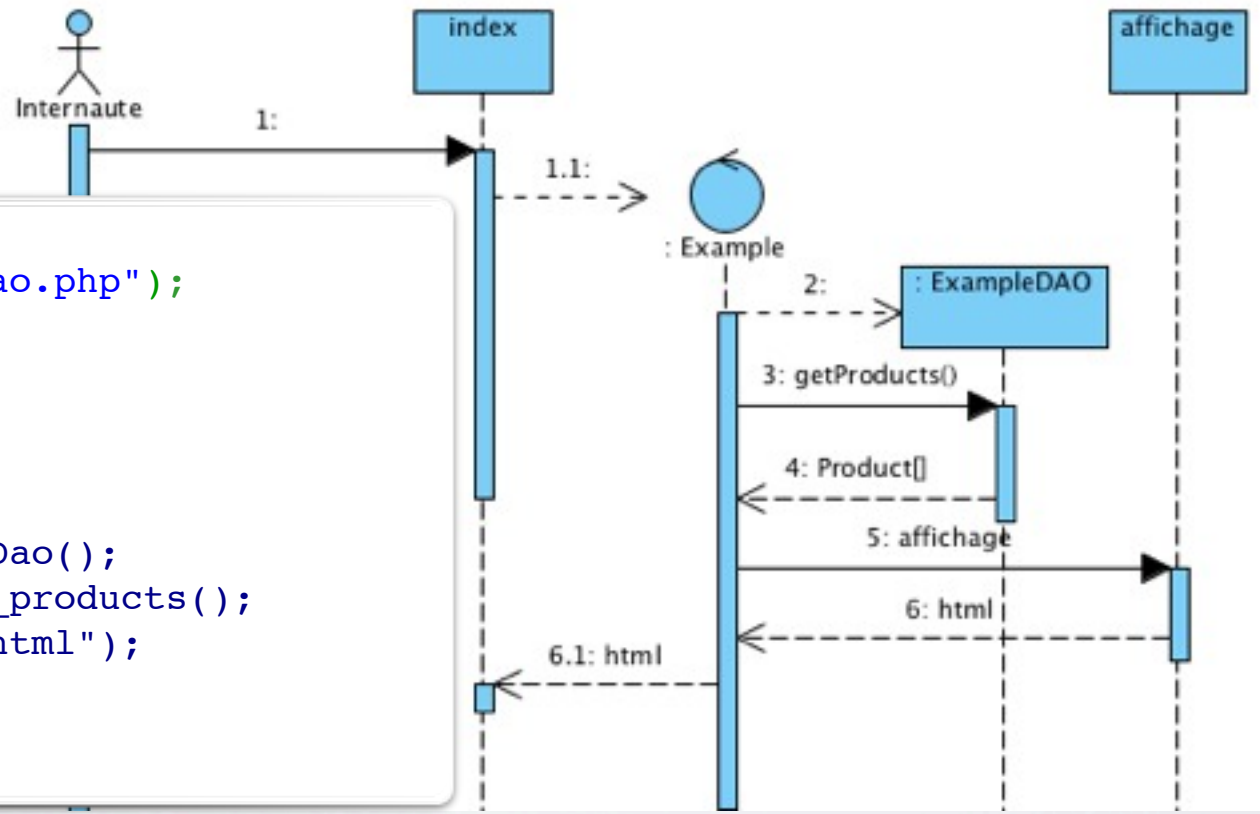

```php
<?php
if  ($_GET['do']=="")
{
  require ("default.html");
}
else
{
  if($_GET['do']=="affichage")
  {
    require("action/class_example.php");
    new Example();
  }
}
?>
```

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Contrôleur

class_example.php:

```php
<?php
 require ("dao/dao_example_dao.php");

  class Example
 {
  public function example()
  {
    $exampleDao= new ExampleDao();
    $data = $exampleDao->get_products();
    require("web/affichage.phtml");
  }
}
?>
```



Il ne fait pas grand chose...

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Visualisation

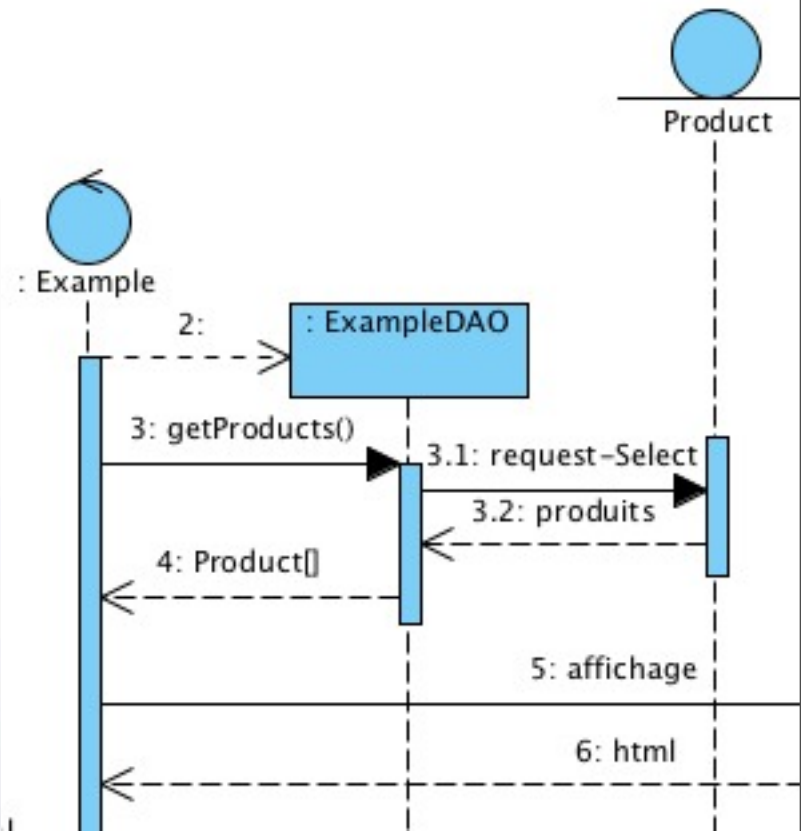
affichage.phtml:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en_US" xml:lang="en_US">
  <head>
    <title>
      Exemple
    </title>
  </head>
  <body>
    <?
foreach ($data AS $row )
{
?>
   Nom :
<? echo $row["name"]; ?>
 Prix :
<? echo $row["price"]; ?> <br/>
<?
}
?>
  </body>
</html>
```

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Accès aux données : DAO (Data Access Object)

```php
<?php
 require ("class_connect_bd.php");
 class ExampleDao
 {
   public  function get_products()
   {
    $sql="SELECT product.name,
                  product.price
          FROM product where product.type=42";
    $query = mysql_query($sql) or
          die('Erreur SQL !<br>'.mysql_error());
    $all =  mysql_fetch_all($query);
    return $all;
   }
 }
?>
```

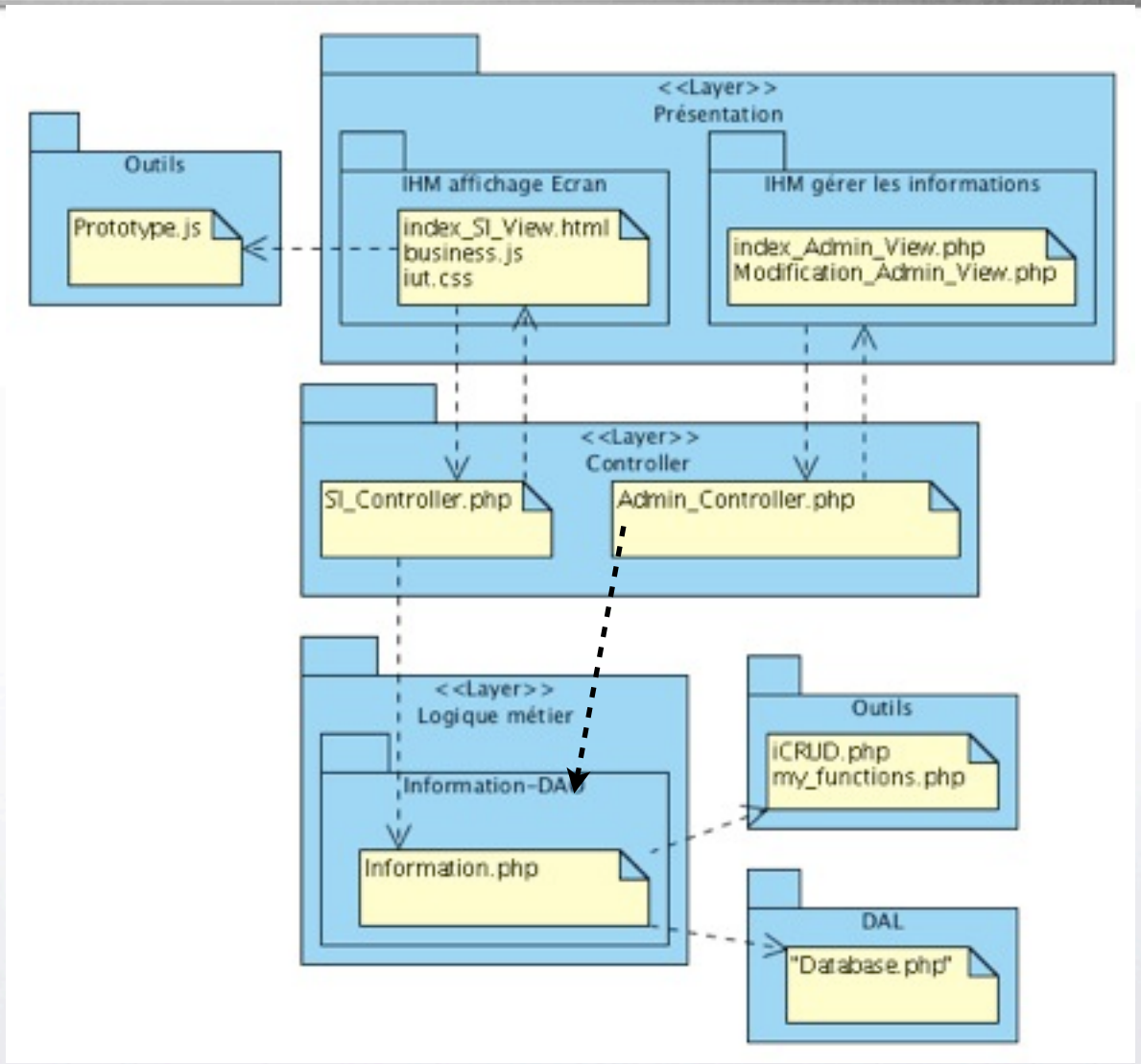


dao_example_dao.php

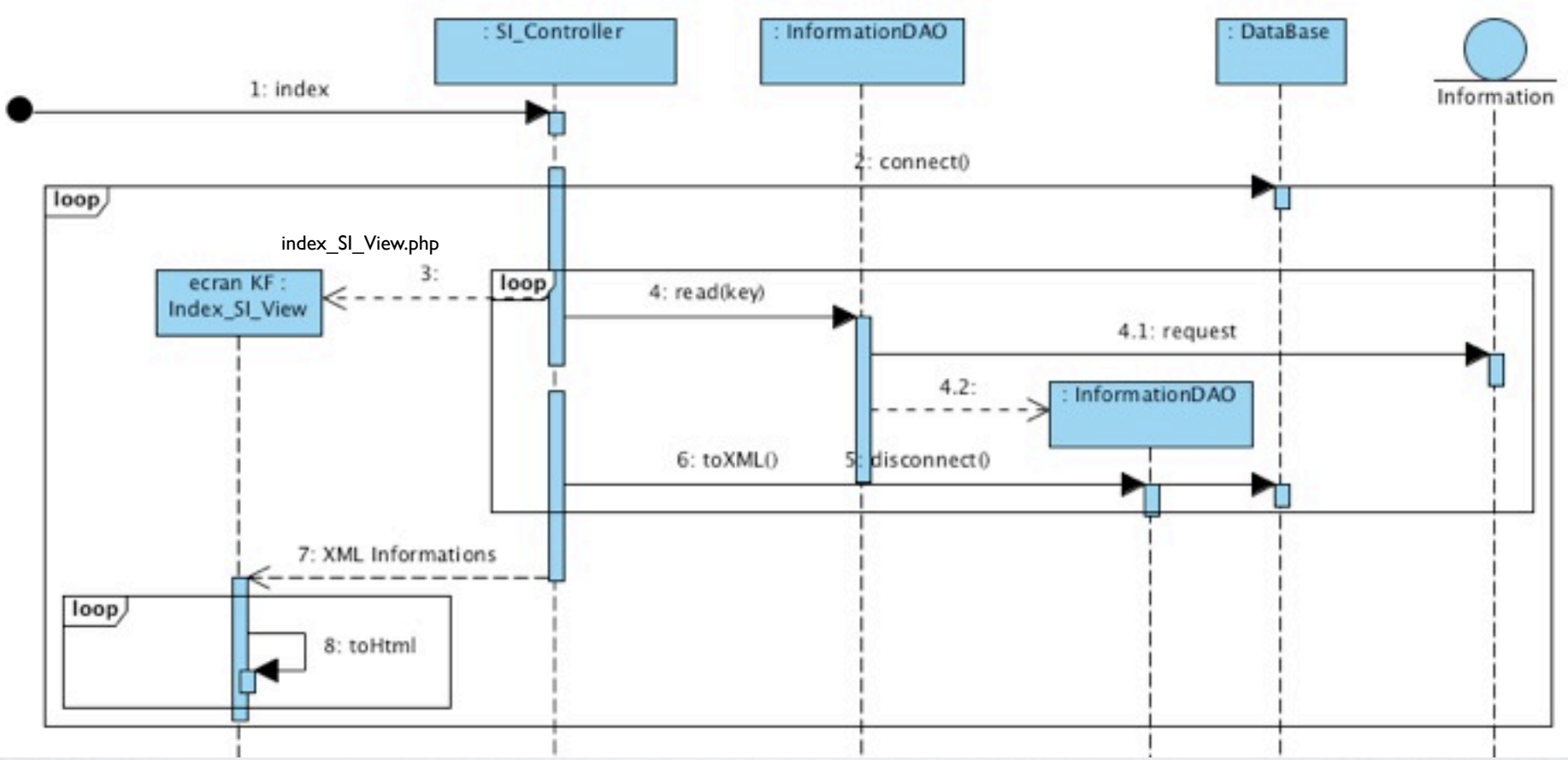
C'est tellement simple qu'il est assimilé au modèle dans cet exemple.

