Web (Persistance)

Andrea G. B. Tettamanzi
Université de Nice Sophia Antipolis
Département Informatique
andrea.tettamanzi@unice.fr

CM - Séance 7

Organisation logicielle d'une application Web

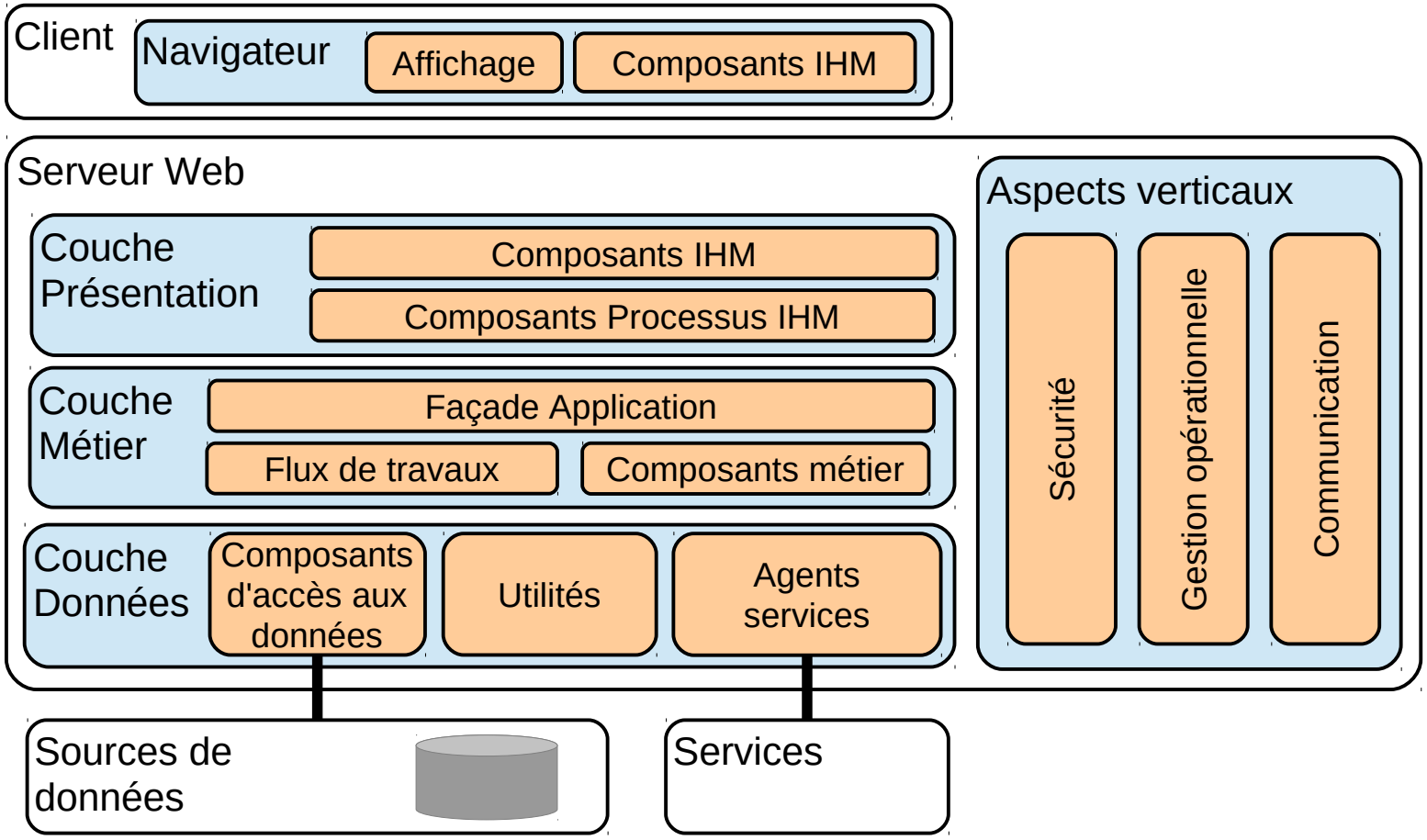
Plan

- Introduction
- Considérations verticales de conception
- Questions spécifiques de conception
- Considérations de conception pour les couches
- Considérations à propos du test et de la testabilité
- Considérations à propos du déploiement
- Patrons de conceptions pertinents

Introduction

- Site Web dynamique → Application Web
  - Application accessible à travers d'un navigateur Web
  - Le navigateur crée des requêtes HTTP pour des URL spécifiques associés à des ressources sur un serveur
  - Le serveurs engendre et renvoie des pages HTML que le navigateur peut afficher
  - Le noyau d'une application Web est le logiciel côté serveur
  - L'application peut être organisée à couches
  - Un exemple typique est l'architecture à 3 couches : présentation, logique métier, données

