Technologies Web avancées

Technologies Web avancées

Olivier Pons / 2014 - 2015

Objectif

Appréhender trois outils complexes

Sommaire

1. Rappel MVC
2. ExtJS
3. NodeJS
4. NoSQL
5. ExtJS + NodeJS + MongoDB

1 – MVC

a – Théorie

MVC

Model View Controller

Modèle – vue – contrôleur

Les grands principes

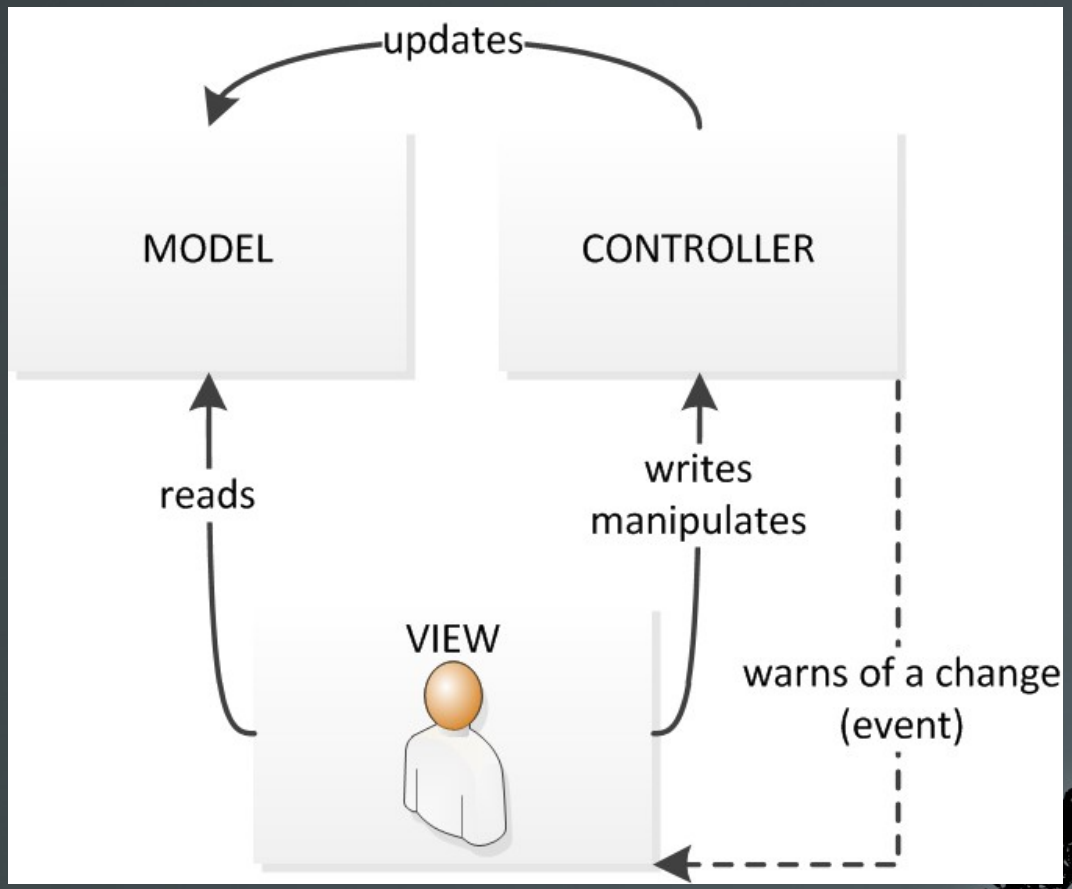
1 – MVC

a – Théorie

Modèle	Modèle de données
Vue	Présentation, interface utilisateur
Contrôleur	Logique de contrôle Gestion des événements Synchronisation