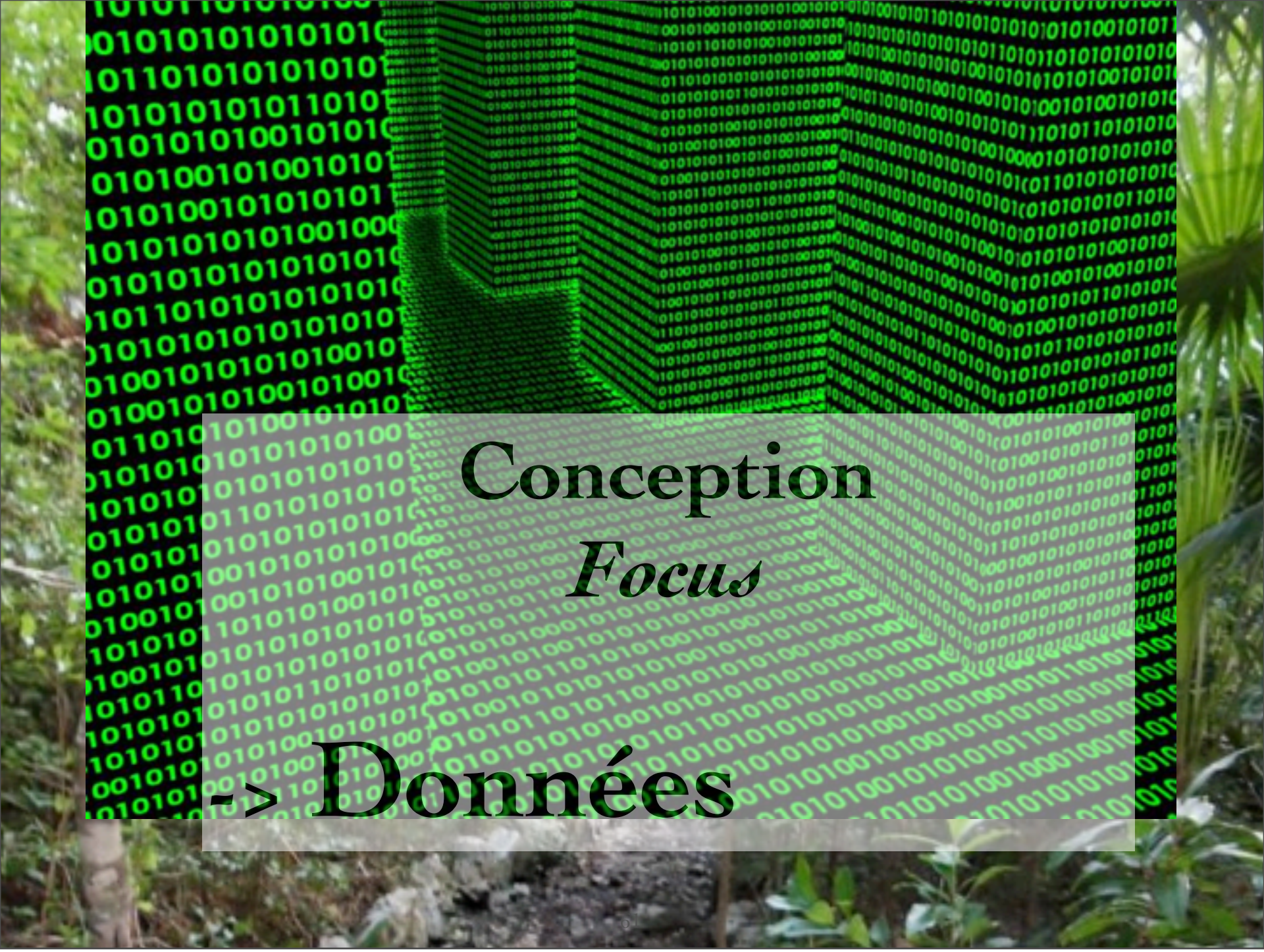
http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Architecture en couches & Fichiers

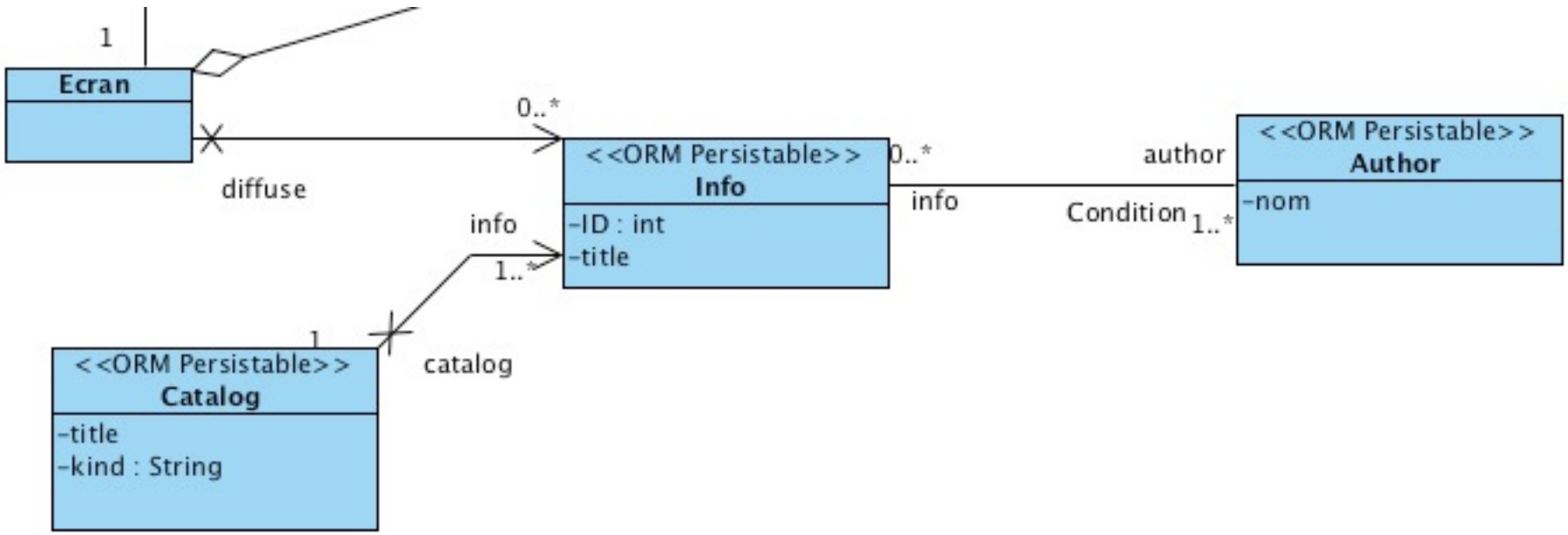


Afficher Informations : niveau Conception

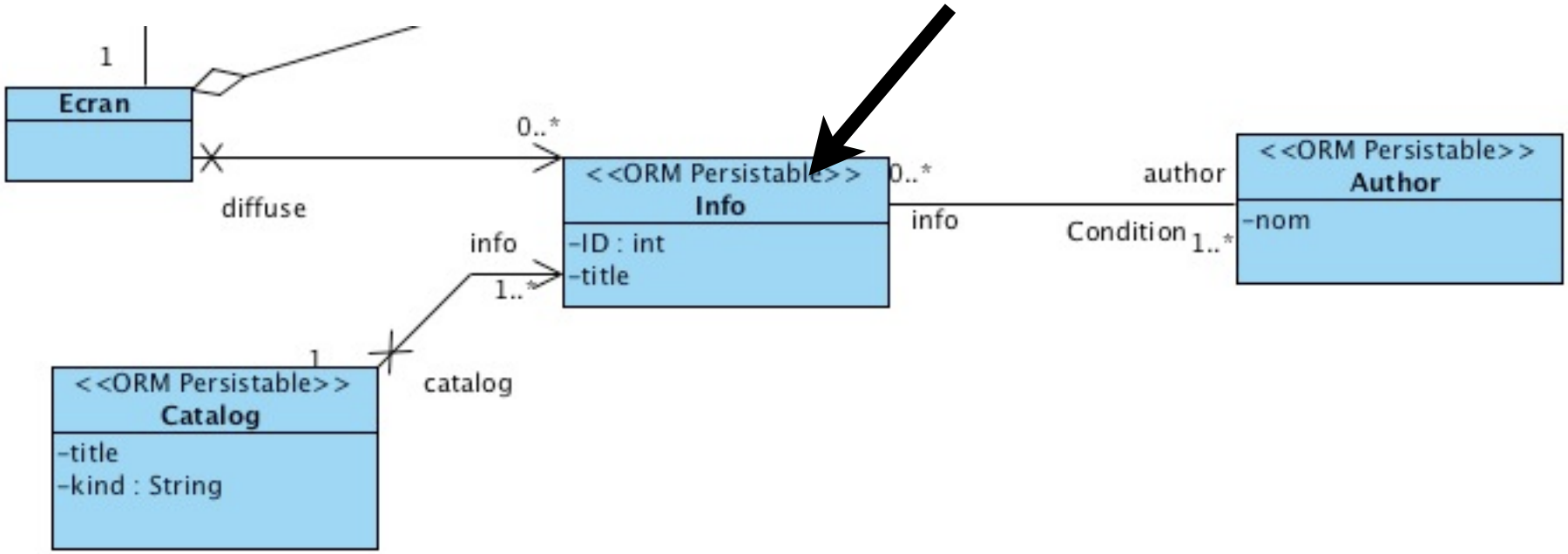



Conception
Focus
-> Données

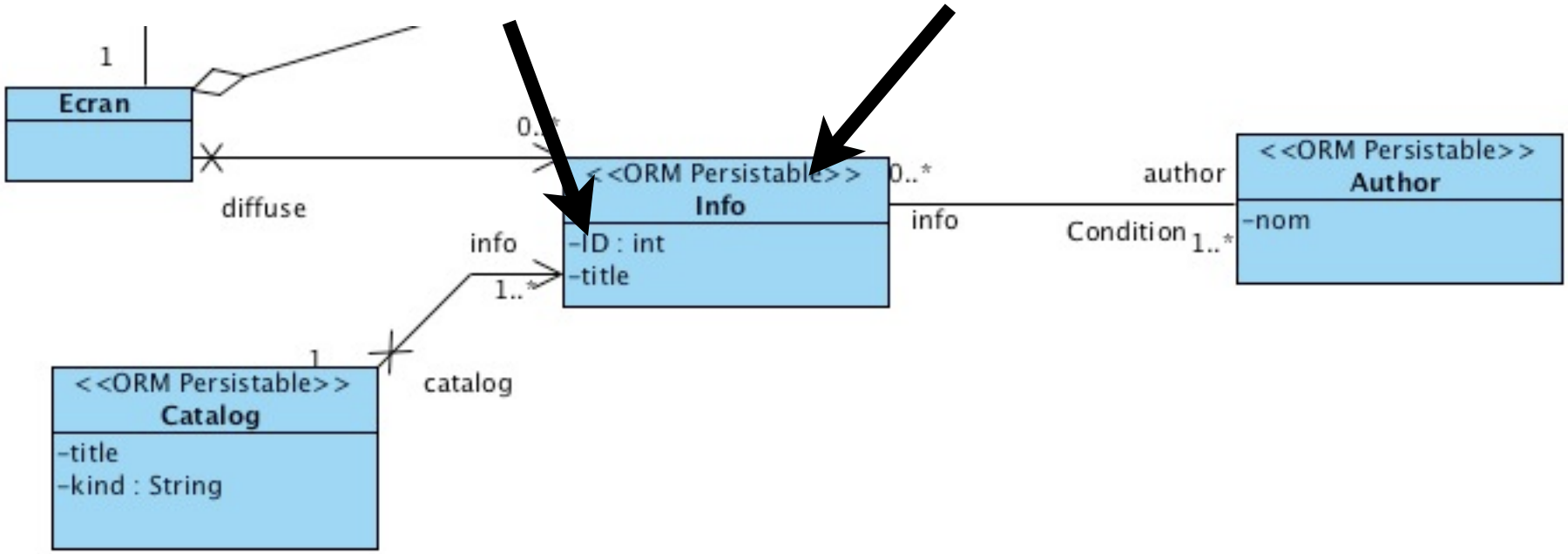
Des classes aux données



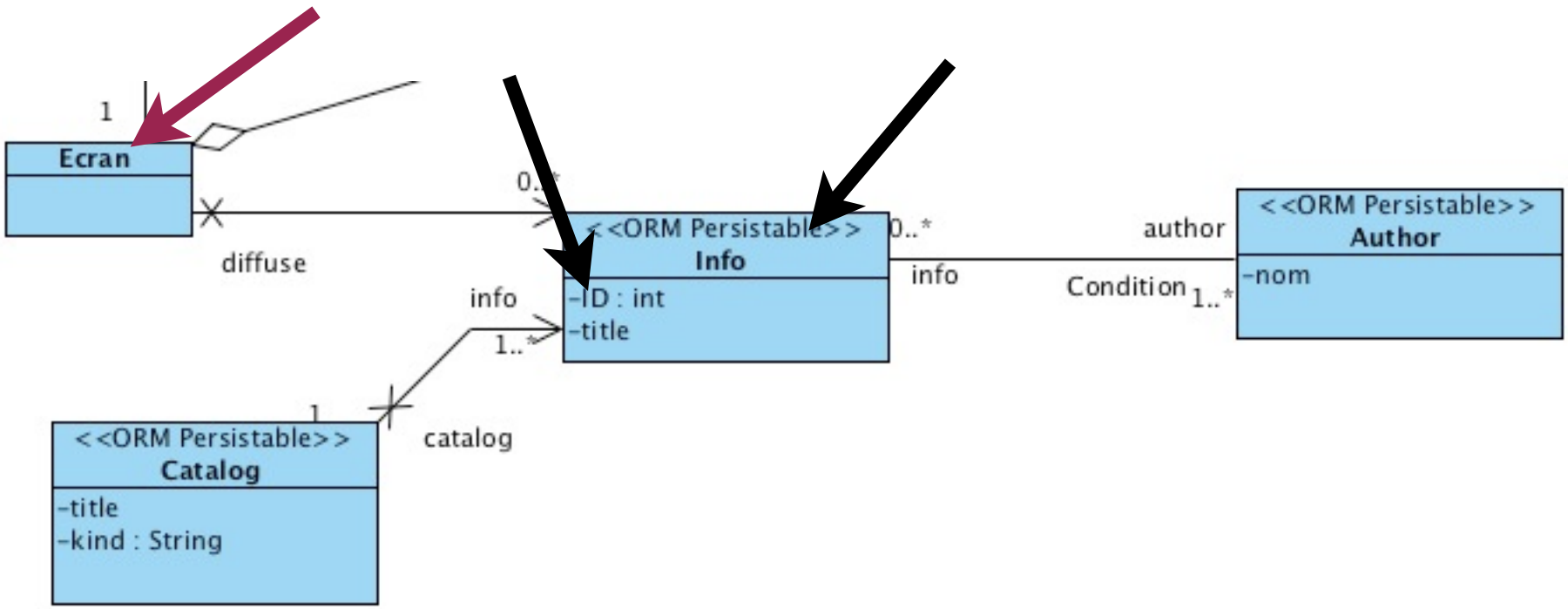
Des classes aux données