Architecture d'une application Web



Considérations générales de conception

- Objectifs de conception
  - Réduire la complexité en regroupant les tâches du même type
  - Veiller à la sécurité et aux performances
- Quelques lignes directrices
  - Partitionner l'application logiquement (ex : à couches)
  - Utiliser l'abstraction pour réaliser un couplage lâche
  - Prévoir les besoins de communication entre composants
  - Utiliser la mise en antémémoire pour réduire la latence
  - Prévoir le *logging* et l'instrumentation du code
  - Authentifier les utilisateurs au passage des frontières de trust
  - Ne jamais transmettre des données sensibles en clair
  - Faire tourner l'application avec le moindre des privilèges

Questions spécifiques de conception

- Point critiques communs dans la conception d'applications Web
- Par aire de conception, ils sont :
  - Traitement des requêtes (form post back / RESTful calls)
  - Authentification et autorisations
  - Mise en antémémoire (cache)
  - Gestion des exceptions
  - Logging et instrumentation du code
  - Navigation
  - Structure et affichage des pages
  - Gestion des sessions
  - Validation

Traitement des requêtes

- Deux approches au traitement des requêtes :
  - Approche « Post Back » (que nous connaissons déjà)
    - Développement basé sur la construction de formulaires
    - Centrée sur le serveur
    - Prototypage rapide
  - Approche « RESTful », basée sur l'idée de service Web
    - Plus centré sur le client
    - Contrôle plus fin de l'IHM et plus de flexibilité
- Le choix entre ce deux approches doit prendre en compte le degré de contrôle sur l'IHM souhaité, l'approche de développement et l'éventuel passage à l'échelle

Representational State Transfer (REST)

- Style d'architecture pour les systèmes hypermédia distribués
- Proposé en 2000 par Roy Fielding
- Défini par six contraintes :
  1) Client-serveur : séparation nette de responsabilités
  2) Sans état : chaque requête doit contenir aussi son contexte
  3) Propension à la mise en cache doit accompagner les données
  4) Interface uniforme : (a) ressources identifiée unitairement ; (b) ressources ont une représentation définie ; (c) messages auto-descriptifs ; (d) accès aux états suivants est décrit dans le message courant (principe de l'hypermédia)
  5) Hiérarchie par couches
  6) Le client peut exécuter des scripts reçus du serveur

Authentification et Autorisation

- Authentification = être sûr que l'utilisateur est celui qu'il dit être
  - Identifier des frontières de confiance (aires sécurisées)
  - Utiliser des bonnes pratiques de gestion des comptes (mots de passe forts, péremption, blocage, etc.)
  - Utiliser des mécanismes supportés par la plate-forme (sous Apache : .htaccess, .htpasswd et directives associées)
- Autorisation = définition de qui a le droit de faire quoi
  - Choix de la granularité critique :
    - Trop fin = coûteuse à gérer
    - Trop grosse = réduit la flexibilité
- Aspects critiques pour la sécurité et la fiabilité d'une application

Mise en cache

- La mise en cache améliore les performances et réduit la latence
- Pourtant, c'est une lame à double tranchant
- Utiliser pour
  - Optimiser l'accès à des données de référence
  - Éviter des allées/retour sur le réseau et duplication de calculs
- Essayer de mettre en cache de façon asynchrone ou batch
- Réserver pour des données / pages relativement statiques
- Mettre en cache les données en format prêt pour l'utilisation

Gestion des exceptions

- Important pour la sécurité et fiabilité d'une application :
  - Évite de révéler des détails sensibles à l'utilisateur
  - Améliore la robustesse de l'application
  - Évite de tomber dans un état incohérent en cas d'erreur
- Fournir des messages d'erreurs conviviaux sans pourtant exposer des informations sensibles
- S'assurer que toute exception soit interceptée et traitée
- Il vaut mieux prévoir un gestionnaire global d'exception qui produit une page pour tous les types d'exceptions
- Ne pas utiliser les exceptions pour contrôler le flux logique de l'application

