Design Pattern

→ Modèles de conception

→ Bonnes pratiques à suivre

du Gang of 4 23 patterns généraux

→ Factory, singleton, façade, iterator, presenter

Patterns d'architectures :

→ Model Views Controller
↳ ASP.net MVC

→ Object Relational Mapping
↳ Entity Framework

→ AOP, IoC, ...

MVC est Commun en JEE

fortement conseillé en Php
(Symfony → complexe)

ASP.net MVC → relativement facile
→ "bare metal" (brut)

Pensé pour être simple, minimum d'efforts, efficace ⇒ bon produit

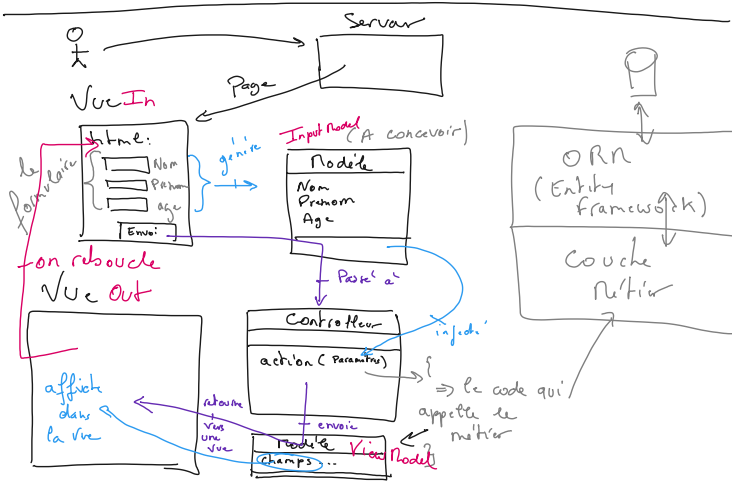
Concepts Génériques du MVC

(chaque implémentation peut être +/- différente)

Objectif du Pattern :

Découpler la présentation (ce que l'utilisateur final obtient) → La **Vue**, des opérations à effectuer (**actions**), dans des **Controlleurs** → 2 niveaux

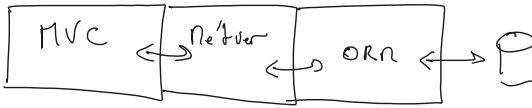
Les Données qui passent entre Vues et Controlleurs sont les **Modèles**



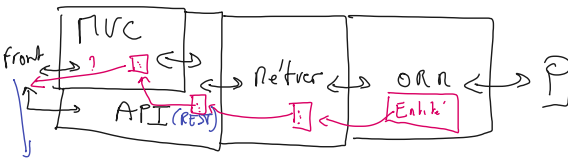
Notion de Model et ORM

Input View Model ⇒ MVC

Domain Model ⇒ ORM
(Entités)



Voir



Projet .net Type Appli Web

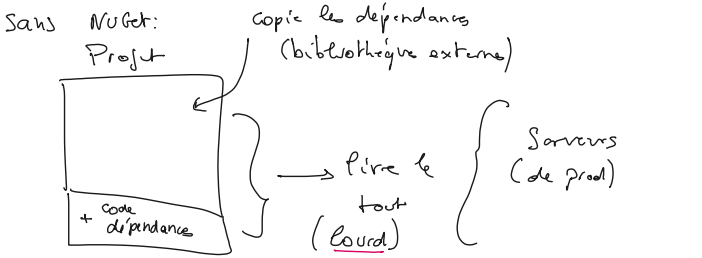
Mix possible :

- WebForm → OK en unitaire } mais pas recommandé
- MVC, API } Pratique normale

Utilisation des packages NuGet.