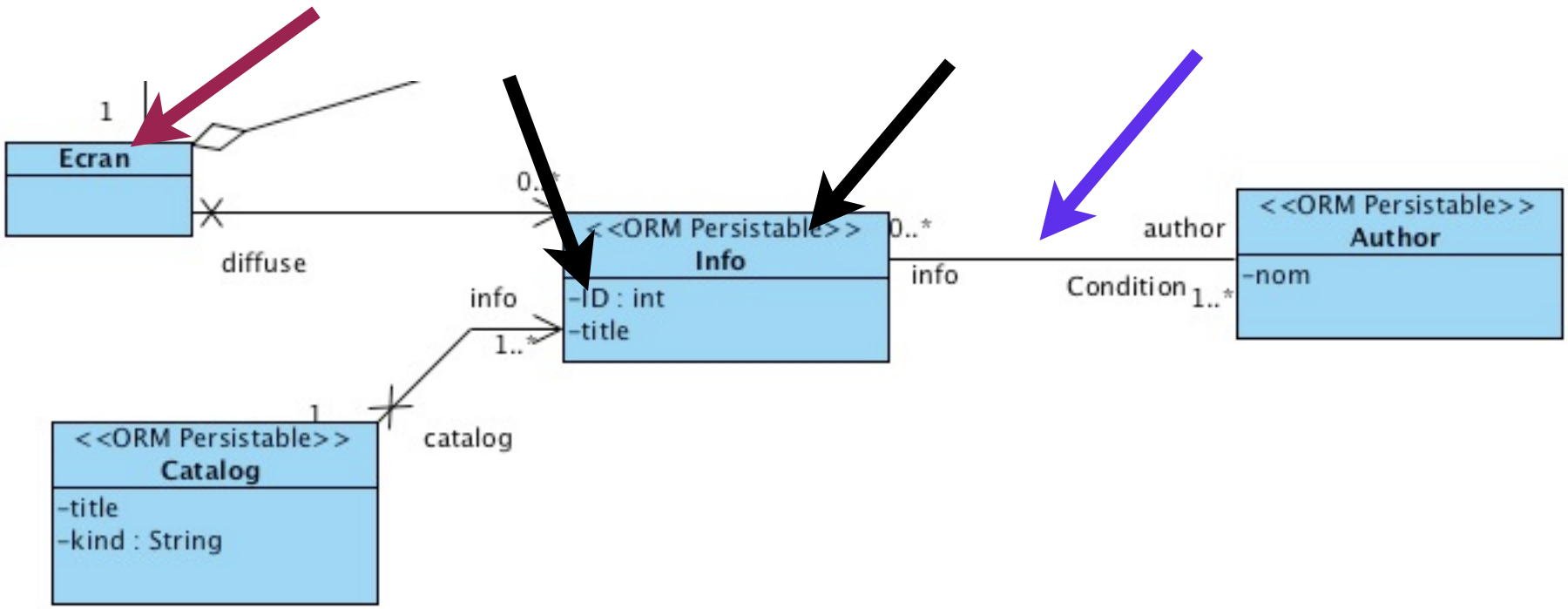
Des classes aux données



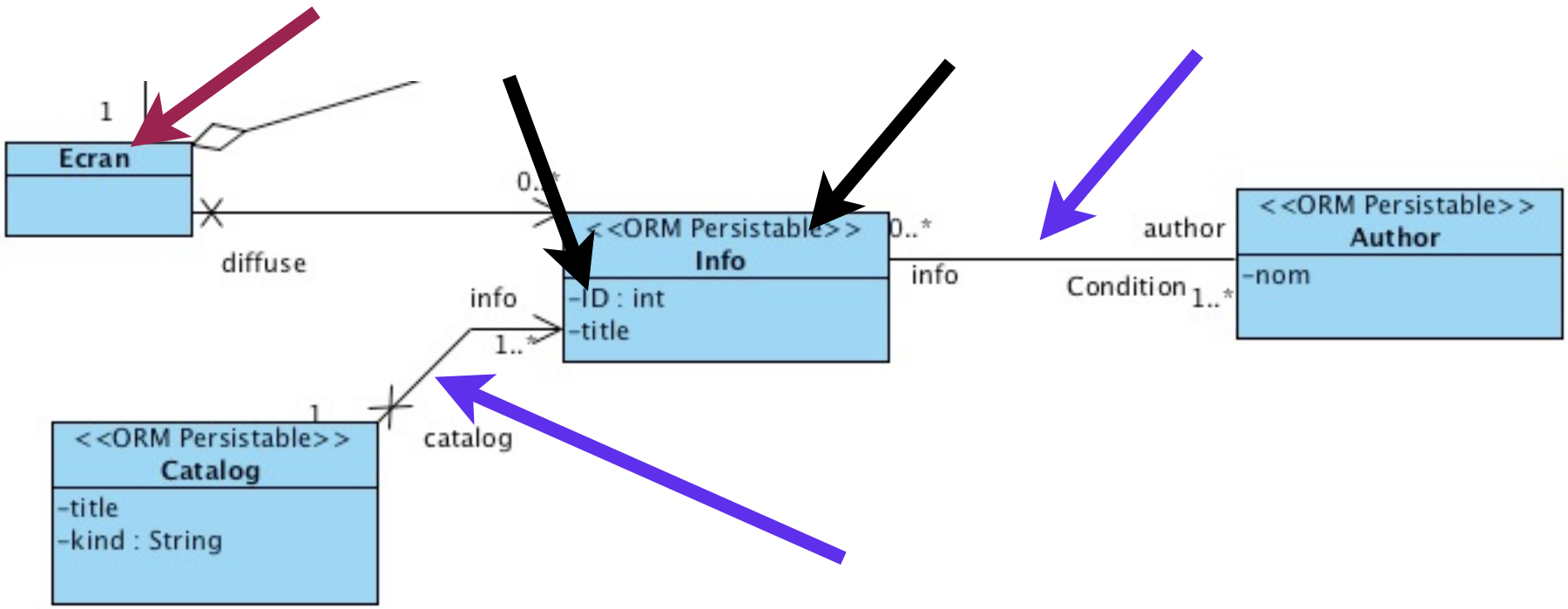
Des classes aux données



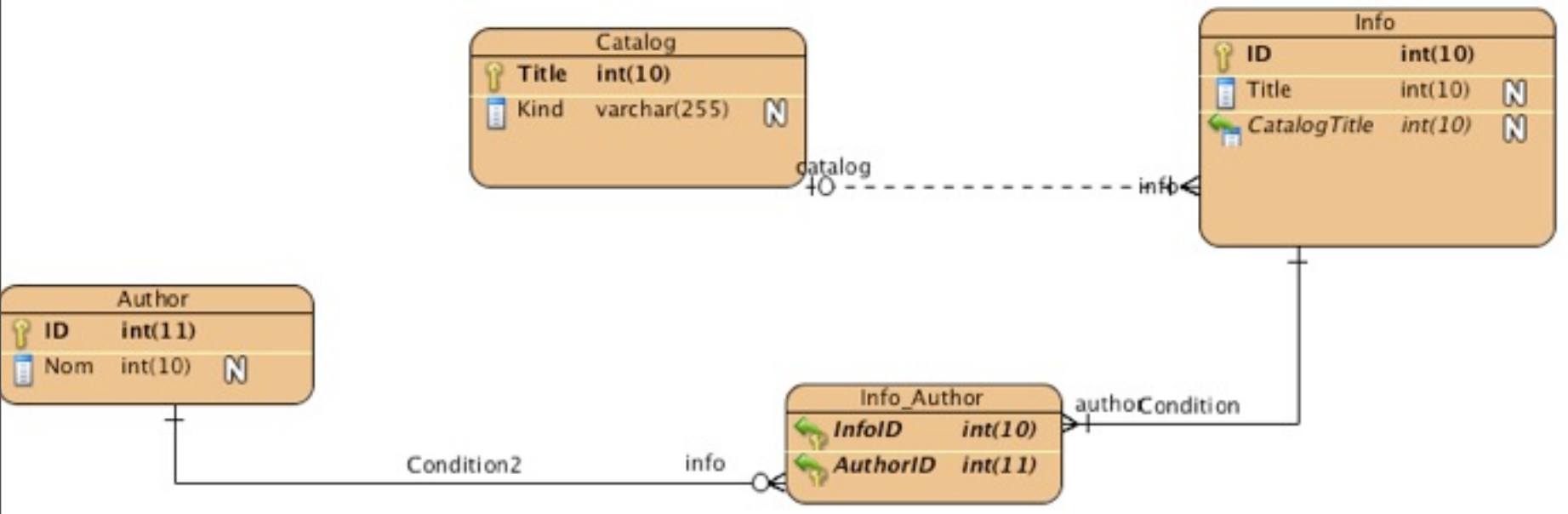
Des classes aux données

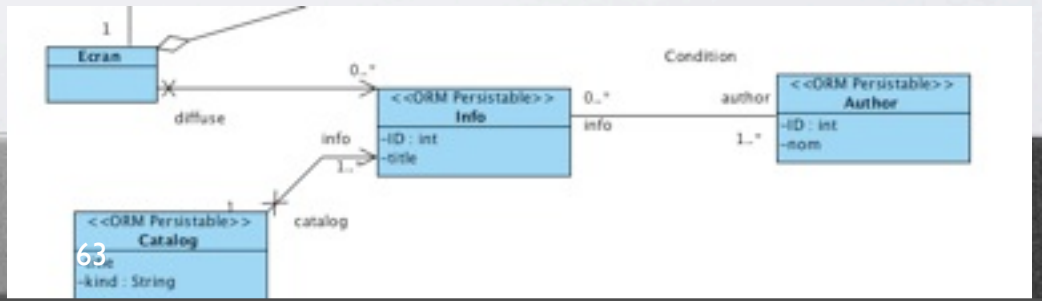


Des classes aux données

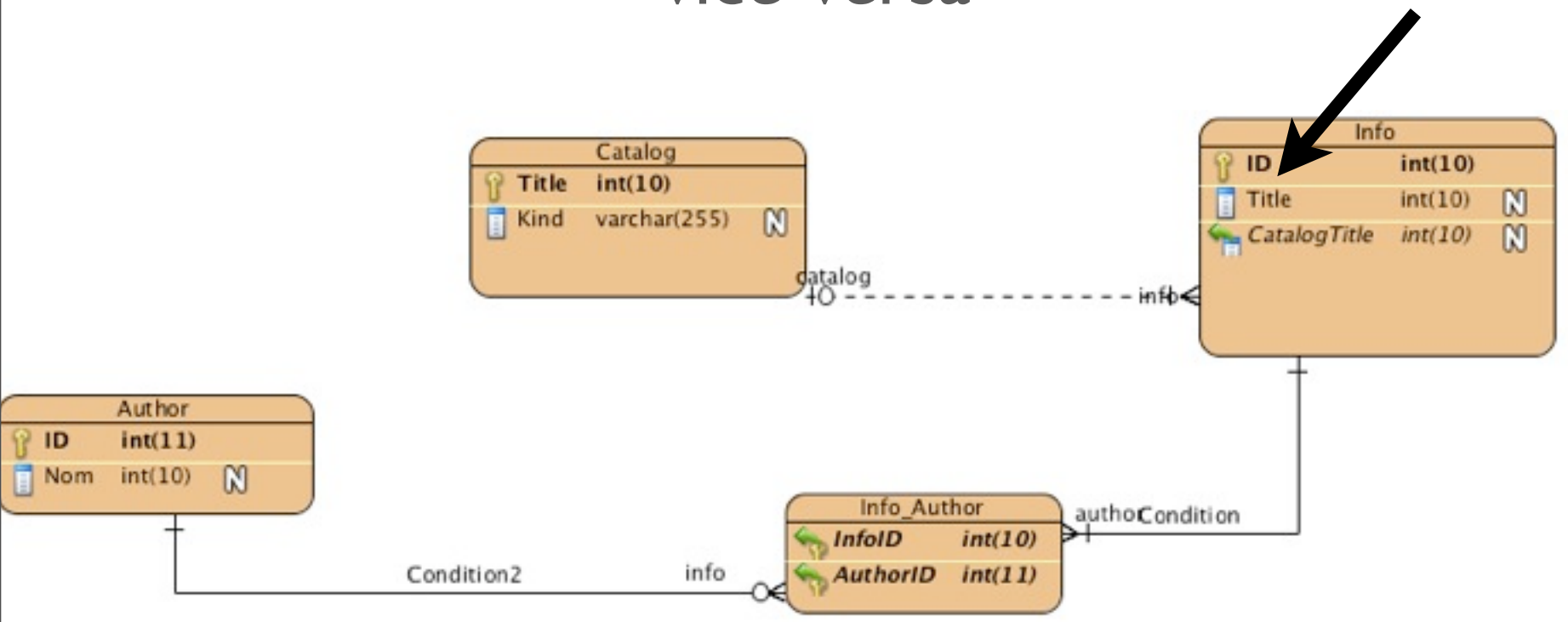


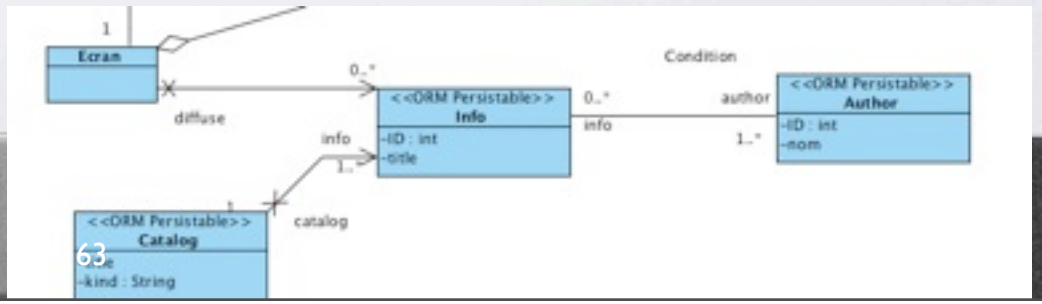
Des classes aux données (entité-relation) et vice-versa



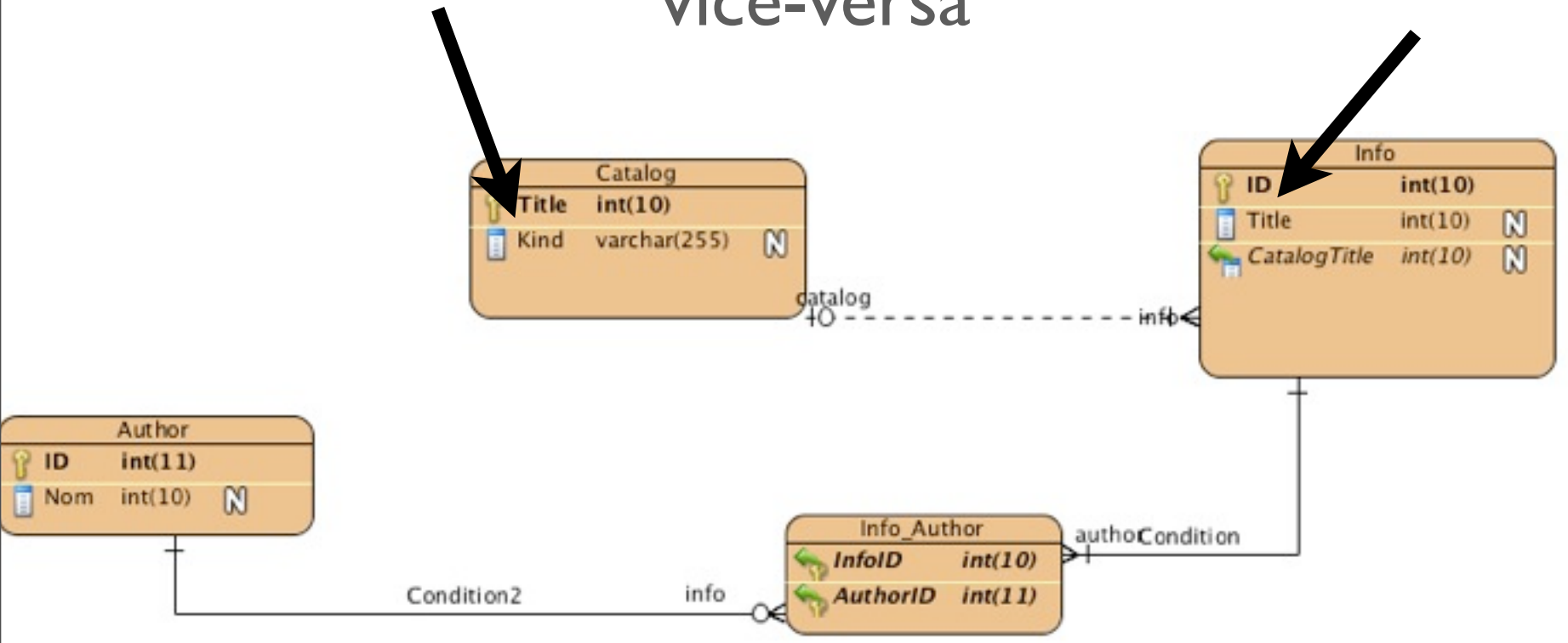