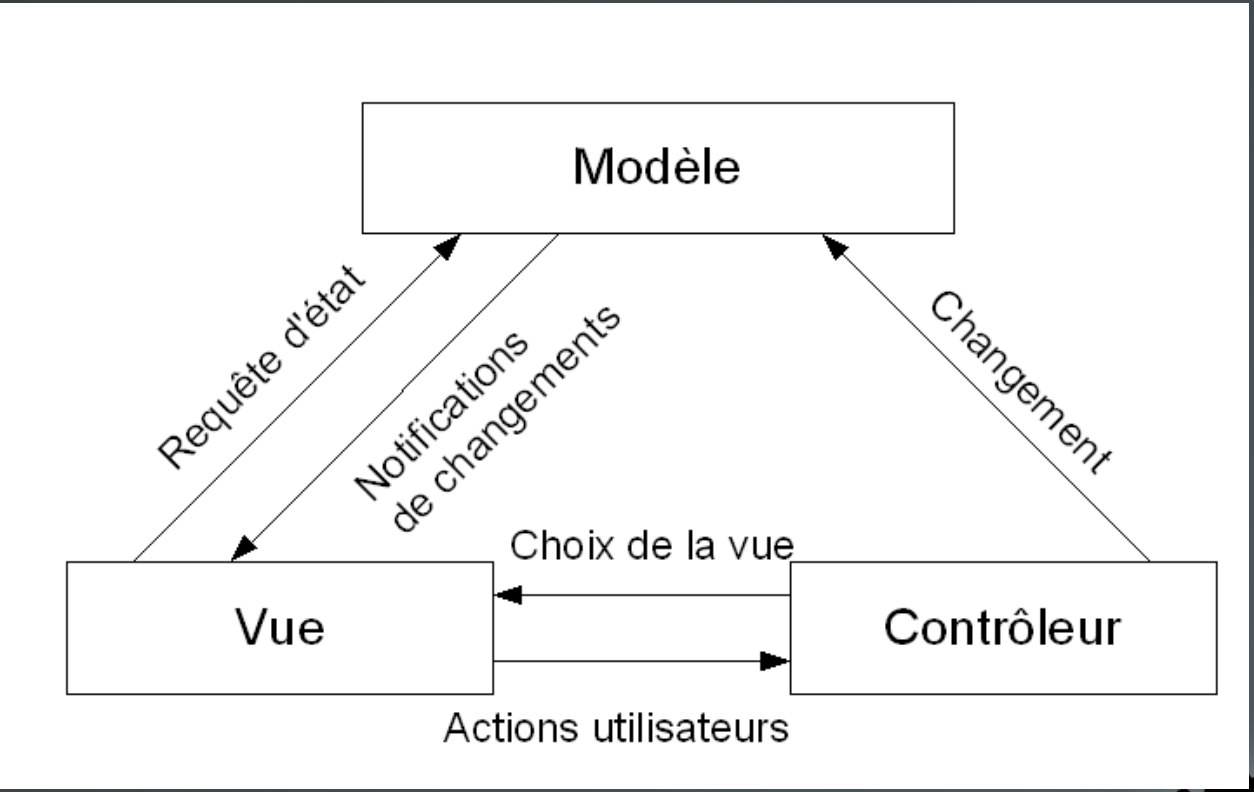
1 – MVC

a – Théorie



1 – MVC

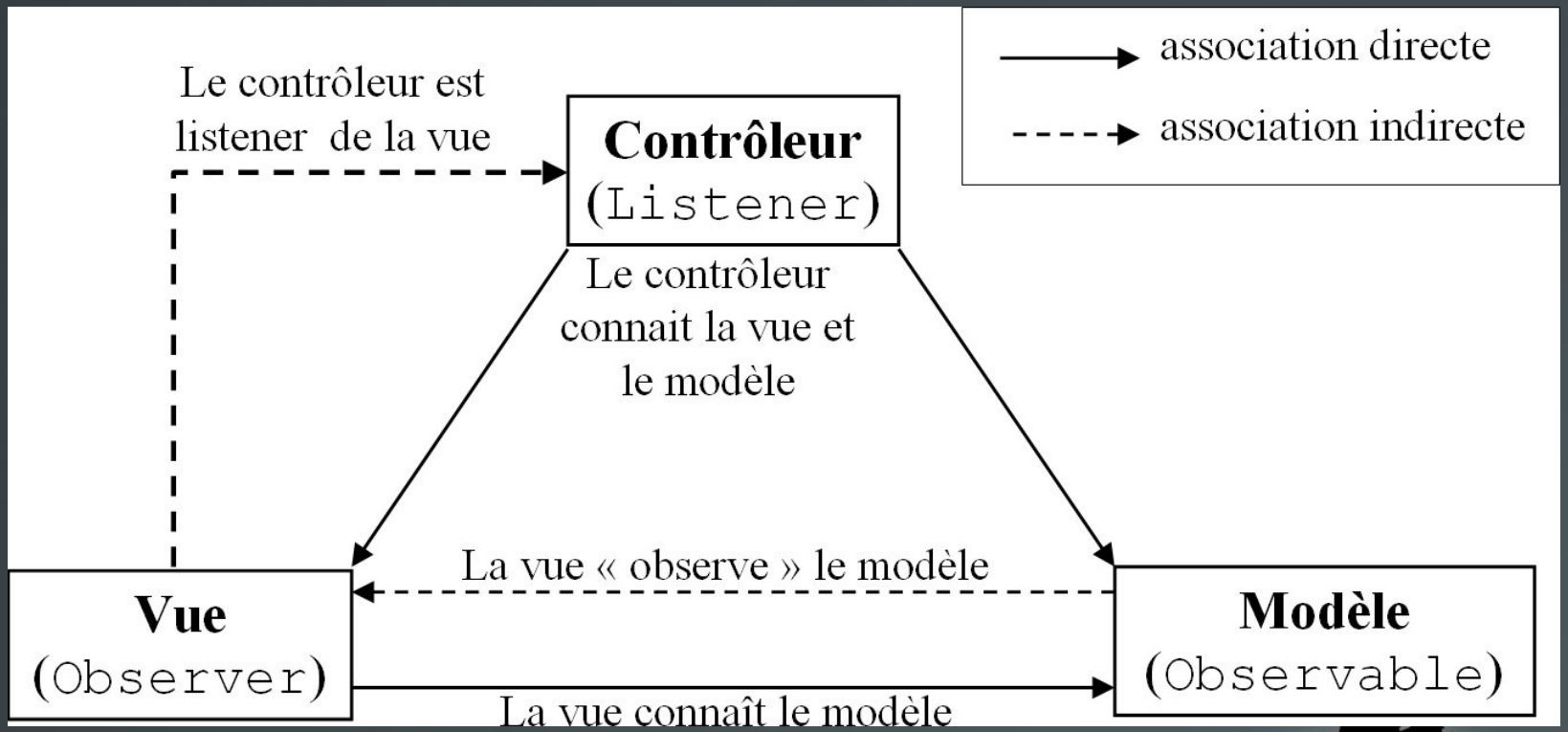
a – Théorie



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

1 – MVC

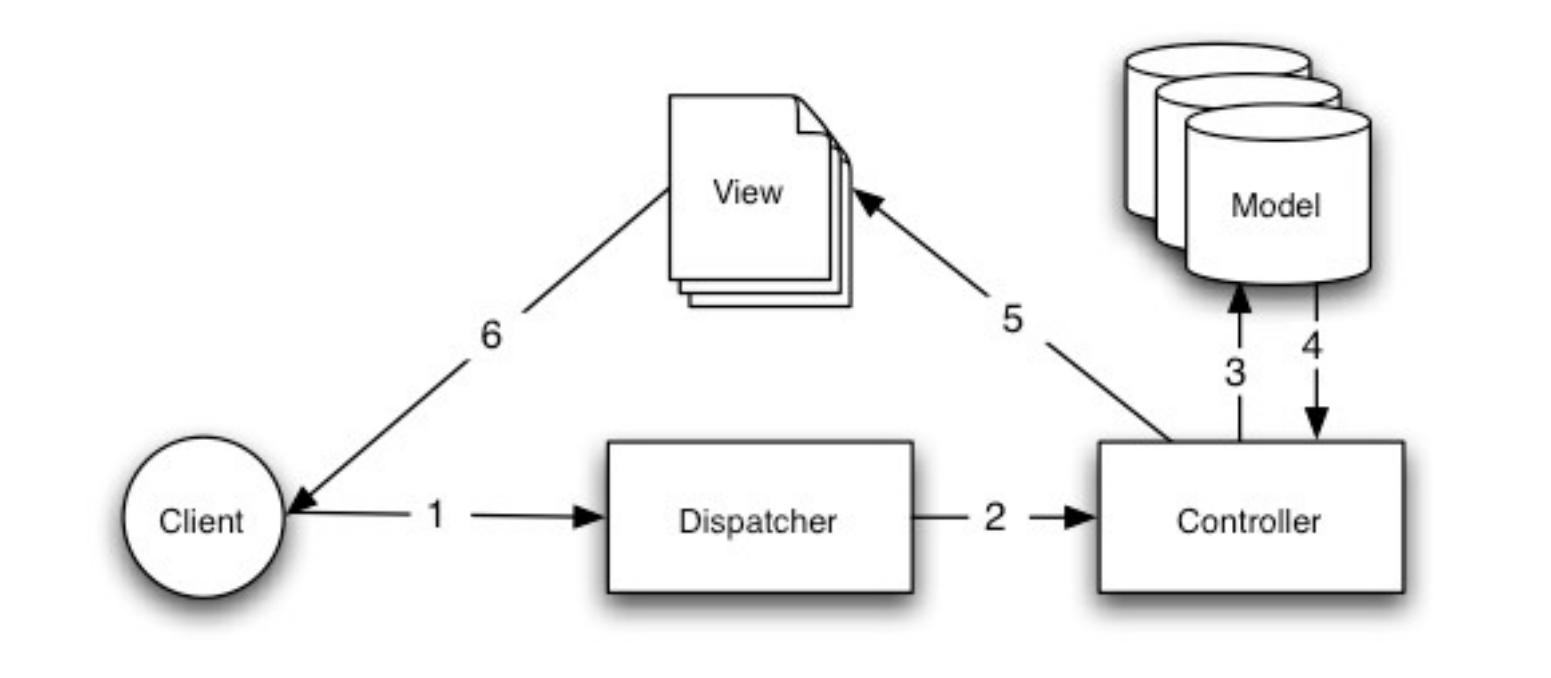
a – Théorie



http://perso.telecom-paristech.fr/~hudry/coursJava/interSwing/boutons5.html

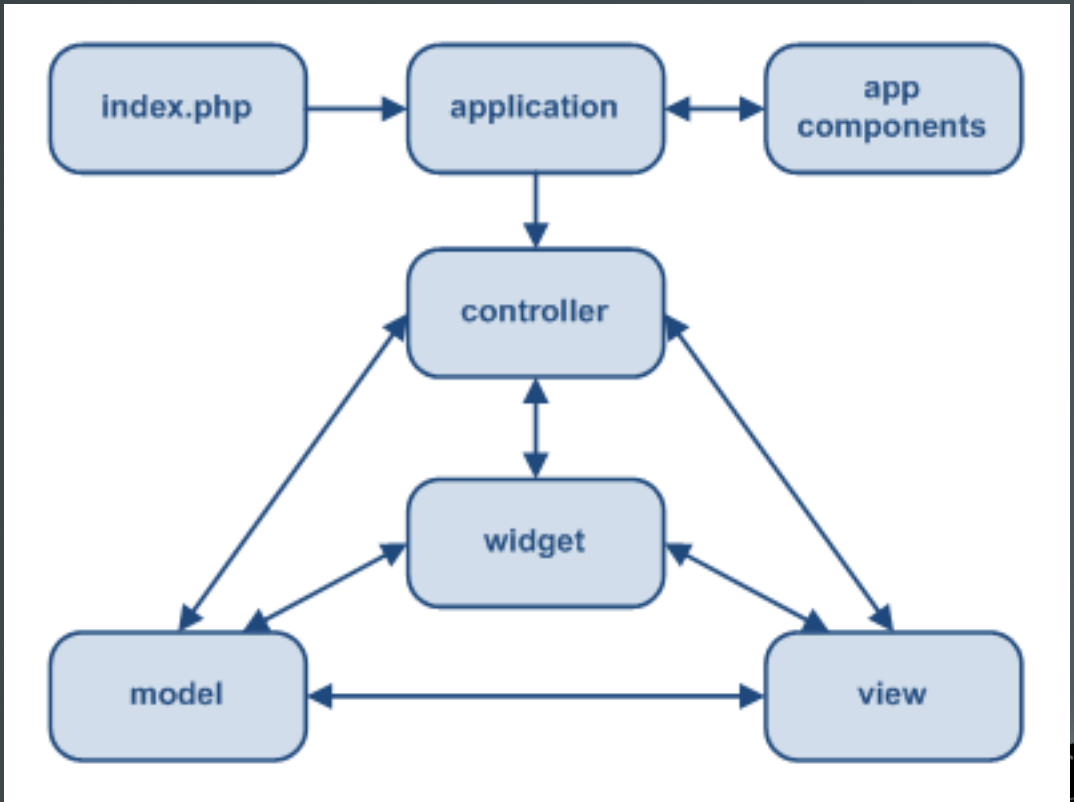
1 – MVC

b – Exemples – Framework CakePHP



1 – MVC

b – Exemples – Framework YII



1 – MVC

b – Exemples – Modèle

```
class Partenaire {
   private $_nom;

   __constructor () {
      $this->_nom = '';
   }

   public int getNom() {
      return $this->_nom;
   }

   public void setNom(int $nom) {
      $this->_nom = $nom;
   }
}
```

1 – MVC

b – Exemples – Symfony 2 – Routing

```
hqf_pizzas_towns_detail:
    pattern:  /ville/{url}/{cp}
    defaults: { _controller: HQFPizzasBundle:Default:townsDetail }
    requirements:
        url:   "[a-z0-9-]+"
        cp:    "^([0-9]{5}|2[a|b]){1,2}$"

hqf_pizzas_partenaire_inscription:
    pattern:  /partenaire/inscription
    defaults: { _controller:
