

Sommaire

Tour d'horizon de ce que l'on va aborder...



Achitecture n-tiers

Client Tier
Browser, Applets, Applications, Clients

Presentation Tier JSP, Servlets, UI Elements

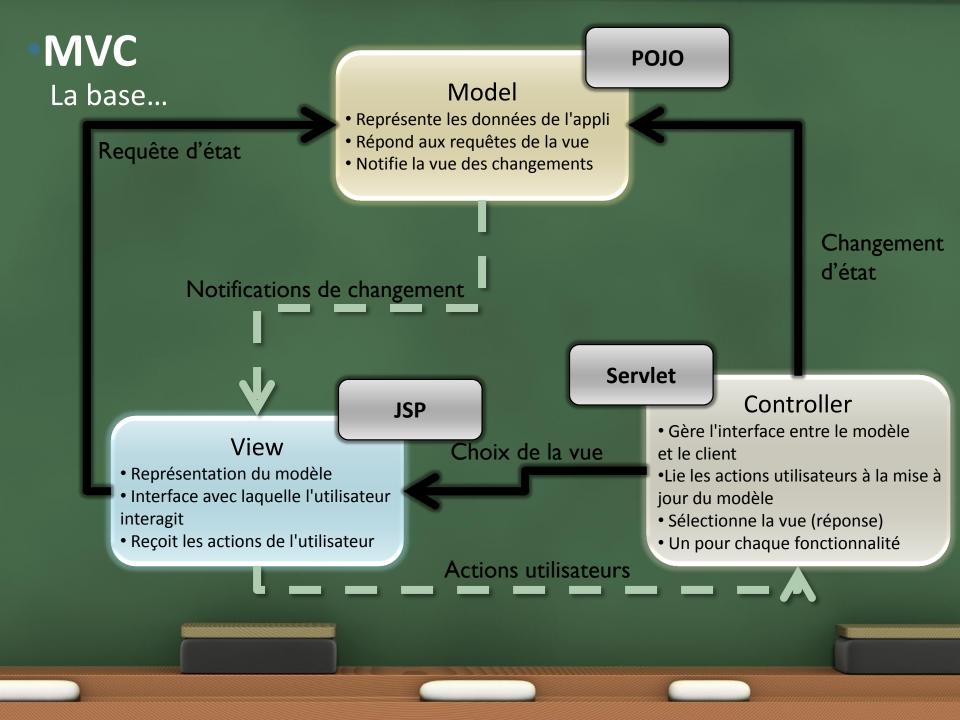
Business Tier
EJB and Business Objects

Integration Tier JDBC, JMS, Connectors

Resource Tier
Databases, Systems Legacy

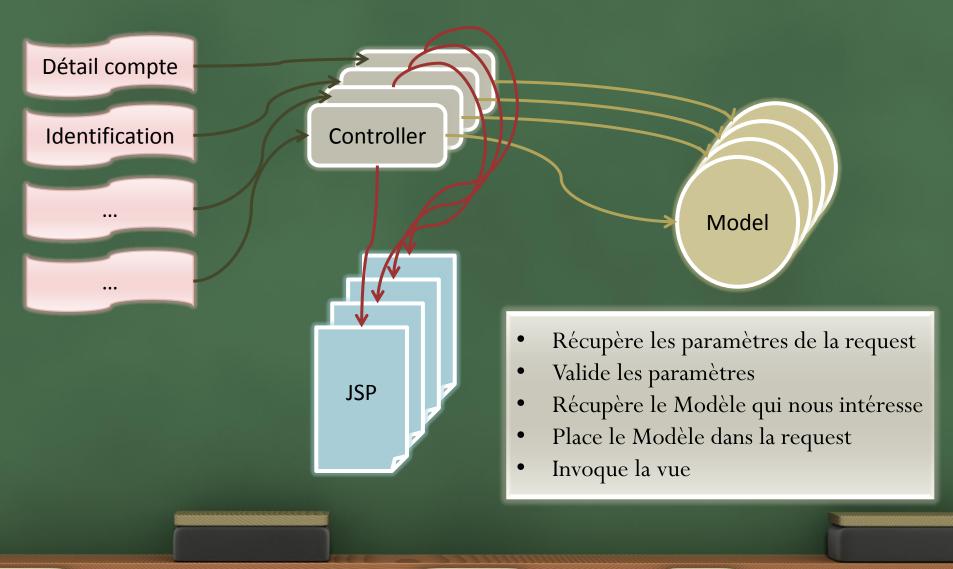
Core J2EE Patterns, 2nd Edition Apply zero or more -LEGEND: Presentation Tier Intercepting Filter **Business Tier** Delegate Control -Centralize Control New Integration Tier Application Controller Front Controller Create Create Composite View Dispatch to Dispatch to Delegate processing New target View View to Helpers Context Object View Helper Compose View from Sub-Views Dispatch Dispatch ---- Front Controller Front Controller to View to View Lightweight Control Processing Control Processing Access Dispatcher View Service To Worker