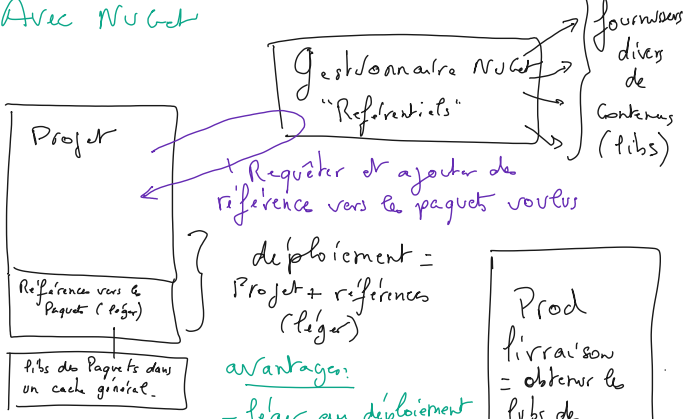
=> Très fortement recommandé

Sans NuGet:



⚠ Pas d'information concernant la mise à jour de versions de dépendances

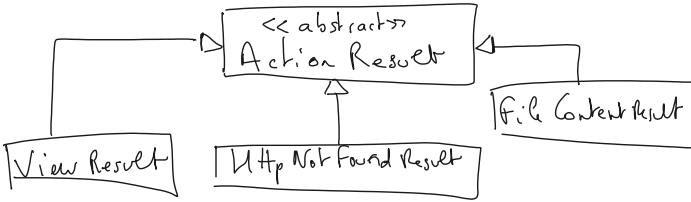
Avec NuGet

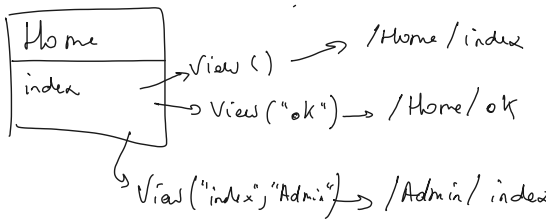


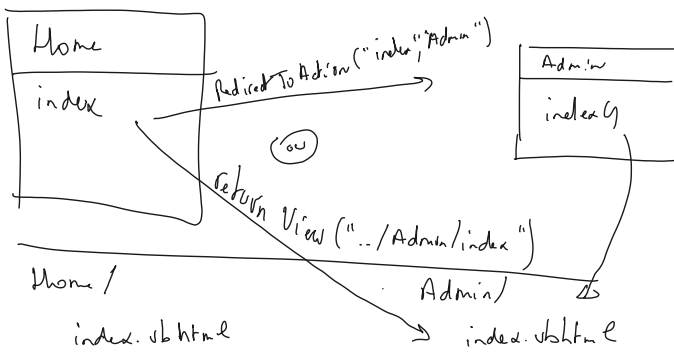
avantages:

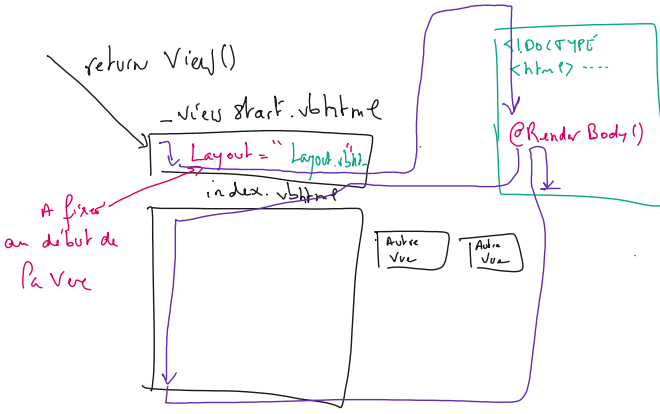
- léger au déploiement
- libs mutualisées entre les Projets
- Une simple requête permet de savoir si une nouvelle version est disponible

Resultats:


<<abstract>>
Action Result
View Result
HttpNotFoundResult
FileContentResult


Home
index
View()
/Home/index
View("ok")
/Home/ok
View("index","Admin")
/Admin/index


Home
index
RedirectToAction("index","Admin")
ou
Return View("../Admin/index")
Admin
index()
Home/
index.vbhtml
Admin/
index.vbhtml



Passer des paramètres de la vue vers le Layout:

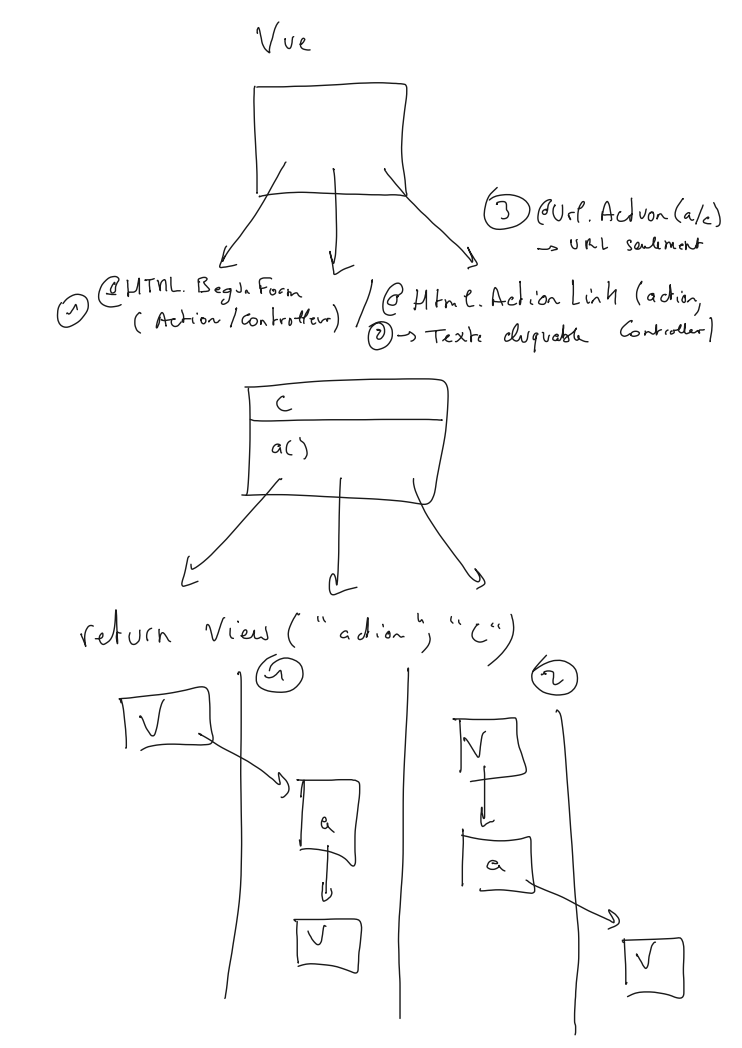
index.vbhtml

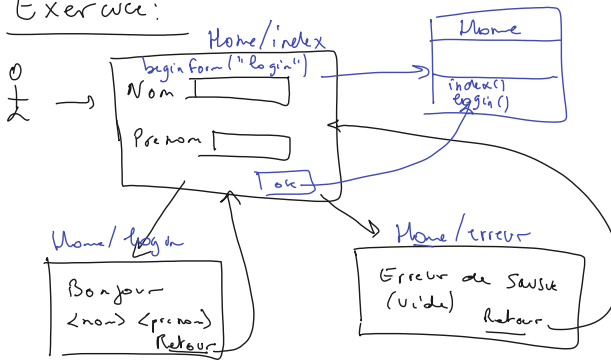
```
@ Code
    Layout = " ____ "
    ViewBag.clef = valeur
@ End Code
```

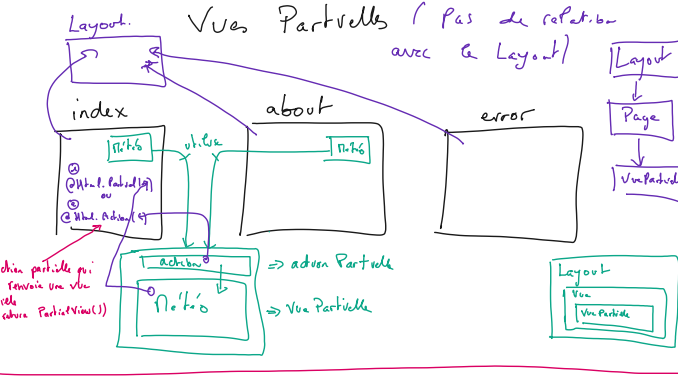
Layout.vbhtml

```
<DOCTYPE  --->
<title>
    @ViewBag.clef
</title>
```