                    HQFPizzasBundle:Default:partenaireInscription }

hqf_pizzas_partenaire_inscription_ok:
    pattern:  /partenaire/inscription/ok
    defaults: { _controller:
                    HQFPizzasBundle:Default:partenaireInscriptionOk }
```

1 – MVC

b – Exemples – Symfony 2 – Routing

```
_controller: HQFPizzasBundle:Default:XxxYyy
```

Décomposition du chemin :

```
HQFPizzasBundle
=>
HQF – PizzasBundle
=>
src/HQF/Bundle/PizzasBundle

_controller + Default
=>
/Controller/DefaultController.php
```

Les deux mis ensemble :

```
_controller: HQFPizzasBundle:Default:XxxYyy
src/HQF/Bundle/PizzasBundle/Controller/DefaultController.php
```

1 – MVC

b – Exemples – Symfony 2 – Contrôleur

```
public function partenaireSansIdAction($url)
{
    $em = $this->get('doctrine')->getManager('ma_bd');
    $repo_partenaire =
        $em->getRepository('HQFPizzasBundle:Partenaire');

    $partenaires = $repo_partenaire
        ->findAllActiveByURL($url)
        ->getQuery()
        ->getResult();

    return $this->render(
        'HQFPizzasBundle::partenaire.html.twig',
        array('partenaires' => $partenaires)
    );
}
```