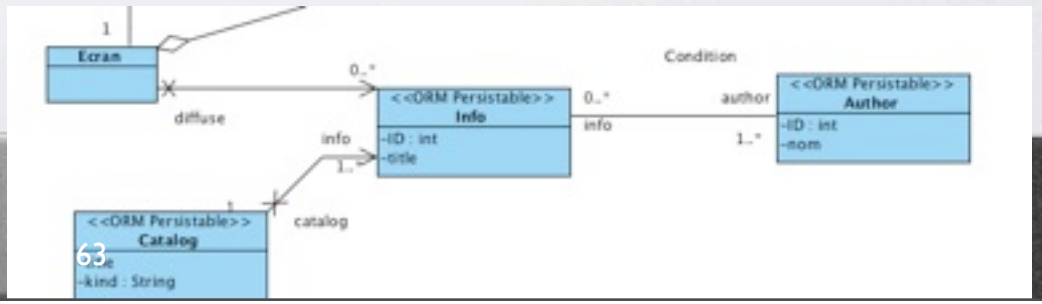
Des classes aux données (entité-relation) et vice-versa



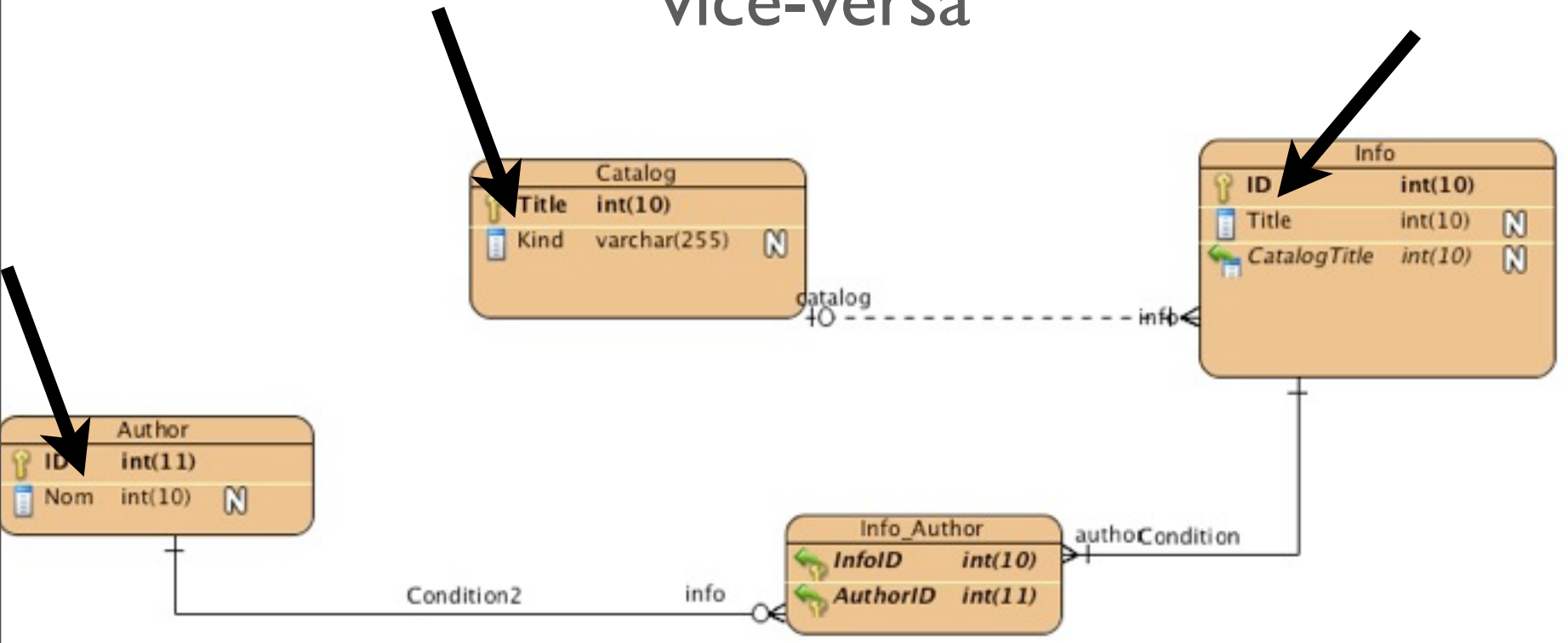


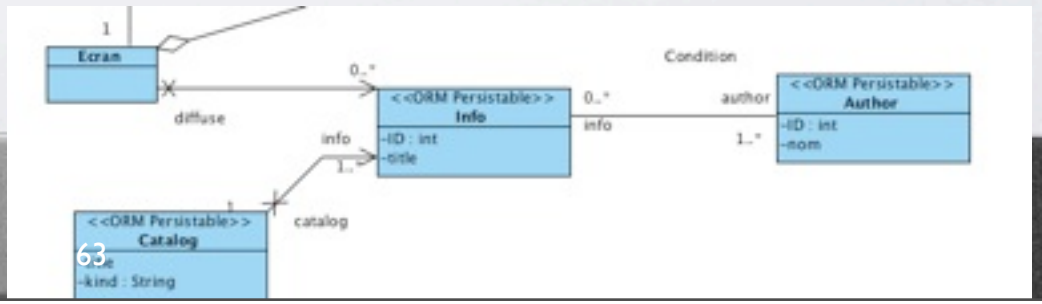
Des classes aux données (entité-relation) et vice-versa



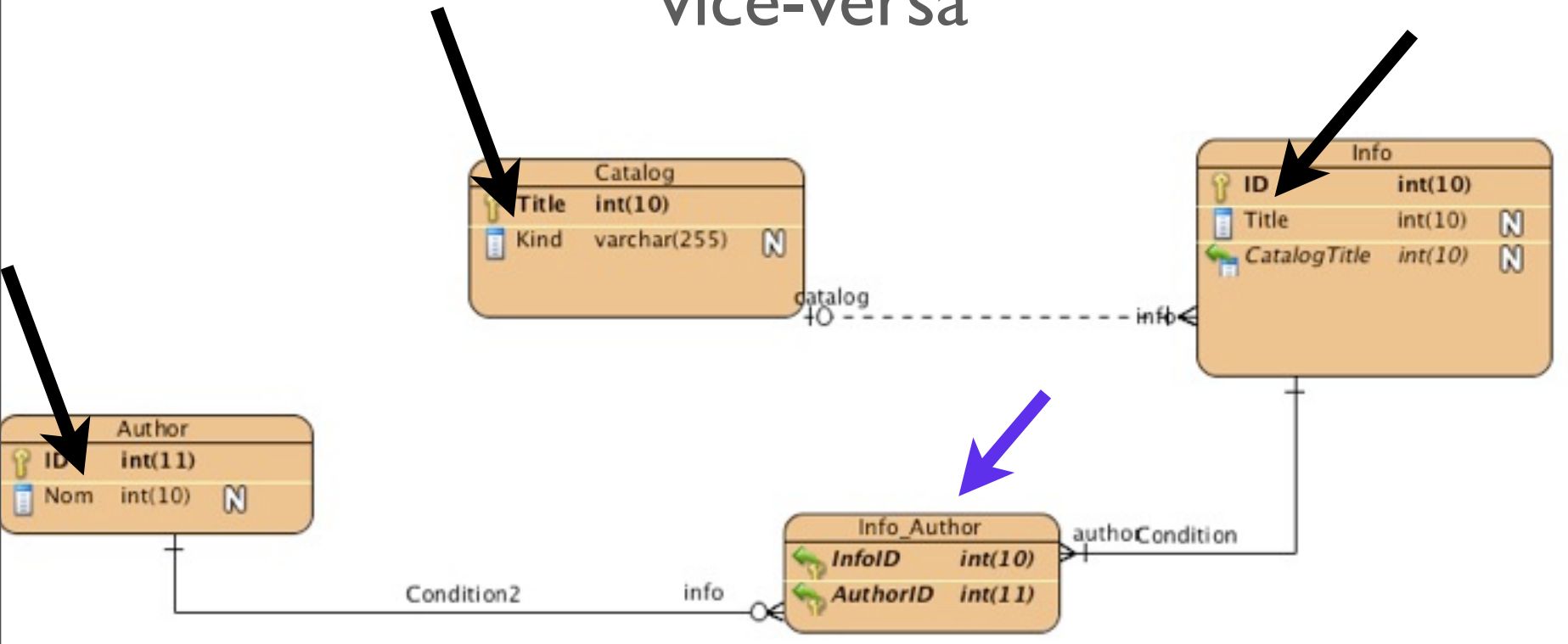


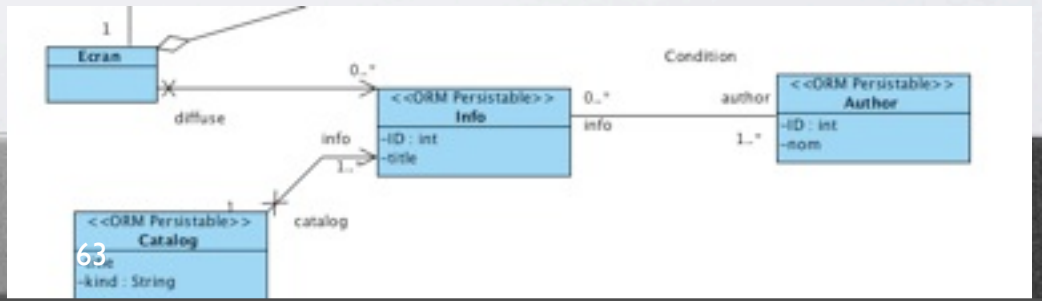
Des classes aux données (entité-relation) et vice-versa



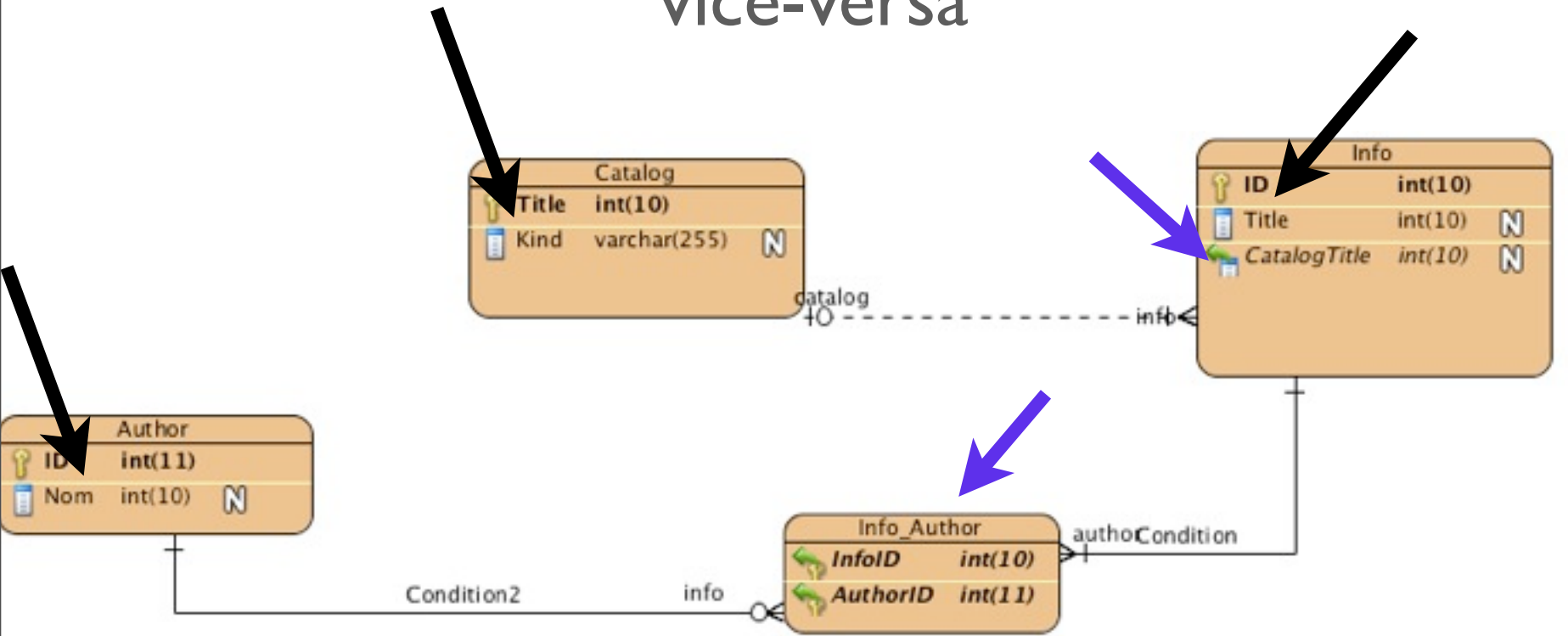


