Applications orientées données (NSY135)

4 – Modèle-Vue-Contrôleur (MVC)

Auteurs: Raphaël Fournier-S'niehotta et Philippe Rigaux
(philippe.rigaux@cnam.fr,fournier@cnam.fr)

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

Plan du cours

MVC ?

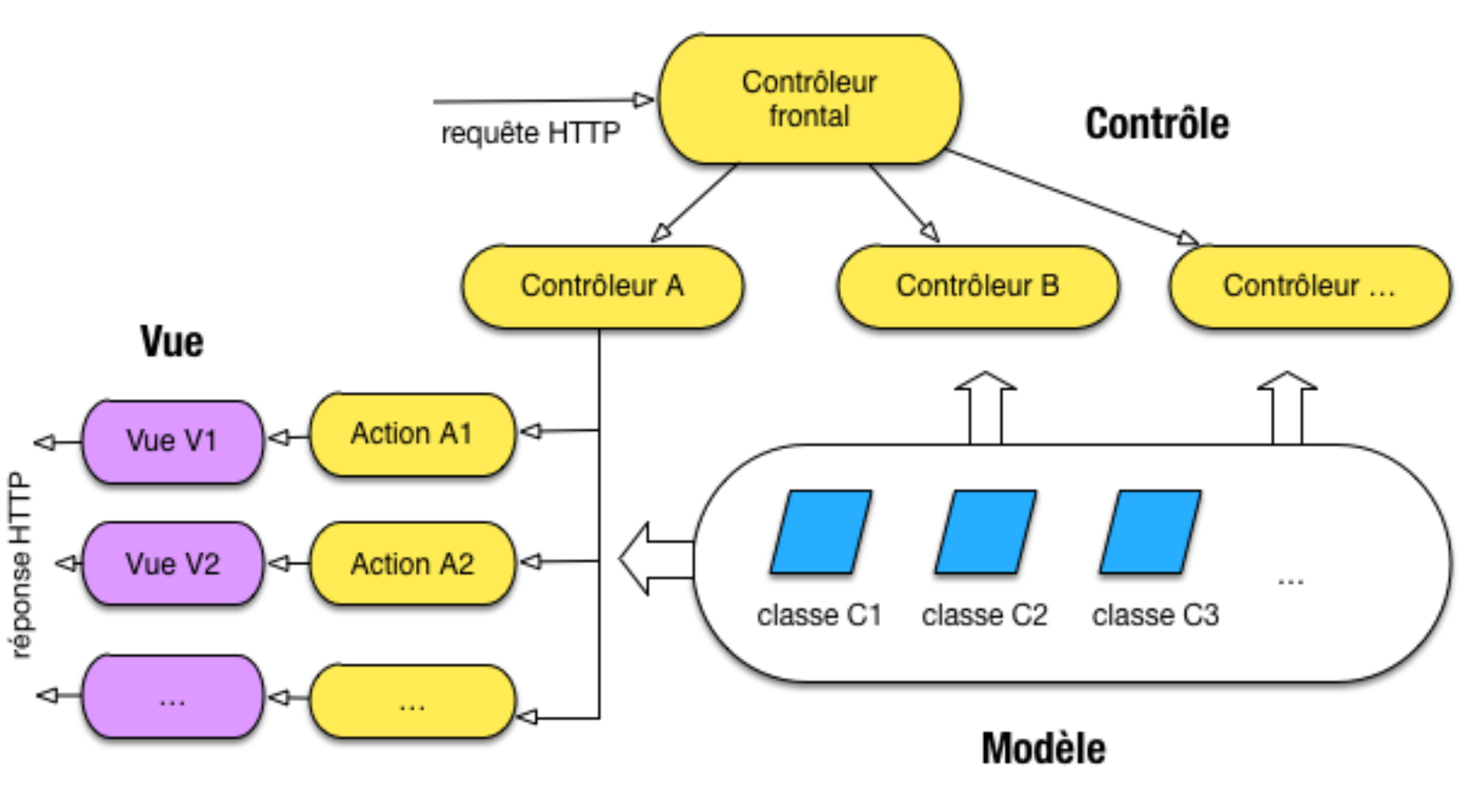
- MVC est un motif de conception ("design pattern")
- il mène à une organisation rigoureuse et logique du code
  - séparation/indépendance des couches
  - définition de règles permettant de savoir où ajouter une fonctionnalité
  - simplification de la maintenance et de l'évolutivité de chacune
  - clarté indispensable pour les gros projets : **gain de temps**
- il est très répandu et résulte de la formalisation de bonnes pratiques

Principe général de MVC

Le MVC sépare :

- données (Modèle)
- présentation (Vue)
- traitement (actions, coordonnées par des Contrôleurs)

Architecture MVC



(cas framework web)

Modèle

Le modèle

- implante les fonctionnalités de l'application, indépendamment des aspects interactifs.
- préservation de l'état d'une application entre deux requêtes HTTP, ainsi que des fonctionnalités qui s'appliquent à cet état.
- Toute donnée persistante doit être gérée par la couche modèle,
- des objets métiers non persistant implantant une fonctionnalité particulière (un calcul, un service) sont également dans cette couche.

Modèle 2/2

- Le modèle gère les données de session (le panier dans un site de commerce électronique par exemple) ou les informations contenues dans la base de données (le catalogue des produits en vente, pour rester dans le même exemple).
- Cela comprend également les règles, contraintes et traitements qui s'appliquent à ces données, souvent désignées collectivement par l'expression **logique métier de l'application**

♠ Les composants de la couche modèle doivent pouvoir être utilisés dans des contextes applicatifs différents: Ils doivent donc impérativement être **totalement indépendants** de la gestion des interactions avec les utilisateurs, ou d'un environnement d'exécution particulier.

Persistance

La persistance

- est la propriété de l'état d'un objet à être préservé au-delà de la durée d'exécution de l'application en général, et d'une action en particulier
- est obtenue le plus souvent par stockage dans une base de données.
- est un aspect d'un objet métier :
  - elle ne doit pas (ou très peu) impacter la logique applicative implantée par cet objet
  - on devrait pouvoir ajouter/modifier/supprimer le comportement de persistance d'un objet indépendamment des services qu'il fournit

Rôle de la partie Vue

La vue

- est responsable de l'interface (HTML ici)
- assure la mise en forme des données
- ne doit pas intégrer les traitements complexes ("métiers")
- n'accède pas au modèle, obtient ses données de l'action
- est souvent implantée par un moteur de templates

Intégration de code dans la Vue ?

- où placer les décisions/traitements relatifs à la vue ?
  - si on intègre beaucoup de code de traitement, on empiète sur la logique métier
  - ne pas trop surcharger la Vue
- **à éviter**
- Dans le cas de Java :
  - Java Server Pages (JSP) initialement
  - Java Standard Tag Library (JSTL)

Contrôleurs

Le rôle des contrôleurs est

- de coordonner les séquences d'actions/réactions d'une application Web
- de récupérer les données utilisateurs,
- de les filtrer et de les contrôler,
- de déclencher le traitement approprié (via le modèle),
- de déléguer la production du document de sortie à la vue adaptée

Ils permettent de

- structurer hiérarchiquement l'application
- faciliter la compréhension du code et la maintenance

♠ Comme pour la Vue, on évitera de placer de logique applicative dans un contrôleur, car le code **n'est pas réutilisable dans un autre contexte**