1 – MVC

b – Exemples – Symfony 2 – Vue

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0" />
    <title>{% block title %}Titre par défaut</title>
</head>
<body>
    {% for partenaire in partenaires %}
        <h2>{{ partenaire.nom }}</h2>
    {% endfor %}
</body>
</html>
```

2 – ExtJS

a – Sencha

2007	**2009**	**2011**	**2014**
V2	V3	V4	V5
RIA / JavaScript	REST	Class	MVVM
Documentation++	Diagrammes	MVC	Responsive
APIs + exemples	ListView	Animation	Routing

2 – ExtJS

a – Sencha – Clients

2 – ExtJS

a – Sencha – Distribution

GPL v3 = Public releases:

- Sencha Touch
- Sencha Touch Charts
- Sencha Ext JS
- Sencha GXT

Sencha commercial licence:

- Sencha Touch Grid
- Sencha Architect
- Sencha Cmd

2 – ExtJS

a – Sencha

GPL v3

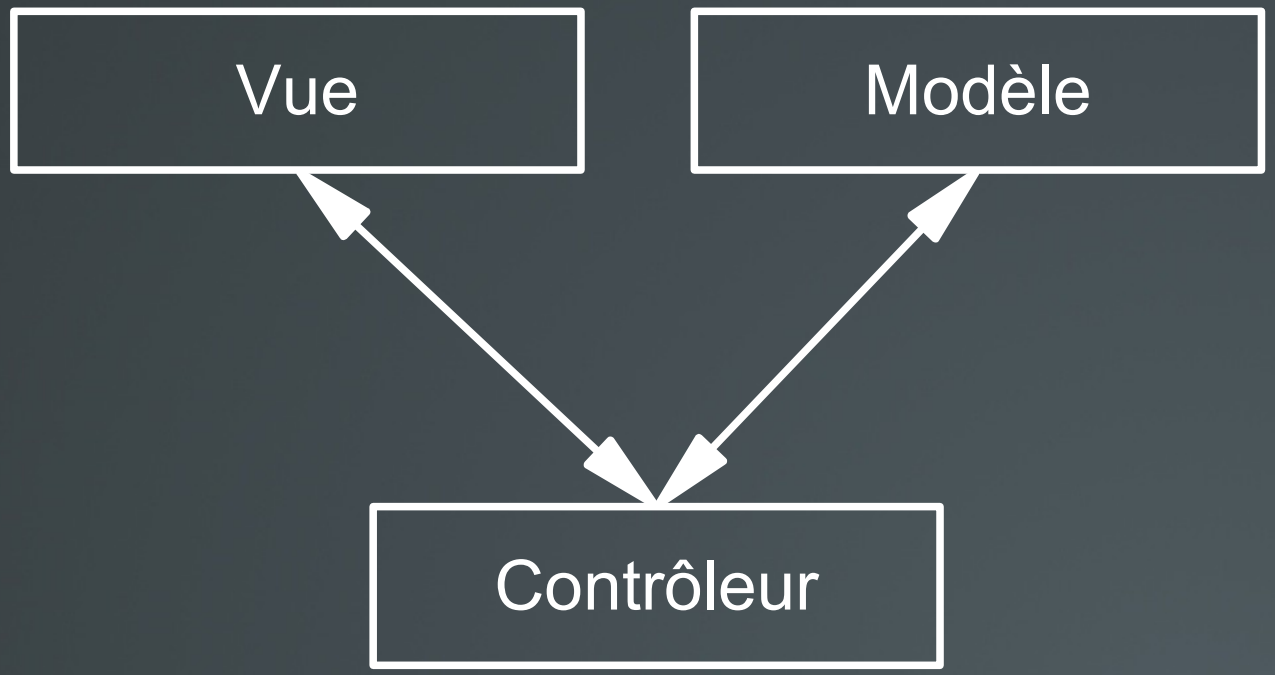
Personne ne doit être limité par les logiciels qu'il utilise.