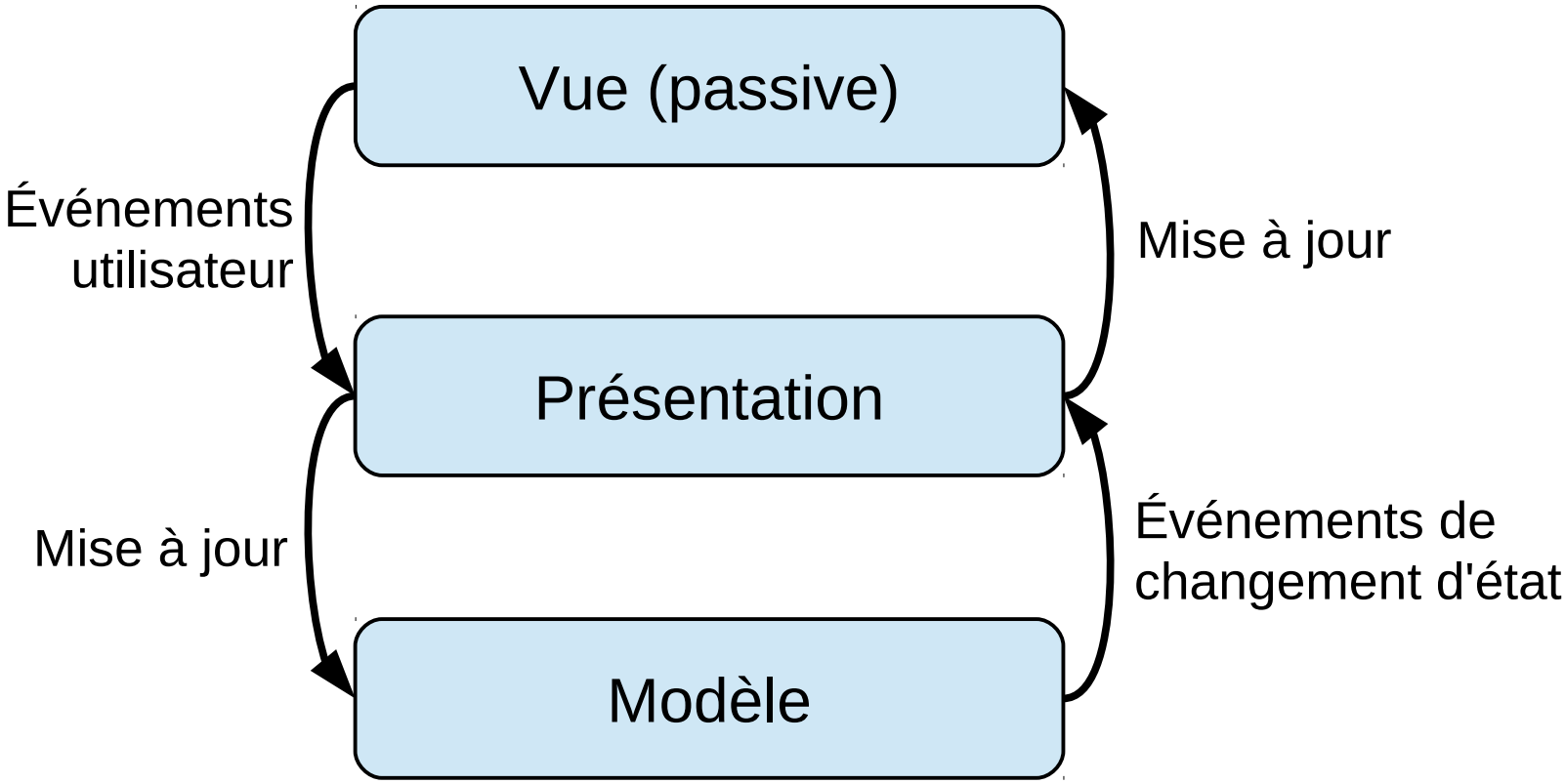
Logging et instrumentation

- Logging = enregistrement d'un historique des événements
- Instrumentation = mise en œuvre d'instruments de mesure en vue de créer un système d'acquisition de données
- Les deux activités sont complémentaires et ont le même but :
  - Permettre de reconstruire le fonctionnement du système et comprendre les causes d'une anomalie
  - Faire des audits et découvrir et documenter des intrusions ou des utilisations illicites du système
  - Lutter contre le problème de la répudiation
- Enregistrer tous les événements critiques à tous les niveaux
- Le système de logging doit être sécurisé
- Les informations sensibles ne doivent pas être enregistrées !