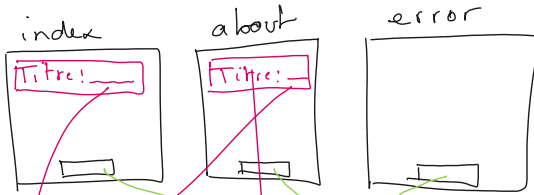
Sera remplacé par sa valeur


Vue
3 @Url.Action(a/c)
→ URL seulement
1 @Html.BeginForm
(Action/Controller) / @Html.ActionLink(action,
2 → Texte cliquable Controller)
C
a()
return Views("action";"C")
1
V
a
V
2
V
a
V


Exercice :
Home/index
beginForm("login")
Nom
Prenom
ok
Home
index()
login()
Home/login
Bonjour
<nom> <prenom>
Retour
Home/erreur
Erreur de saisie
(vide)
Retour


Layout.
Vues Partielles (Pas de relation avec le Layout)
index
about
error
Méth
vue
@Html.Partial("")
ou
@Html.Action("")
Layout
Page
VuePartielle
action partielle qui renvoie une vue Partielle (return PartialView())
action
Méth
=> action Partielle
=> Vue Partielle
Layout
Vue
VuePartielle

Exercice Vues Partielles :



l'heure d'exécution de la Page

→ @Html.Action()

Action : extraire l'heure formatée → ViewBag

le Contenu de ViewBag.Title

→ @Html.Partial

⇒ parce que pas d'action nécessaire

Vue :

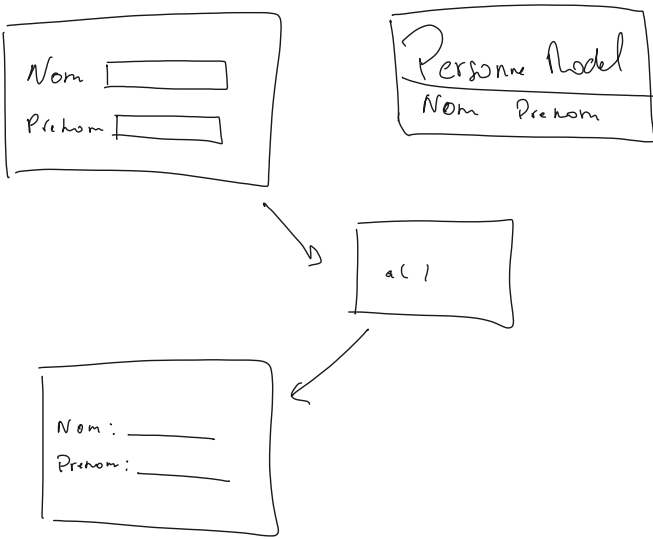
```
@ViewBag.___
```

action :

```
ViewBag.Moment =
    DateTime.Now
return PartialView
```

Vue :

```
@ViewBag.Moment
```



Bouclage de collections : https://www.w3schools.com/asp/razor_vb_loops.asp

Boucle MVC Complète :

input View :

```
<body>
    @Using Html.BeginForm("Afficher", "Personne")
        @<text>
    Nom : @Html.TextBox("Nom")<br />
    Prénom : @Html.TextBox("Prenom")<br />
    <input type="submit" value="Envoyer" />

        </text>
    End Using
</body>
```

```
Public Class PersonneInputModel
    Public Property Nom As String
    Public Property Prenom As String
End Class
```

```
Namespace Controllers
    Public Class PersonneController
        Inherits Controller

        ' GET: Personne
        Function Index() As ActionResult
            Return View()
        End Function        ' GET: Personne
        Function Afficher(inputModel As PersonneInputModel) As ActionResult
            Dim model As New PersonneViewModel With {
                .NomComplet = inputModel.Prenom & " " & inputModel.Nom
            }

            Return View(model)
        End Function
    End Class
End Namespace
```

Affichage :

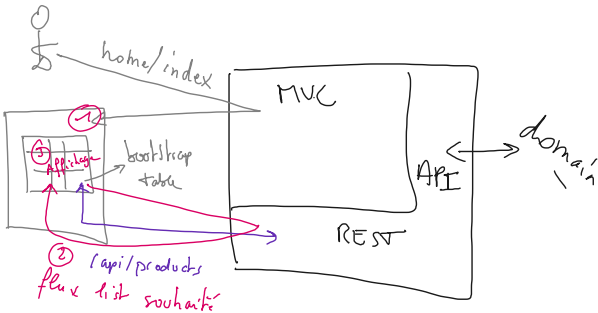
```
</head>
<body>
    <div> Bonjour : @Model.NomComplet
    </div>
</body>
</html>
```

Génération de Tables / Data Grid

→ Pas de composant

→ Doit être faite à la main via les boucles !