Business Service Access Access Business Service Business -Service Invoke Business Processing-New Locate Application Service **Business Delegate** Service Locator Services Connect and Invoke Delegate Business Service Processing Layers Coordinate Invoke Session Facade Transfer Object Business Business Processing Processing Facade Send Assembles for Data Encapsulate and coordinate Model List New Business Object Invoke Value List Handler Business Implement Send Processing **Entity Beans** Data uses Transparently Persist Composite Entity Transfer Object Assembler Service Activator Retrieve **Business List** uses Chain New Asynchronous **Domain Store** Transfer Object **Business Object** Processing uses Delegate Uses for Asynchronous Bean-managed Send Data Processing Persistence New Web Service Broker Data Access Object

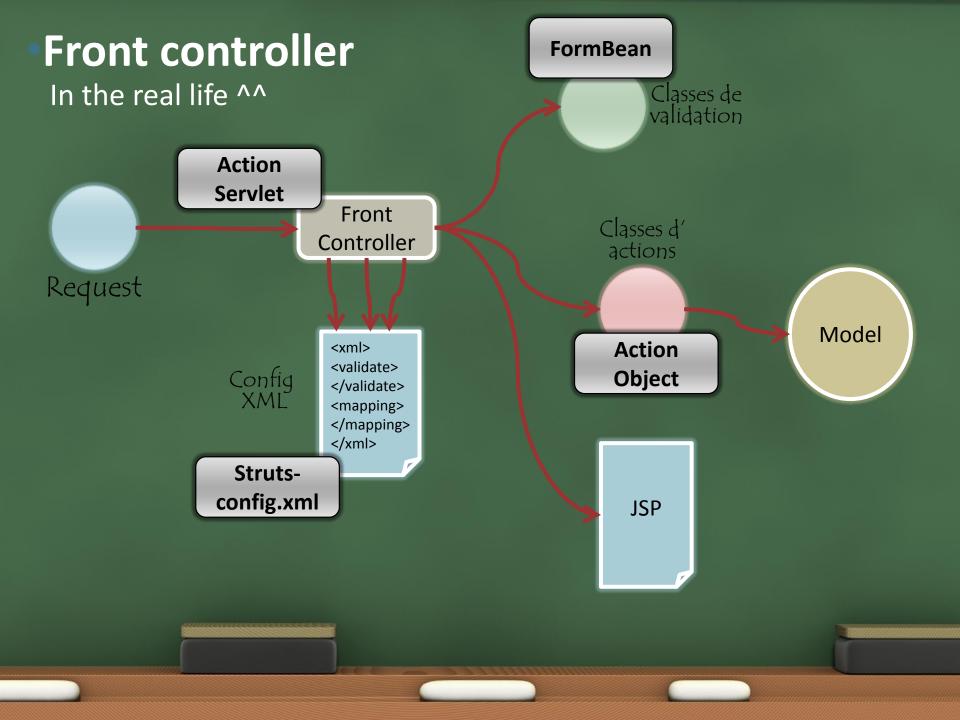
(c) 2003 corej2eepatterns.com. All Rights Reserved.