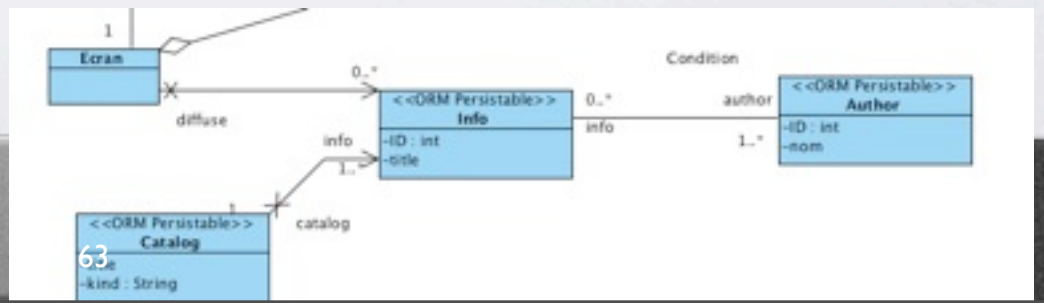
Des classes aux données (entité-relation) et vice-versa



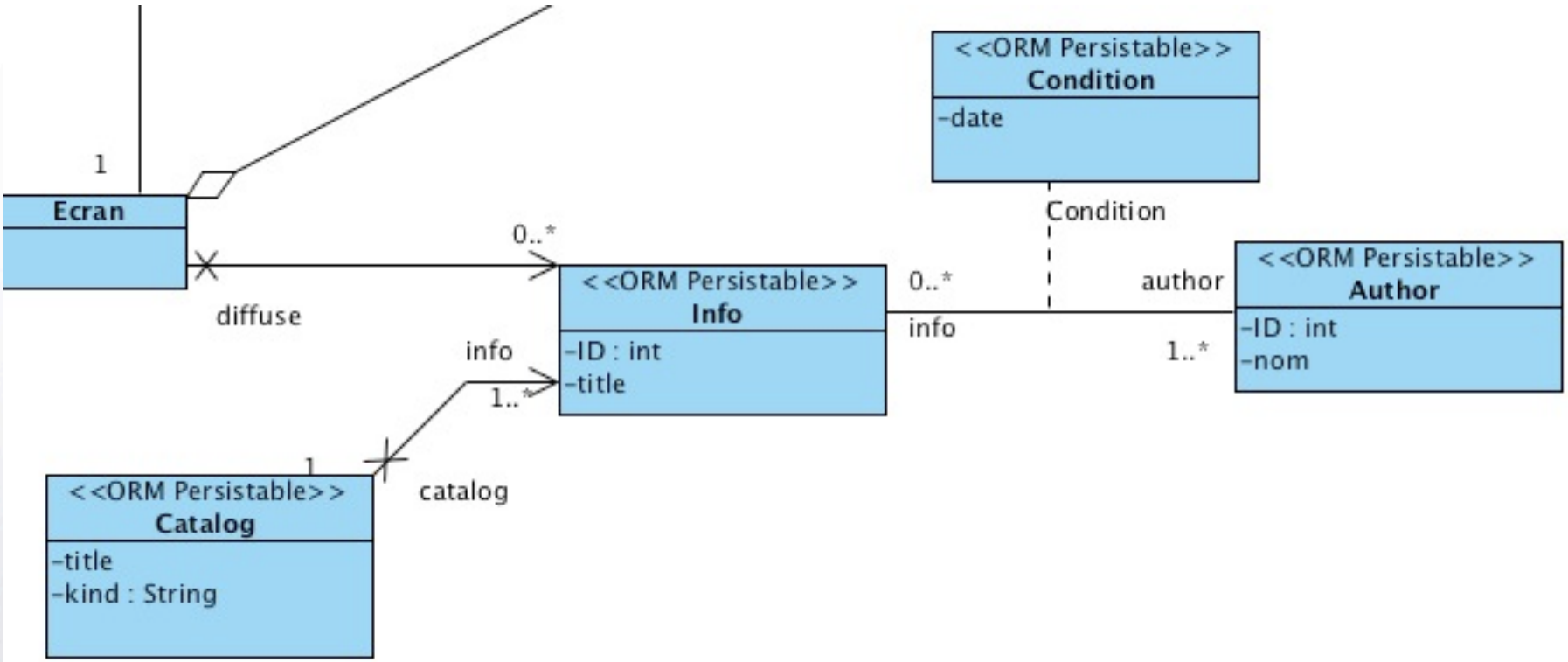


Des classes aux données (entité-relation) et vice-versa

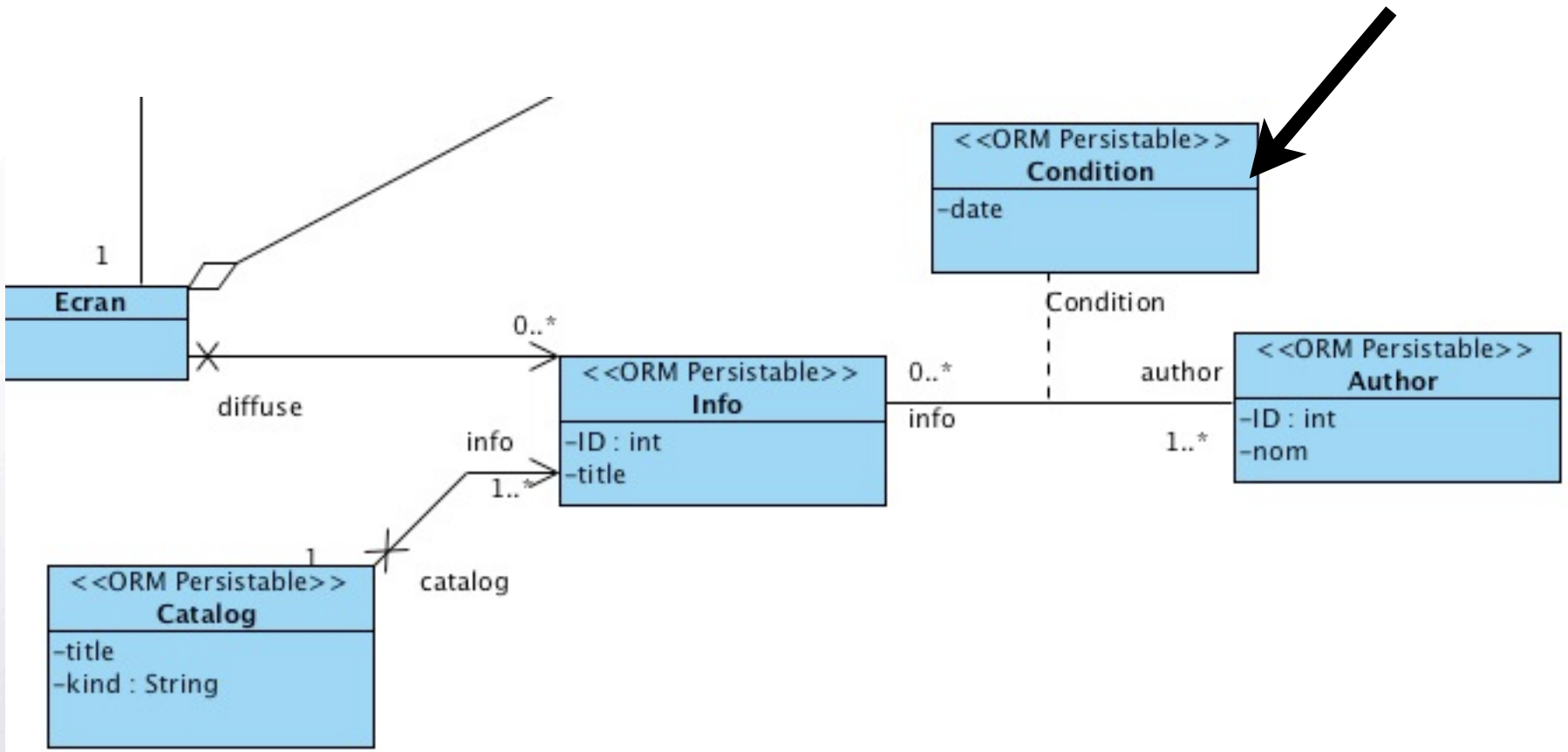




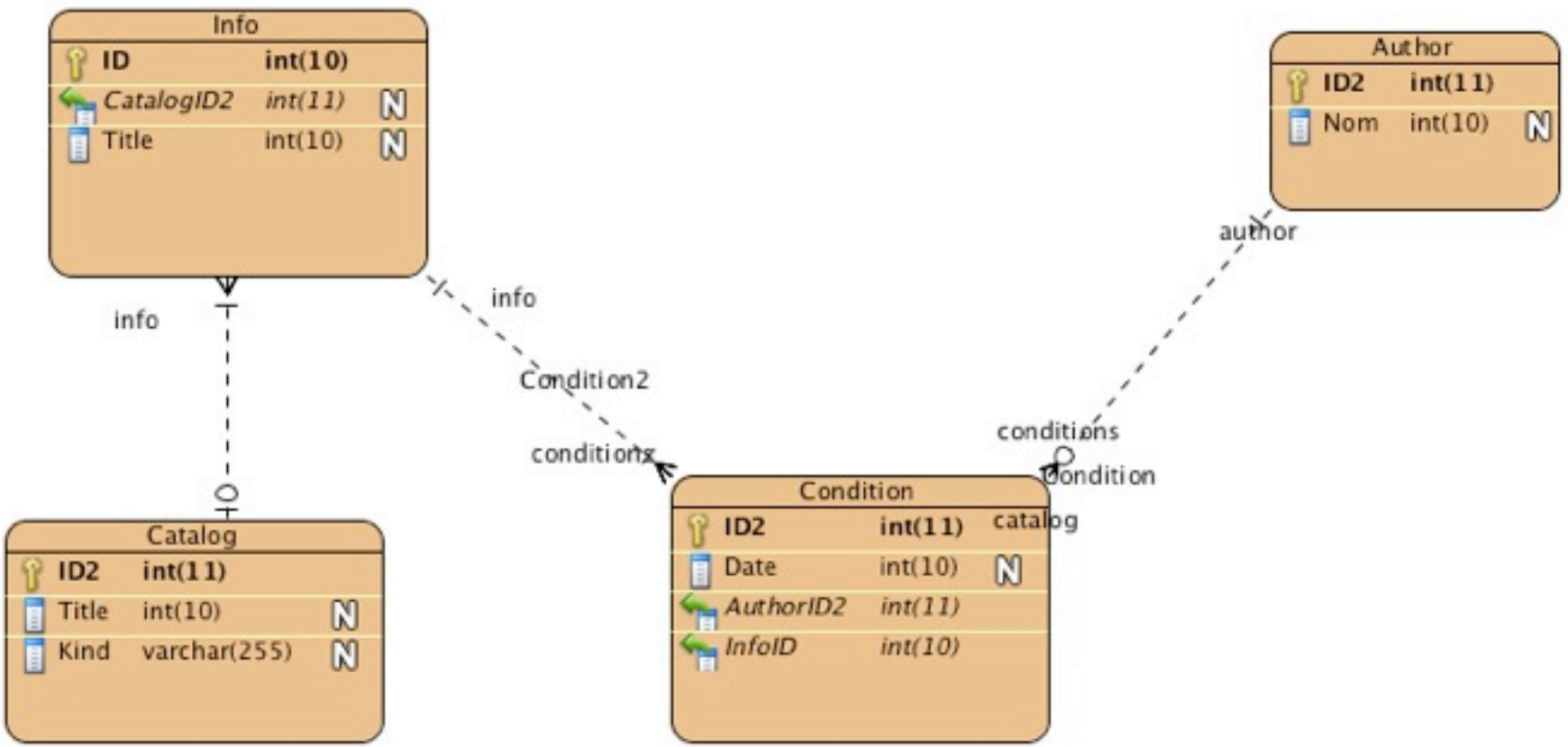
Des classes aux données



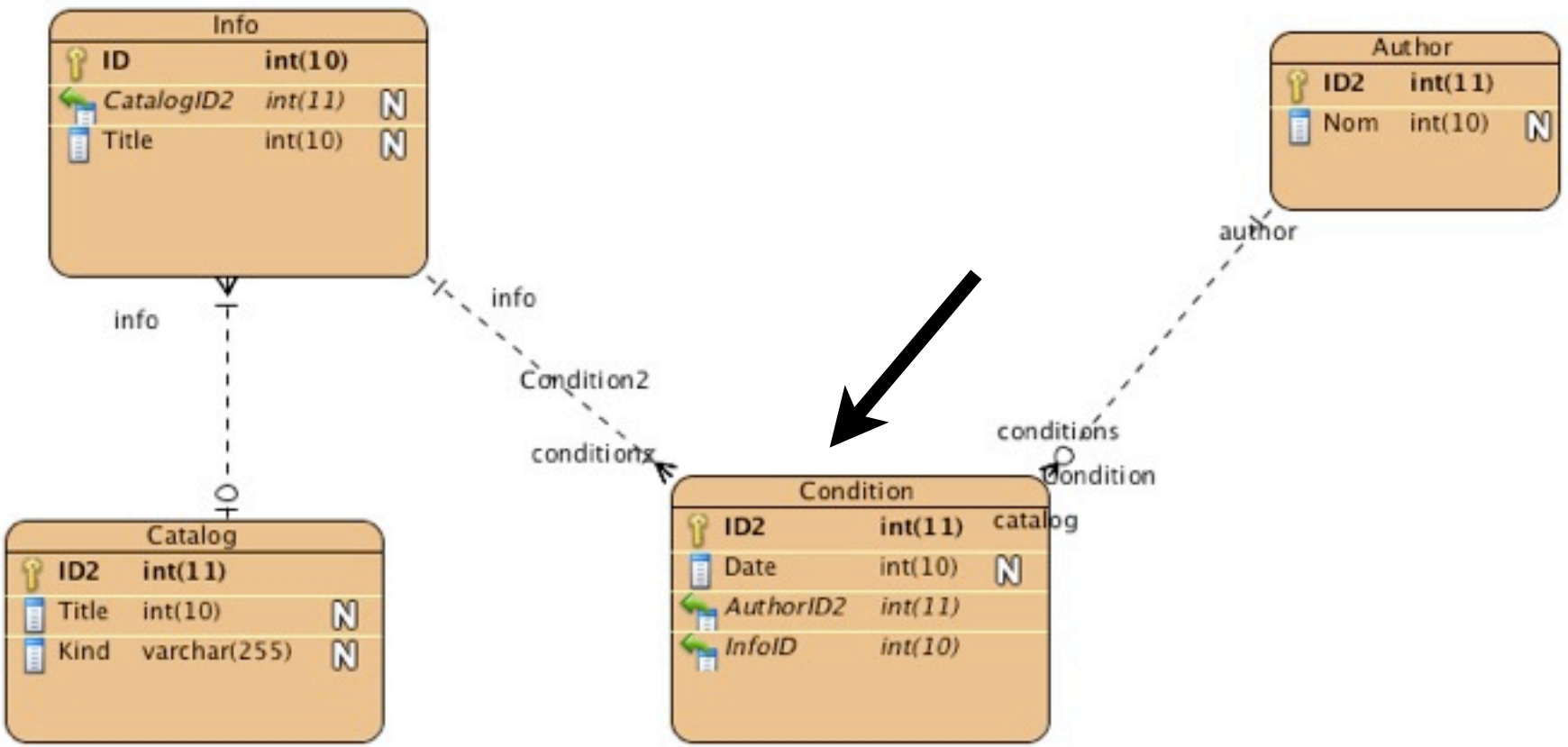
Des classes aux données



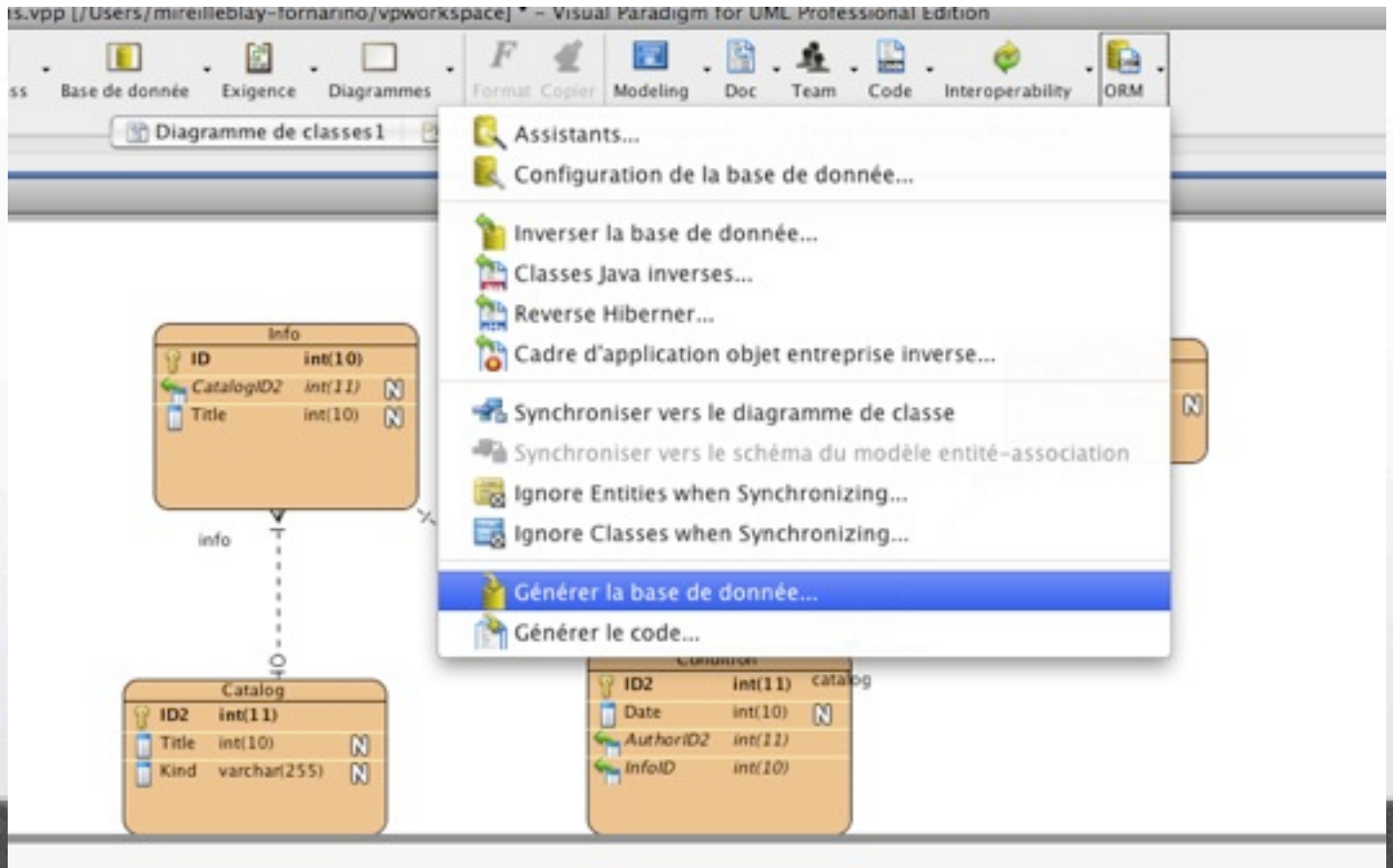
Des classes aux données