Autre Solution : API REST / JSON et Tableaux Ajax jQuery (Best Practice de Microsoft)



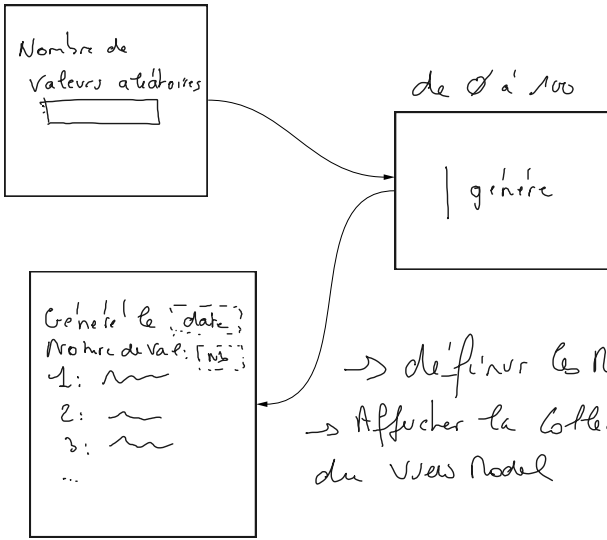
Projet : App_Data
Model
View
Controller
Scripts
- .js → jquery + dépendances
css, Content, ...

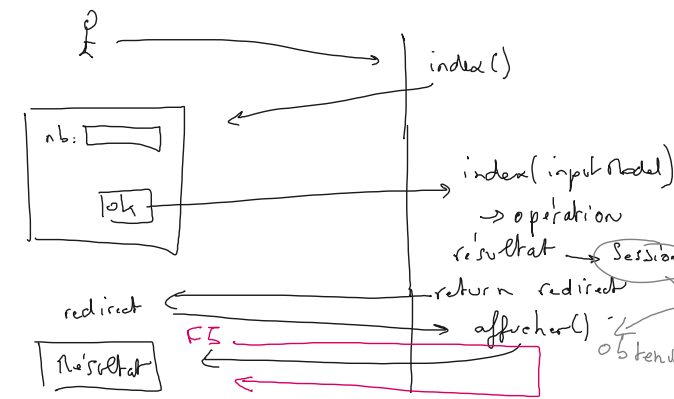
```
<script src="~/.../.js"

Cpt.dataURL = "~/api/products"
Cpt.display();
```

/api/products
/api/2/clients/ALFKI

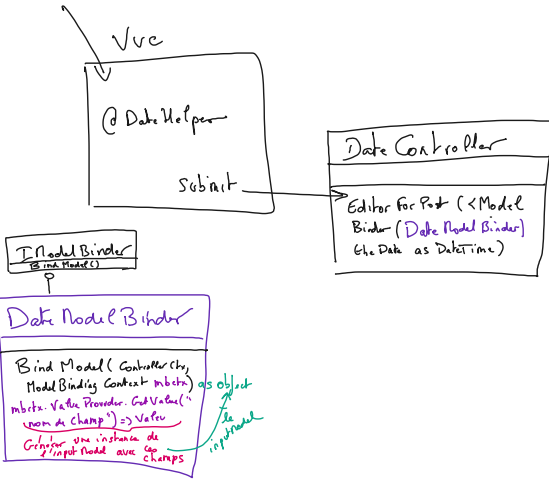
/api/[version]/<ressource>[/id]


Nombre de
valeurs aléatoires
de 0 à 100
génère
Crée/ré le "date"
Nombre de val: "nb"
1:
2:
3:
...
→ définir le Modèle
→ Afficher la Collection
du View Model