MVC, dans une appli web « Réelle »

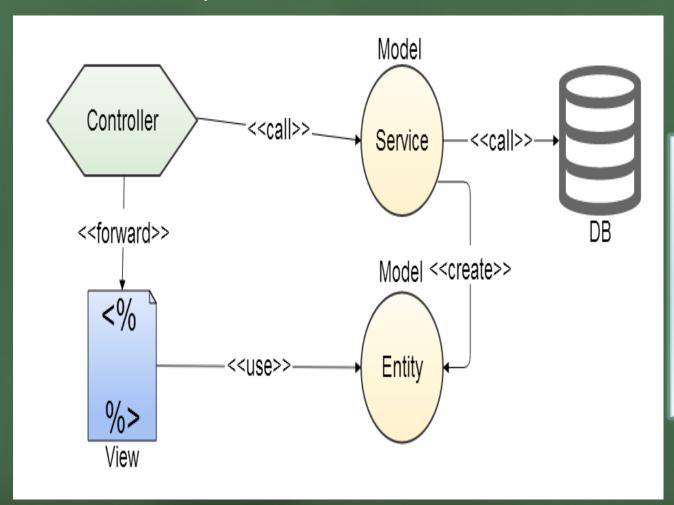
Ou comment gérer le code dupliqué...





•MVC

Quand on est que sur une VM



Oui mais voilà, et quand le modèle n'est pas sur le même serveur ???

Les outils du « remote »

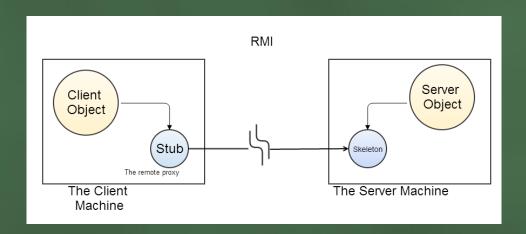
Ou les outils pour comprendre les pattern « remote »

JNDI

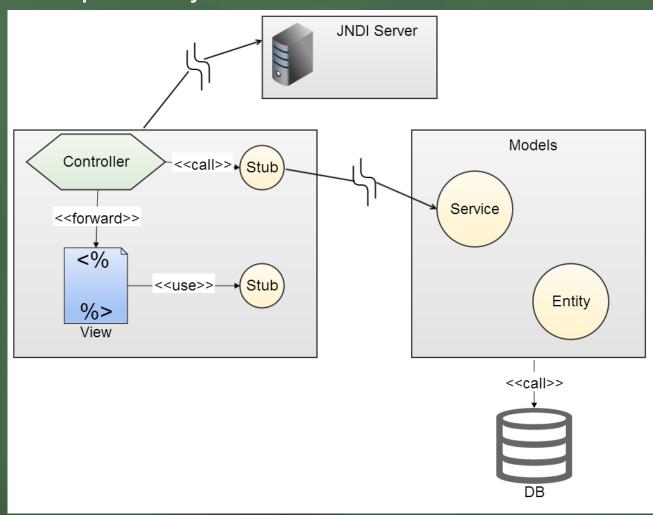
- Java Naming and Directory Interface
- Le but c'est de rajouter un élément à l'architecture pour centraliser la location des objets (un annuaire)
- Si on veut partager un objet on l'enregistre sur l'annuaire
- Si on en cherche un, on regarde dans l'annuaire

RMI

- Remote Method Invocation
- Le but est de récupérer un objet au travers du réseau