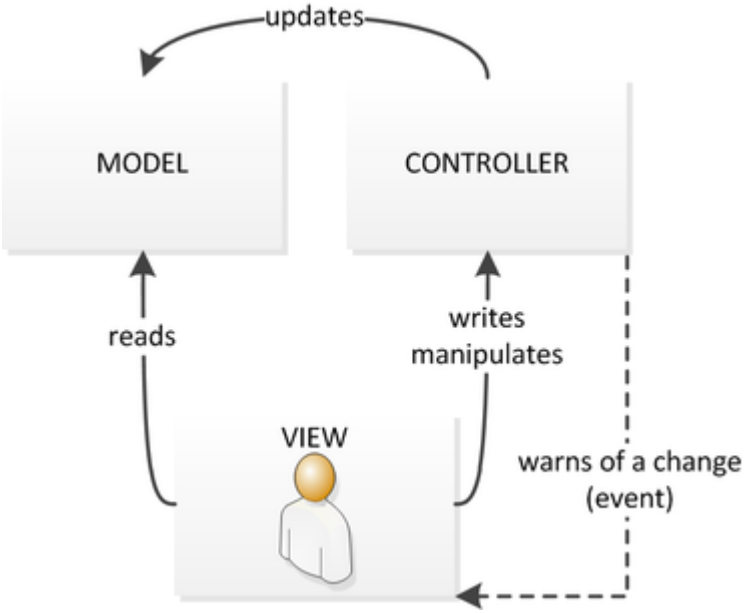
Navigation

- But : permettre à l'utilisateur de se déplacer facilement parmi les pages et le écrans de l'application
- La navigation doit être séparée de la logique du traitement
- La conception d'une structure de navigation contribue à minimiser la confusion de l'utilisateur et réduire la complexité de l'application
- Utiliser des patrons de conception tels que Modèle-Vue-Présentation ou Modèle-Vue-Contrôleur
- Incapsuler la navigation dans une page principale
- Fournir à l'utilisateur et aux moteurs de recherche le plan du site

Patron Modèle-vue-présentation



Patron Modèle-vue-contrôleur



Structuration des pages

- Idée de la séparation des aspects
- Utiliser CSS plutôt que des tables, sauf lors que les donnée à présenter sont de forme tabulaire
- Utiliser une structure commune et cohérente pour toutes les pages
- Éviter de créer des grandes pages qui effectuent des tâches multiples et privilégier des petites pages, rapides à télécharger et afficher
- Placer les scripts côté client dans des fichiers séparés pour permettre, entre autre, au navigateur de les tenir en cache
- Éviter de mélanger des scripts avec du code HTML

Affichage des pages

- Privilégier l'utilisation de scripts côté client pour diminuer les temps de réponse et réduire les allée/retour entre le client et le serveur
- Penser à concevoir l'application avec un support pour la localisation
- Ne pas mélanger le code qui s'occupe de l'affichage avec celui qui s'occupe du traitement métier (utiliser les patrons de conception)

Gestion des sessions

- La gestion des session est strictement liée à la persistance
- Vérifier d'abord s'il est vraiment nécessaire de stocker des information sur l'état de la session
- Le choix d'où et comment stocker l'état d'une session dépend de plusieurs facteurs
  - Un seul serveur Web ou plusieurs serveurs ?
  - Informations sur l'état lourdes ou légères ?
  - Données sensibles ?
- Selon les cas, on peut stocker l'état côté client, utiliser des cookies, stocker côté serveur à des couches différentes, où sur un gestionnaire de bases de données externe.

Validation

- Par « validation » on entend ici la vérification de tout ce qui est rentré par l'utilisateur
- Une validation insuffisante peut permettre des attaques du type cross-site scripting, SQL injection, buffer overflow, etc.
- Partir toujours du principe que les données saisies par l'utilisateur sont malicieuses
- Concevoir des stratégies pour contraindre, stériliser ou rejeter les données envoyées par le client
- Utiliser la validation côté client pour des raisons de performances, mais valider toujours aussi sur le serveur pour des raison de sécurité
- C'est une bonne idée de réutiliser des solutions prêtes à l'emploi

Considérations à propos des couches

- Couche présentation
  - Séparer la logique d'interaction et le composants d'IHM
- Couche métier
  - Concevoir des objets métier qui représentent les entité réelles et les utiliser pour passer des données d'un composant à l'autre
- Couche données
  - Incapsuler les détail de l'accès à une base de données
- Couche services
  - Incapsuler l'accès à des services de tiers ou d'un serveur distant