Il y a quatre libertés que tout utilisateur doit posséder.
La liberté :

- d'utiliser le logiciel à n'importe quelle fin,
- de modifier le programme pour répondre à ses besoins,
- de redistribuer des copies à ses amis et voisins,
- de partager avec d'autres les modifications qu'il a faites.

http://www.gnu.org/licenses/quick-guide-gplv3.html

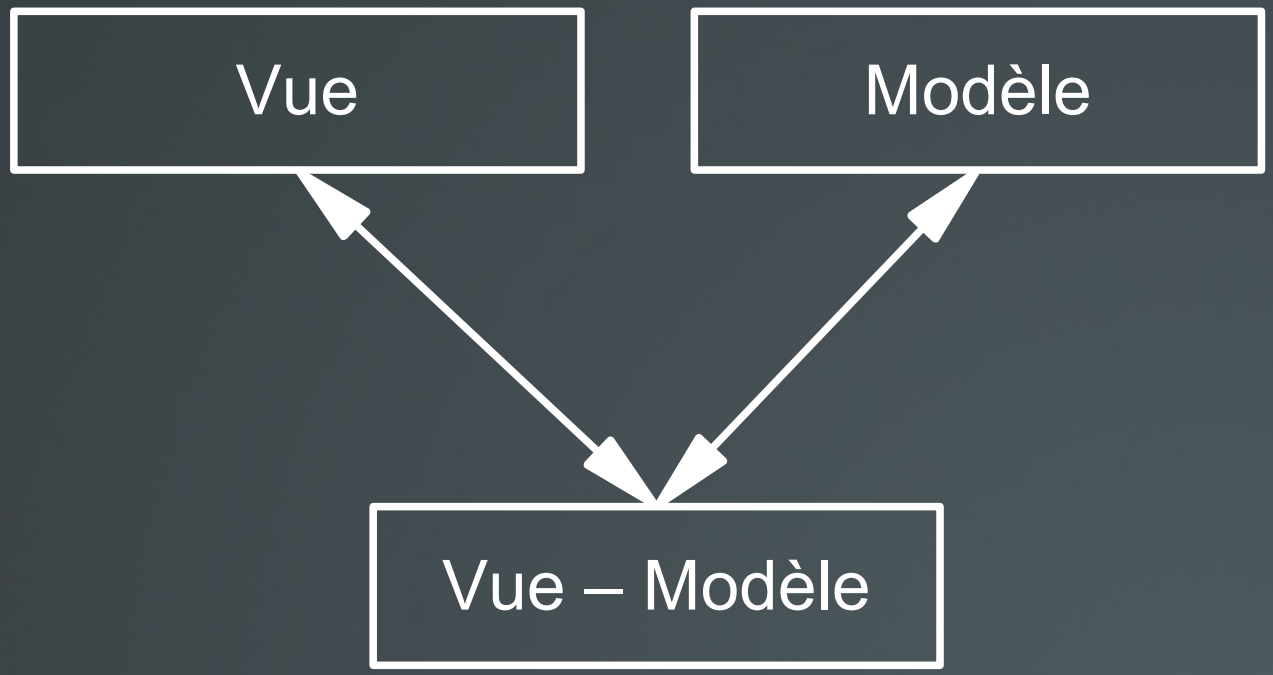
2 – ExtJS

b – ExtJS – MVC



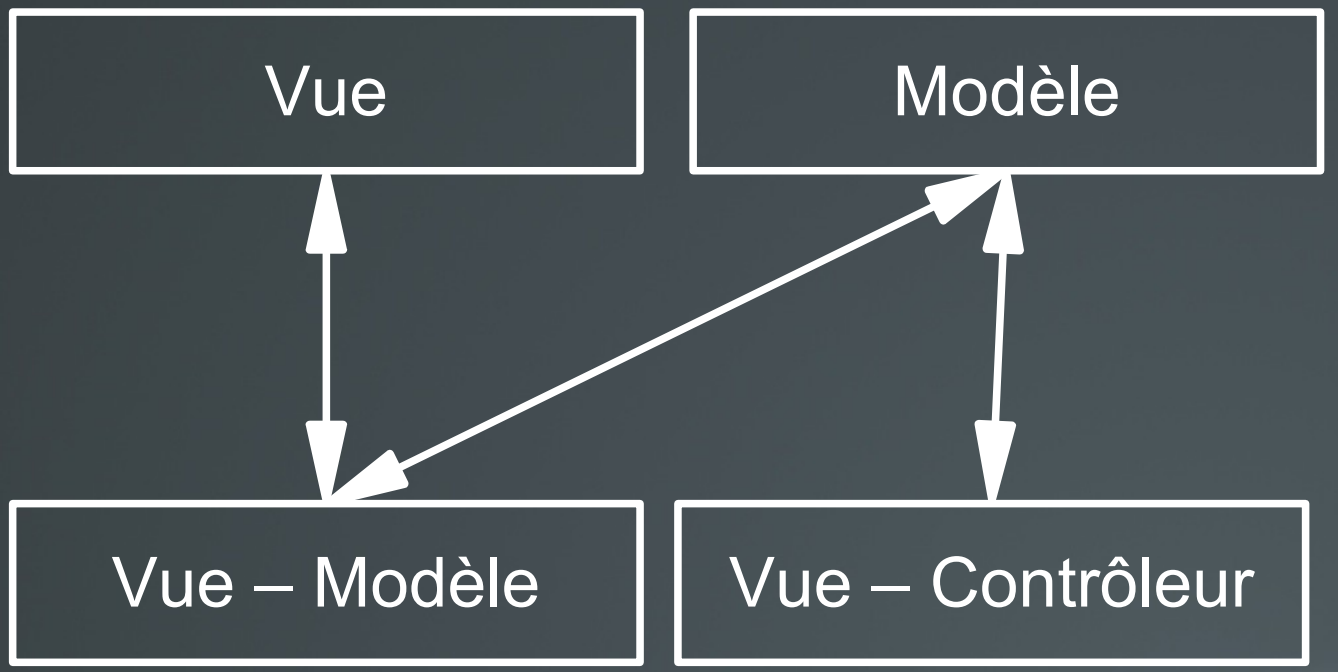
2 – ExtJS

b – ExtJS – MVVM



2 – ExtJS

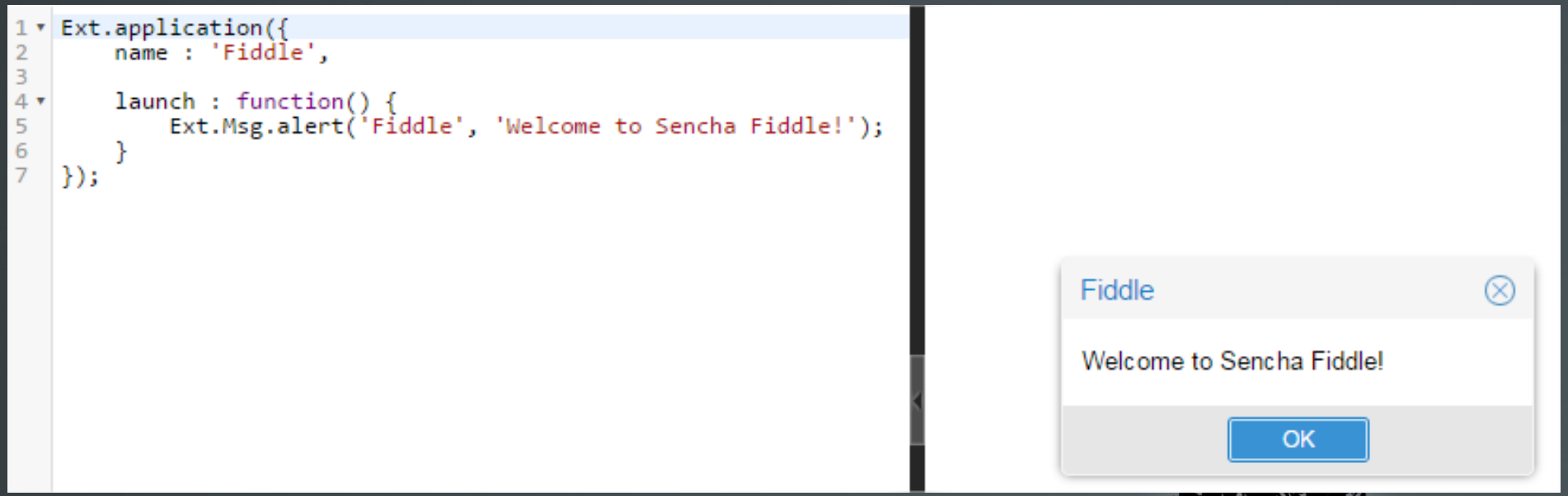
b – ExtJS – MVVM



2 – ExtJS

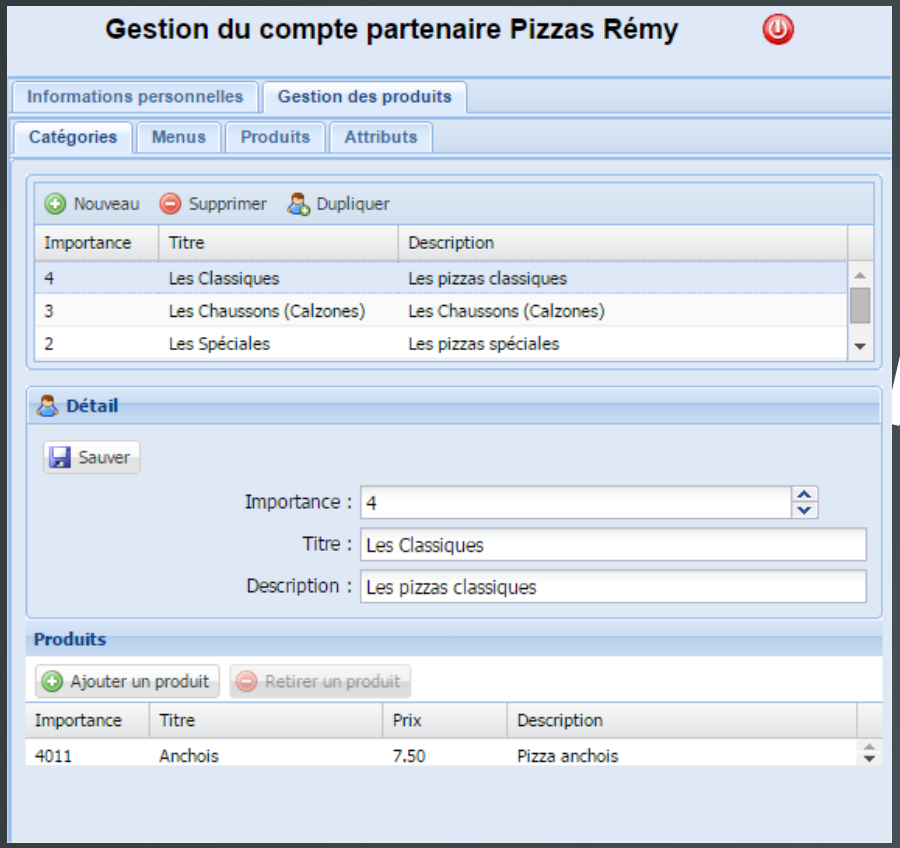
b – ExtJS – Pratique

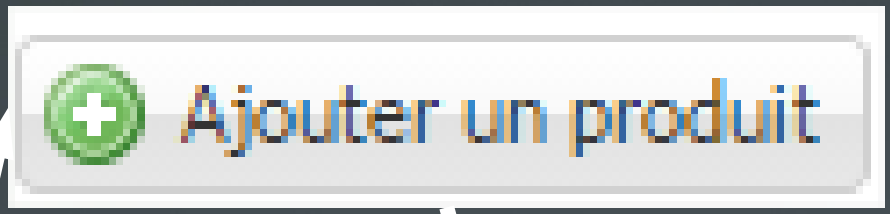
https://fiddle.sencha.com/#home

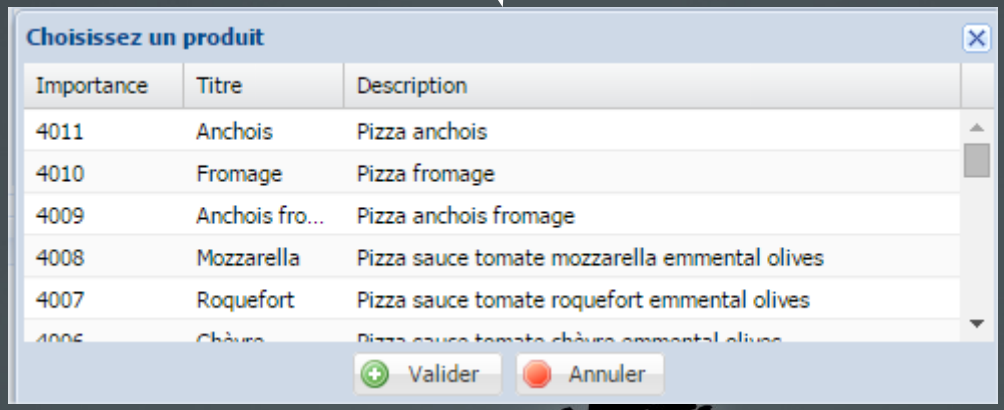


2 – ExtJS

b – ExtJS – Pratique







2 – ExtJS

b – Classes

Chargement dynamique des classes