• MVC avec Model « remote » Un premier jet

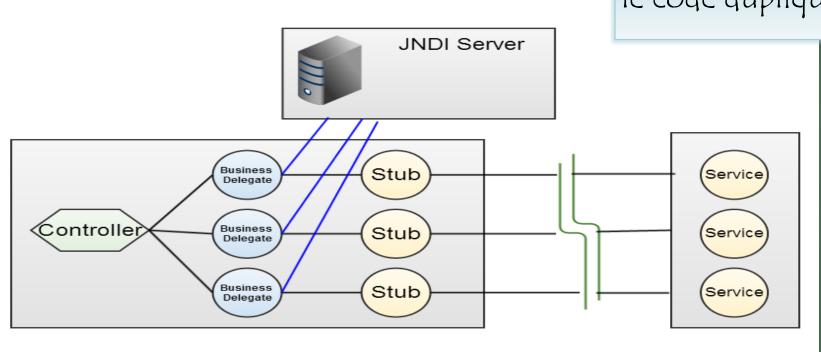


Oui mais voilà, des fois on est « remote » Et des fois pas ©

Le pattern Business Delegate

Ou comment cacher la complexité du JNDI et RMI

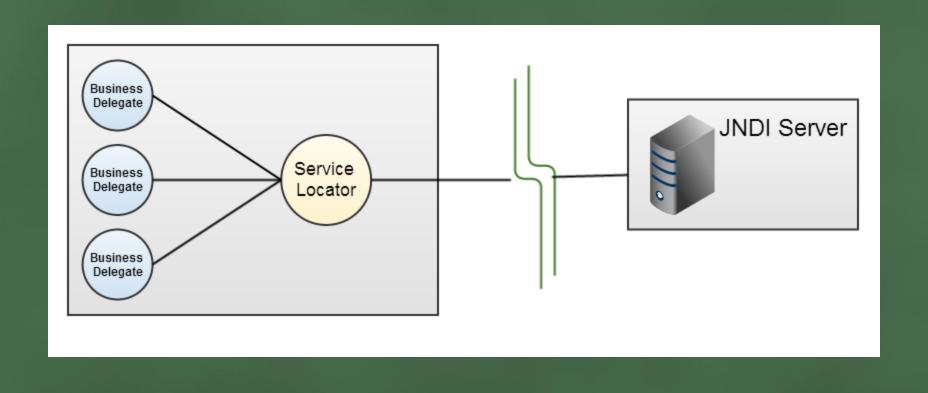
Et cette fois il est où le code dupliqué?



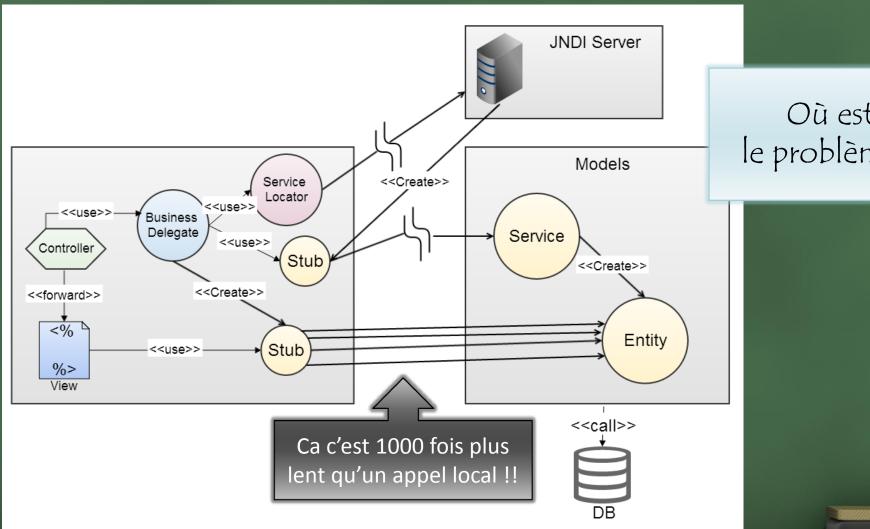
- Réduit le couplage entre les couches présentation et Business
- Encapsule la recherche dans l'annuaire
- Proxy pour le client

• Le Pattern Service Locator

Ou comment éviter de dupliquer le code du « lookup »