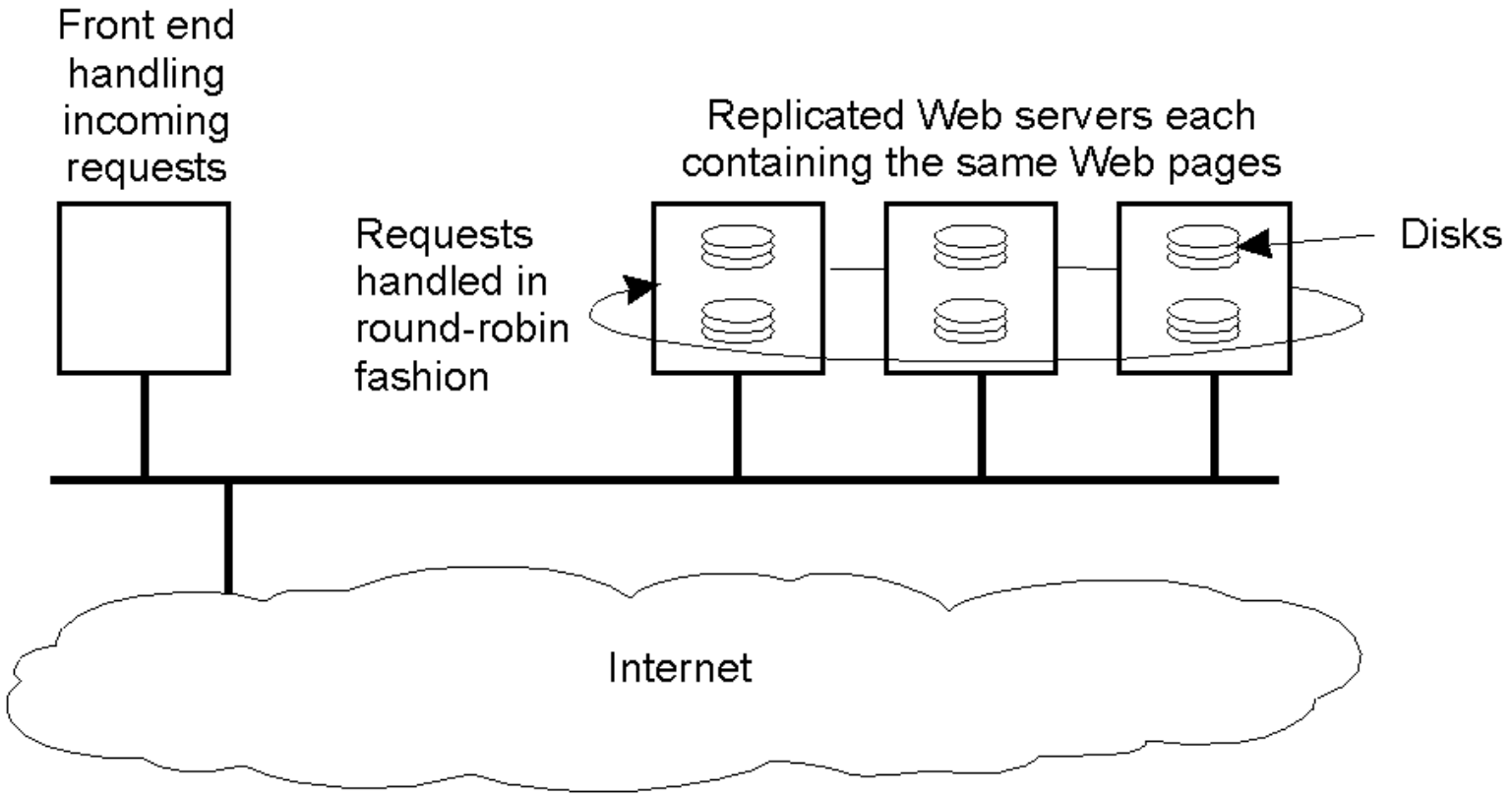
Test et testabilité

- La testabilité est une mesure de combien un système permet de définir des critères et exécuter des tests pour vérifier si les critères sont satisfaits
- Pour obtenir une application testable :
  - Définir clairement entrées et sorties pour chaque composant
  - Utiliser des patrons de conception pour la couche présentation
  - Concevoir une couche séparée pour la logique métier et le flux de travaux
  - Concevoir des composants lâchement couplés qui peuvent être testés individuellement
  - Instrumenter le code et faire du logging

Considérations à propos du déploiement

- L'option de base pour une application Web prévoit son déploiement, côté serveur, sur une seule machine, équipée typiquement avec
  - Serveur Web (ex. : Apache HTTP Server)
  - Serveur d'applications (ex. : Apache Tomcat)
  - Système de gestion de bases de données (ex. : mySQL)
- Une telle solution présente plusieurs avantages (pas de communication avec d'autres serveurs, accès sécurisé aux données, facilité de gestion) mais ne passe pas à l'échelle
- Les solutions distribuées utilisent plusieurs machines connectées sur le réseau
- SGBD et autres services délocalisés sur des machines dédiées

Solution distribuée moderne



Merci de votre attention