```
Ext.require(['monenv.A'], function() {

    /* fonction à executer une fois chargé */

});
```

2 – ExtJS

b – Classes

```
Ext.define('Gestion.Categorie', {
    extend: 'Ext.data.Model',
    fields: [{
            name: 'id',
            type: 'int',
            useNull: true
        },
        'id_partenaire',
        'importance',
        'titre',
        'description'
    ],
    hasMany: {
        model: 'Gestion.Produit',
        name: 'produits'
    }
});
```

```
Ext.define('Gestion.Produit', {
    extend: 'Ext.data.Model',
    fields: [{
            name: 'id',
            type: 'int',
            useNull: true
        },
        'id_partenaire',
        'importance',
        'titre',
        'description',
        'prix'
    ],
    hasMany: {
        model: 'Gestion.Attribut',
        name: 'attributs'
    },
    belongsTo: 'Gestion.Categories'
});
```

2 – ExtJS

b – Classes

```
Ext.define('Gestion.Produit', {
    extend: 'Ext.data.Model',
    fields: [{
            name: 'id',
            type: 'int',
            useNull: true
        },
        'id_partenaire',
        'importance',
        'titre',
        'description',
        'prix'
    ],
    hasMany: {
        model: 'Gestion.Attribut',
        name: 'attributs'
    },
    belongsTo: 'Gestion.Categories'
});
```

```
Ext.define('Gestion.Attribut', {
    extend: 'Ext.data.Model',
    fields: [{
            name: 'id',
            type: 'int',
            useNull: true
        },
        'id_partenaire',
        'importance',
        'description',
        'valeur',
        'abbreviation'
    ],
    belongsTo: 'Gestion.Produit'
});
```

2 – ExtJS

b – Classes

Attention

```
Ext.define('verroumortel.A', {
    extend: 'verroumortel.B',
});

Ext.define('verroumortel.B', {
    extend: 'verroumortel.C',
});