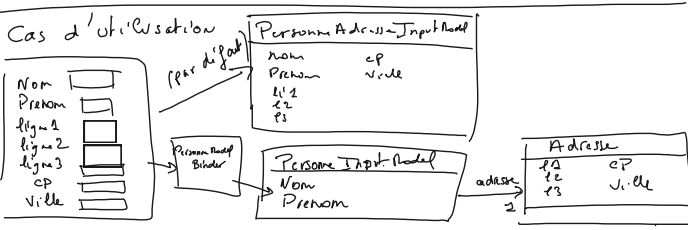

index()
nb:
ok
index(inputModel)
→ opération
résultat
Session
return redirect
redirect
afficher()
FE
Résultat
obtenir

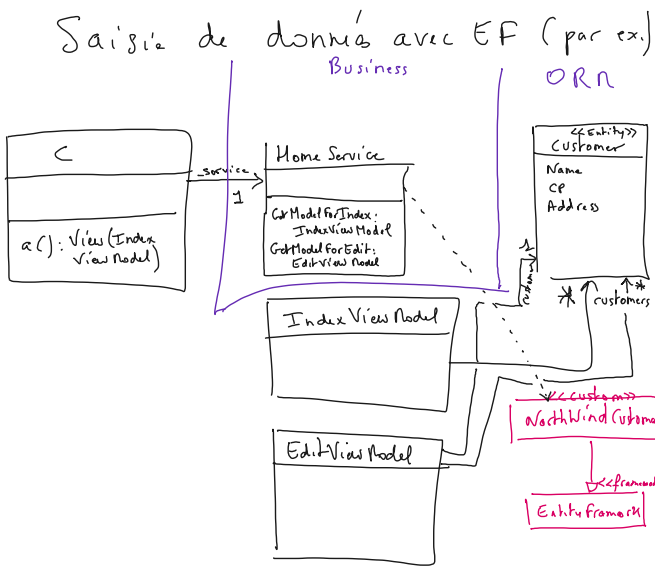
Custom Binders

DateHelper.vbHtml → DateHelper.inputDate(....)



Cas d'utilisation




Saisie de donnés avec EF (par ex.)
Business
ORM
C
-service
1
a() : View(IndexViewModel)
HomeService
CarModelForIndex: IndexViewModel
GetModelForEdit: EditViewModel
<<Entity>>
Customer
Name
CP
Address
context
customers
IndexViewModel
EditViewModel
<<context>>
NorthWindCustomers
<<framework>>
EntityFramework

Expression Lambda (renvoie une valeur)

C#

(c => c.Name;)

(c => {return c.Name;})

VB.net

Func(c) c.Name

Methode Lambda

(c => { return c.Name; })

↳ si une methode Lambda ne fait que renvoyer une valeur, alors → Expression Lambda

```
<div>
    @Using Html.BeginForm()
      @<text>
      @Html.EditorForModel()
    @Html.ValidationSummary()
      <input type="submit" value="Envoyer" />
      </text>
    End Using
  </div>

Imports
System.ComponentModel.DataAnnotations

Public Class PersonneModel
  <Required()>
  Public Property Nom As String


  Public Property Prenom As String

End Class
```

3 Jours Entity Framework (connexion Wifi Limité / pas de tablette)

```
Namespace Controllers
    Public Class HomeController
        Inherits Controller

        ' GET: Home
        <AcceptVerbs(HttpVerbs.Get)>
        Function Index() As ActionResult
            Dim model As Employee

            Using ctx As New Northwind
                model = ctx.Employees.FirstOrDefault()
            End Using

            Return View(model)
        End Function

        <AcceptVerbs(HttpVerbs.Post)>
        <ActionName("Index")>
        Function IndexPost(inputModel As Employee) As ActionResult
            Using ctx As New Northwind
                ctx.Entry(inputModel).State = Entity.EntityState.Modified
                ctx.SaveChanges()
            End Using

            Return RedirectToAction("Index")
        End Function

        '<AcceptVerbs(HttpVerbs.Post)>
        '<ActionName("Index")>
        'Function IndexPost() As ActionResult

        '    Return RedirectToAction("Index")
        'End Function
    End Class
End Namespace
```