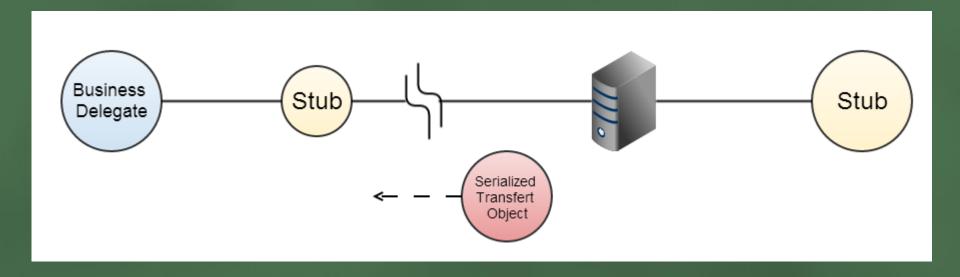
MVC avec Model « remote » Deuxième jet



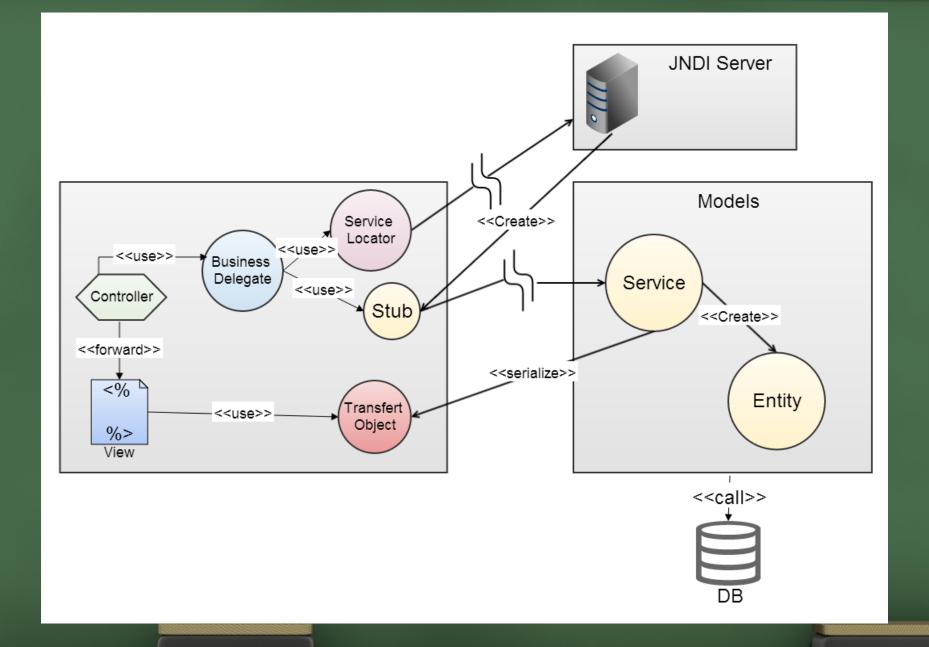
Où est le problème?

Le pattern DTO (Data Transfert Object)

Ou à quoi sert la sérialisation...

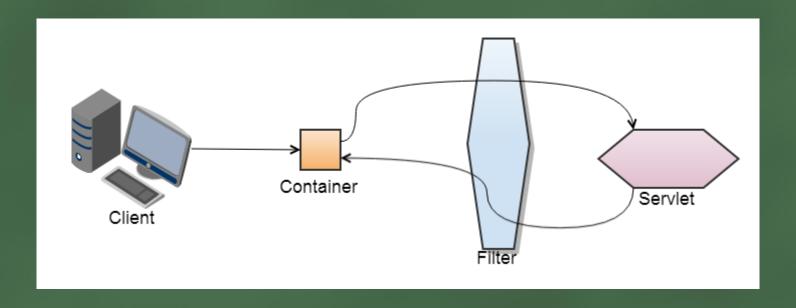


```
try {
    TransfertObject o = serviceStub.serviceMethod(param);
} catch (RemoteException e) {
    throw new ApplicativeException();
}
```



Le Pattern Intercepting Filter

Ou comment rajouter de l'information partout sans tout changer...



- Peut intercepter et/ou modifier la requête avant qu'elle arrive à la servlet
- Peut intercepter et/ou modifier la réponse avant qu'elle soit retournée au client
- Déclarée dans le web.xml
- Les filtres peuvent être chaîné