Ext.define('verroumortel.C', {
    extend: 'verroumortel.A',
});
```

2 – ExtJS

c – ExtJS – TP

MVVM

Relation maître détail :

- grille qui liste des champs qui sont recherchés en AJAX

- quand on clique sur un champ

- détail affiché dans un tableau

3 – NodeJS

a – Rappel JavaScript

Bases JavaScript

Portée des variables

Closures

Fonctions anonymes

Faits en cours

3 – NodeJS

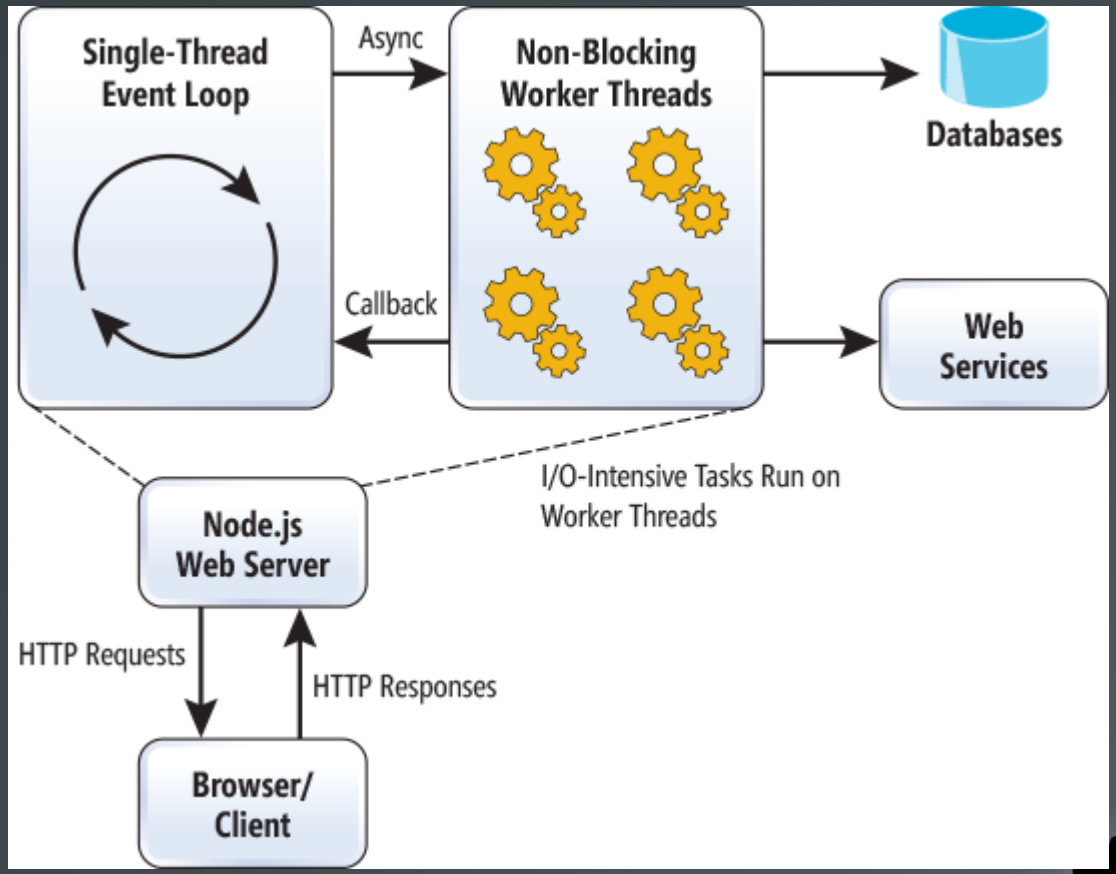
a – Principes synchrone / asynchrone

Avantages et inconvénients

Faits en cours

3 – NodeJS

a – Principes synchrone / asynchrone



http://msdn.microsoft.com/fr-fr/magazine/dn754378.aspx

3 – NodeJS

b – Standardisation SQL / Nouveaux besoins

Faits en cours

3 – NodeJS

b – TP

Microsoft Visual Studio

Client + serveur synchrone

Client + serveur asynchrone

3 – NodeJS

b – TP – Tests MVC

sailsjs.org
www.totaljs.com
www.partialjs.com
koajs.com
locomotivejs.org
expressjs.com
flatironjs.org
express-io.org
geddyjs.org
backbone.js

4 – NoSQL

a – Principe

En informatique, NoSQL (Not only SQL en anglais) désigne une catégorie de systèmes de gestion de base de données (SGBD) qui n'est plus fondée sur l'architecture classique des bases relationnelles.

L'unité logique n'y est plus la table, et les données ne sont en général pas manipulées avec SQL.

NoSQL = pas de SQL

http://fr.wikipedia.org/wiki/NoSQL

4 – NoSQL

b – Pourquoi

Supporter des montées en charge (scalabilité)

Pouvoir gérer une énorme quantité de données

Les données peuvent être structurées… ou pas

Supporte les ajouts fréquents

Asynchrone : envoi des informations

→ réponse client immédiate

4 – NoSQL

c – Types

NoSQL ”document” : une clé liée à un ”document” = structure de données. Les document peuvent eux-même contenir d'autres paires clé-valeur (ou clé-document).

NoSQL graph stocke des informations sur les réseaux. Cf Neo4J et HyperGraphDB.

Cassandra et HBase sont optimisés pour des requêtes sur de gros volumes de données, et stockent les données par colonnes, au lieu de les stocker par lignes.

4 – NoSQL

d – MongoDB – Introduction

MongoDB autorise la lecture d'enregistrements qui ne sont pas commités.

Un client peut lire des informations qui ne seront **jamais** écrites en base (ex : on redémarre le serveur mongodb avant qu'il n'ait écrit le ou les enregistrements).

Ce n'est que sur retour callback d'une écriture réussie, qu'on est sûr que les données sont bien écrites, et que même sur redémarrage elles seront présentes.

4 – NoSQL

d – MongoDB – Installation

http://docs.mongodb.org/manual/tutorial/install-mongodb-on-debian

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10

echo 'deb http://downloads-distro.mongodb.org/repo/debian-sysvinit dist 10gen' |
sudo tee /etc/apt/sources.list.d/mongodb.list

sudo apt-get update

sudo apt-get install -y mongodb-org
```

4 – NoSQL

d – MongoDB – Introduction

Création de la base de données : rien à faire !

Si la base n'existe pas : créée automatiquement.

Pas d'erreur.

4 – NoSQL

d – MongoDB – Introduction

http://docs.mongodb.org/manual/core/crud-introduction/

http://docs.mongodb.org/manual/reference/write-concern/

http://docs.mongodb.org/manual/tutorial/project-fields-from-query-results/

4 – NoSQL

d – MongoDB – Test installation

http://docs.mongodb.org/manual/tutorial/getting-started/

```
mongo
db
show dbs
use mydb
db
help
j = { name : "mongo" }
k = { x : 3 }
db.testData.insert( j )
db.testData.insert( k )
show collections
db.testData.find()
db.testData.remove({})
```

4 – NoSQL

d – MongoDB – Introduction

Collection = tableau de documents qui ont un ou plusieurs indexes en commun.

```
db.collection.find();
```

Par défaut : 20 éléments maximum

=> Spécifier le nombre d'éléments par défaut :

```
DBQuery.shellBatchSize = 10;
```

4 – NoSQL

d – MongoDB – Introduction

Tout charger en mémoire - mais ce n'est pas fait pour ça :

```
var myCursor = db.inventory.find( { type: 'food' } );
var documentArray = myCursor.toArray();
var myDocument = documentArray[3];
```

Itération nécessaire :

```
var myCursor = db.inventory.find( { type: 'food' } );
while (myCursor.hasNext()) {
   print(tojson(myCursor.next()));
}
```

4 – NoSQL

d – MongoDB – Relationnel

Recherche = "**WHERE**" en SQL.
Ce ne sont que des filtres simples : clé / valeur.

```
mongoDbObj.livres.find().toArray(
    function(err, data){
        if(err){
            // gestion de l'erreur
            console.log(err);
        } else {
            // gérer les données
    }
});
```

4 – NoSQL

d – MongoDB – Exemples – Filtres

Livres dont l'`id=2`

```
mongoDbObj.livres.find({ id: 2 })
```

Livres dont l'`id <= 21`

```
mongoDbObj.livres.find({ id: {$gte:21} })
```

Livres qui, parmi leurs "emprunteurs", ont une valeur `id=10`

```
mongoDbObj.livres.find({ "emprunteurs.id": 10 })
```