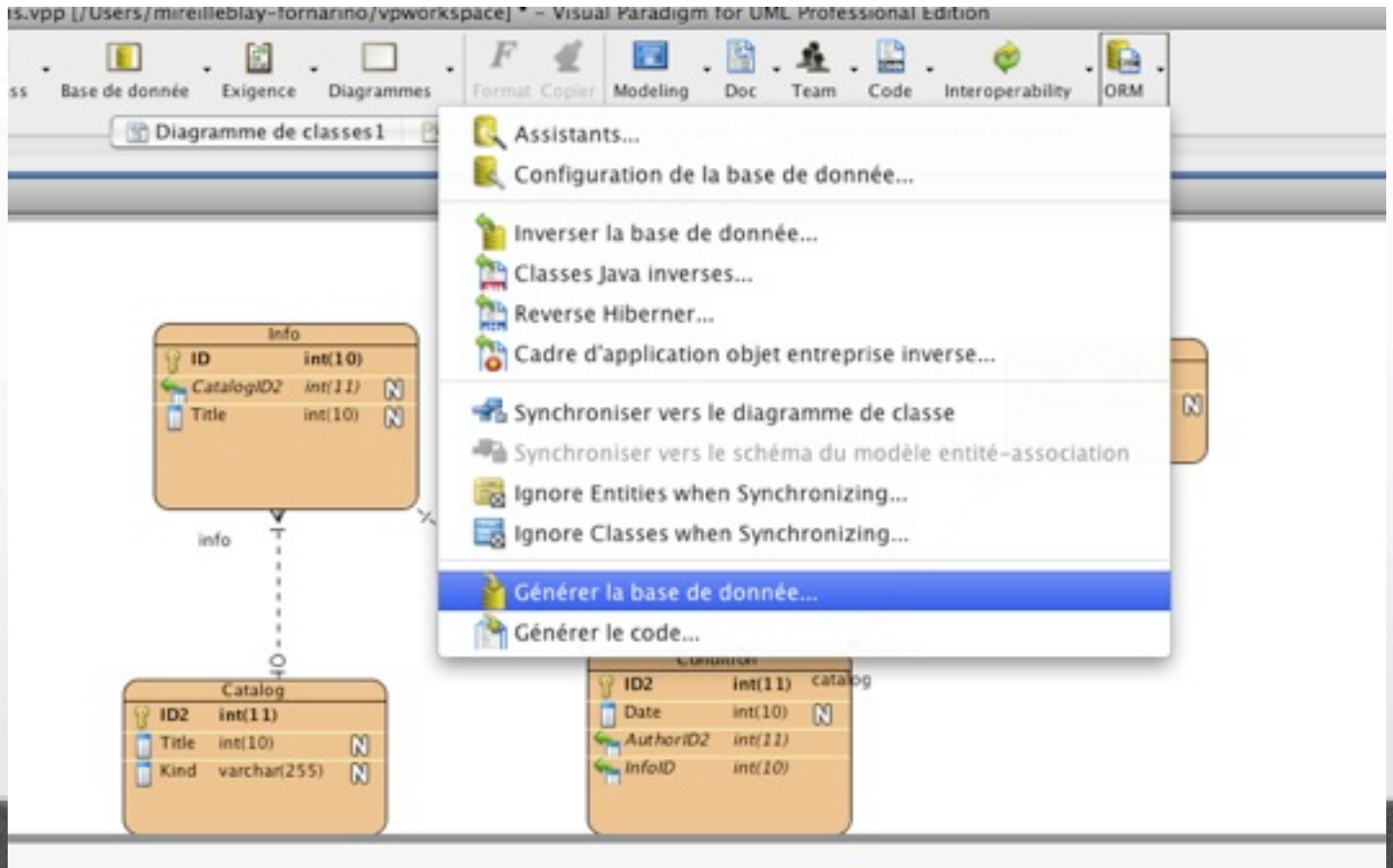
Des classes aux données



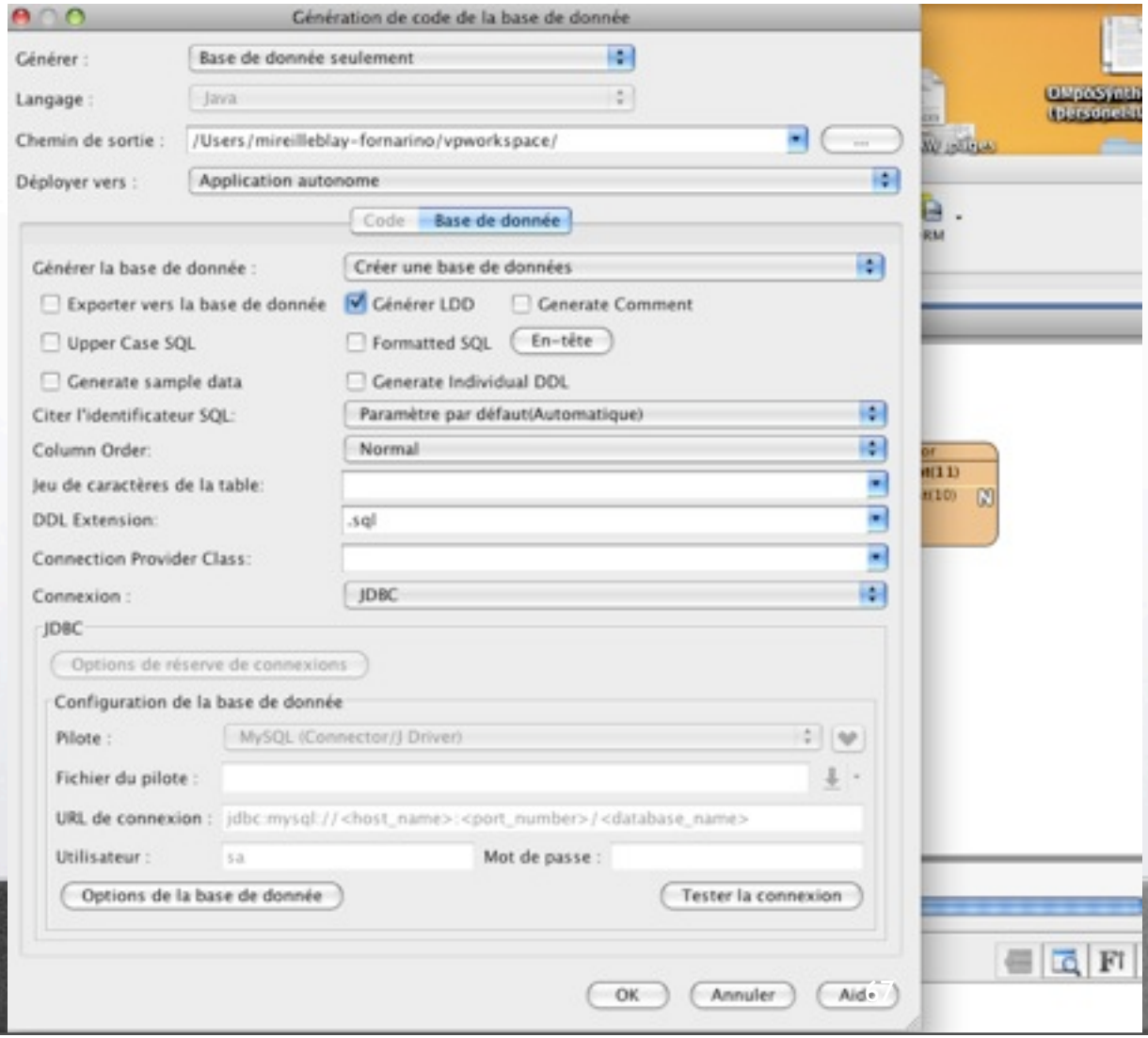
Des classes aux données



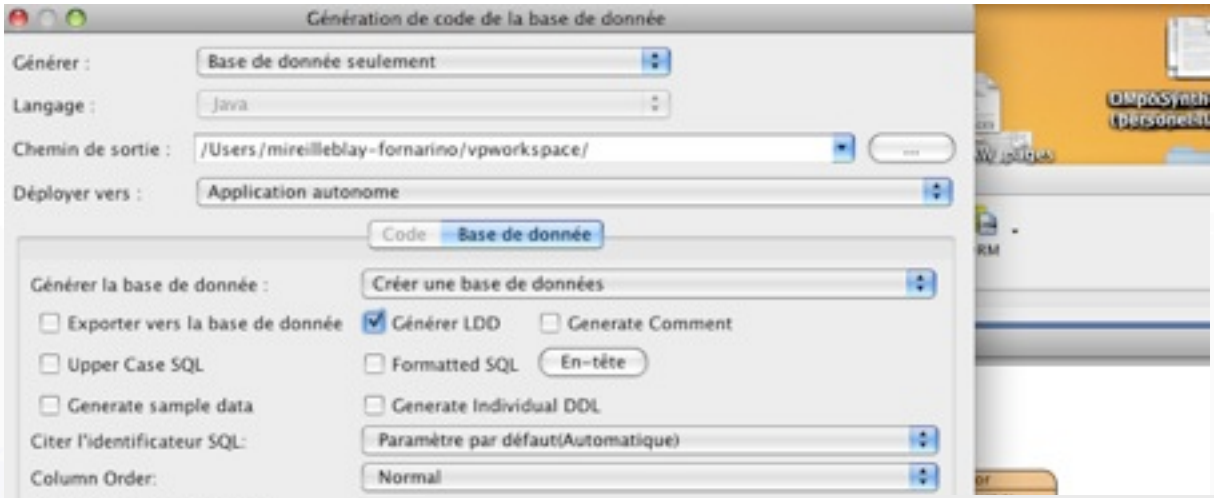
Des classes aux données



Des classes aux données



Des classes aux données




```
create table Info (ID int(10) not null auto_increment, CatalogID2 int(11), Title int(10), primary key (ID));
create table Author (ID2 int(11) not null auto_increment, Nom int(10), primary key (ID2));
create table Catalog (ID2 int(11) not null auto_increment, Title int(10), Kind varchar(255), primary key (ID2));
create table `Condition` (ID2 int(11) not null auto_increment, `Date` int(10), AuthorID2 int(11) not null, InfoID int(10) not null, primary key (ID2));
alter table `Condition` add index `Condition` (AuthorID2), add constraint `Condition` foreign key (AuthorID2) references Author (ID2);
alter table `Condition` add index Condition2 (InfoID), add constraint Condition2 foreign key (InfoID) references Info (ID);
alter table Info add index FKInfo468757 (CatalogID2), add constraint FKInfo468757 foreign key (CatalogID2) references Catalog (ID2);
```

En guise de conclusion

- L'architecture doit supporter la séparation entre métier, interfaces homme-machine et données.
- MVC, DAO, Observer, ... sont des patrons pour guider les mises en oeuvre.