Livres qui ont un moins un de leurs emprunteurs avec un `id<=5`

```
mongoDbObj.livres.find({ "emprunteurs.id": {$lte: 5} })
```

4 – NoSQL

d – MongoDB – Exemples – Projections

Projection = filtrer sur les champs pour éviter que le document en entier soit renvoyé.
Exemple - documentation officielle

```
db.inventaire.find(
    { type: 'food' }, { item: 1, qty: 1, _id:0 }
)
```

Qu'une partie d'un sous élément dans le document

```
db.inventaire.find(
    { _id: 5 }, { ratings: { $slice: 2 } }
)
```

Récupérer juste une partie d'un sous ensemble
Seulement **2 méthodes disponibles** : `$elemMatch` et `$slice`

4 – NoSQL

d – MongoDB – Exemples – Insertion

Insertion

```
mongoDbObj.livres.insert(
    NouveauLivre,
    {w:1},
    function(err, result){
        if (err) {
            // échec
            console.log(err);
        } else {
            // réussite
        }
    }
});
```

4 – NoSQL

d – MongoDB – Exemples – Mise à jour

```
mongoDbObj.livres.update(
    { id:8 }, {titre:"Le monde du fleuve"}, {w:1},
    function(err, result) { /* réussite/erreur */ }
);
```

(!!) Ici TOUT l'enregistrement va être écrasé
=> perte de tous les emprunts, et de l'auteur
Solution : correctement préciser le champ :

```
mongoDbObj.livres.update(
    {id:8}, {$set: {titre:”Le monde du fleuve”}},
    {w:1},
    function(err, result) { /* réussite/erreur */ }
);
```

4 – NoSQL

d – MongoDB – Exemples – Suppression

Tous les enregistrements :

```
mongoDbObj.livres.remove(function(err, result) {
        /* réussite/erreur */
});
```

Avec une condition :

```
mongoDbObj.livres.remove(
    {id: 9}, {w:1},
    function(err, result) {
        /* réussite/erreur */
    }
);
```

4 – NoSQL

d – MongoDB – Exemples – Jointures

http://mongoosejs.com/docs/populate.html :

```
Livre
.findOne({ title: 'Guide du routard' })
.populate('_creator')
.exec(function (err, story) {
    if (err) return handleError(err);
    console.log('The creator is %s',
        story._creator.name);
    // prints "The creator is Aaron"
})
```

4 – NoSQL

d – MongoDB – Modélisation

http://docs.mongodb.org/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/

Modélisation dite ”flexible” car que des documents dont le contenu peut totalement changer.

Un document contient un/des autres documents

4 – NoSQL

d – MongoDB – Modélisation

```
Personne => _id, nom, prenom
Adresse => _id, ligne1, résidents
collection personne {
   _id: objid,
   name: "theo",
   adresses: [objid, objid, objid]
}

collection adresse {
   _id: objid,
   ligne1
   residents: [objid, objid]
}
```

4 – NoSQL

d – MongoDB – Modélisation

```
var personneAdresses =
    db.users.findOne(_id: objid).adresses;

var adresses = db.adresses.find({
    involved : {$in : personneAdresses}
});
```

4 – NoSQL

d – MongoDB – Modélisation

```
var date_debut = new Date(2012, 1, 14);
var date_fin   = new Date(2012, 1, 15);

var adresses = db.adresses.find({
    date: {$gte : date_debut},
    date: {$lt  : date_fin}
});
```

4 – NoSQL

d – MongoDB – Mongoose

Lancer le serveur sinon cela
ne fonctionnera jamais !

```
sudo mongod –noprealloc
--dbpath /pathcomplet/nomfichier
```

4 – NoSQL

e – MongoDB + NodeJS

Connexion

```
// Récupérer
var MongoClient = require('mongodb').MongoClient;
// Connect to the db
MongoClient.connect(
    "mongodb://localhost:27017/exampleDb",
    function(err, db) {}
);
```

http://mongodb.github.io/node-mongodb-native/api-articles/nodekoarticle1.html

4 – NoSQL

d – MongoDB + NodeJS

Insertion de tuples

```
MongoClient.connect("mongodb://localhost:27017/exampleDb",
    function(err, db) {
      if (err) { return console.dir(err); }
      var collection = db.collection('test');
      var doc1 = {'hello':'doc1'};
      var doc2 = {'hello':'doc2'};
      var lotsOfDocs = [{'hello':'doc3'}, {'hello':'doc4'}];
      collection.insert(doc1);
      collection.insert(doc2, {w:1}, function(err, result) {});
      collection.insert(
          lotsOfDocs, {w:1}, function(err, result) {}
      );
});
```

4 – NoSQL

d – MongoDB + NodeJS – Mongoose

http://mongoosejs.com/

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');

var Chien = mongoose.model('Chien', { name: String });

var toutou = new Chien({ name: 'Arthur' });
toutou.save(function (err) {
    if (err) // ...
    console.log('sauvegarde ok');
});
```

4 – NoSQL

d – MongoDB + NodeJS – Mongoose

http://mongoosejs.com/docs/guide.html

1 – Faire un schéma

2 – ”Préparer” le schéma = `mongoose.model()`

3 – On peut instancier le résultat

4 – NoSQL

d – MongoDB + NodeJS – Mongoose

http://mongoosejs.com/docs/guide.html

```
var personneSchema = new Schema({
    nom_complet: {
        prenom: String,
        nom: String
    }
});
var Personne = mongoose.model(
    'Personne', personneSchema
);
var moi = new Personne({
    nom_complet: { nom: 'Pons', prenom: 'Olivier' }
});
```

4 – NoSQL

d – MongoDB + NodeJS – Mongoose

http://mongoosejs.com/docs/guide.html
Dans tous les schémas :
- champ **_id** caché par défaut
- getter id qui renvoie **_id** par défaut

Pour ne pas le préciser :

```
var schema = new Schema(
    { name: String },
    { _id: false,
      id: false }
);
```

4 – NoSQL

d – MongoDB + NodeJS – Mongoose

(!) Spécificité des dates de Mongoose : les changements des dates ne sont ***pas*** prises en compte dans la logique de prise en compte de modification d'un document.
=> Si on ne fait *que* changer les dates, mongoose ne voit aucun changement et l'appel à `monDocument.save()` ne fera rien.
=> Préciser "manuellement" le champ date qu'on a changé `markModified` :

```
monDocument.markModified("monChampDate")
```

Là - et *seulement là*, `monDocument.save()` fonctionnera comme on s'y attend.

4 – NoSQL

d – MongoDB + NodeJS – Mongoose

TP : implémentation d'un diagramme de classe

4 – NoSQL

d – MongoDB + NodeJS – Mongoose

Votre TP : implémentation du
même diagramme de classe,
mais de trois façons différentes,
en expliquant dans
quel but chaque implémentation
peut être utilisée