Bases de données et environnements distribués
Chapitre I : Architecture logicielle technologies de developpement en environnement distribué

Éric Leclercq

Département IEM / Laboratoire LE2i

Septembre 2016

émail : Eric.Leclercq@u-bourgogne.fr
http://ludique.u-bourgogne.fr/leclercq/
http://ufrsciencestech.u-bourgogne.fr

Contenu de la partie de l'UE BD-ED

- Architecture des systèmes d'information modernes : problèmes de distribution, de couplage
- Intergiciels objets distribués
- Intergiciels pour les communications asynchrones et environnement mobiles
- Transactions distribuées
- Persistance dans les architectures de SI distribués
- Intergiciels pour les masses de données (*Big Data*)
- Modèles de composants et serveurs d'applications (Java EE /J2EE)

Volume horaire : 14h CM, 8h TD, 16h 2x8hTP

Ce cours ne traite pas des web services, couverts dans un autre module. Il ne s'agit pas d'un cours sur le développement d'applications.

Objectifs

Connaissances :

- caractéristiques des intergiciels en environnement distribué
- comprendre problématiques liées à la qualité, à l'évolutivité, à la disponibilité, à l'efficacité des applications utilisant des intergiciels mais aussi des architectures distribuées
- type d'architectures de SI et patrons d'interaction de composants distribué

Compétences :

- savoir choisir un intergiciel en fonction du contexte applicatif
- savoir utiliser plusieurs intergiciels dans un environnement complexe (multi-acteurs)
- définir une architecture logicielle pour la communication dans un SI distribué
- développer et déployer des composant JEE
- utiliser l'injection de dépendances, et un framework

Thématiques et questions transversales

- Qualité de service, cohérence des données (distribuées)
- Génie Logiciel, outils et bonnes pratiques de développement :
  - problèmes de couplage, de version, etc.
  - pattern de développement : délégation, objectFactory, MVC, etc.
  - utilisation d'un environnement de développement (Éclipse, Netbeans)
  - utilisation d'un système de gestion de révision de code (Git, Subversion ou CVS)
  - construction d'application / projets : ant, Maven
  - tests unitaires (JUnit), pre/post condition (JML)
  - gestion des bugs (Mantis) , de la documentation (Javadoc, Wiki, Doxygene), etc.
- Modélisation de SI : architecture/urbanisation de système d'information

Pré-requis

- Langage orienté objet : Java et C++
- Connaissance d'au moins un système de gestion de base de données relationnelle : PostgreSQL, Oracle
- Expérience pratique de JDBC
- Programmation TCP/IP et notion de socket
- Maîtrise d'un SE Unix (en ligne de commande)
- Bases UML (diagramme de classes et des cas d'utilisation)

Bibliographie intergiciels et composants

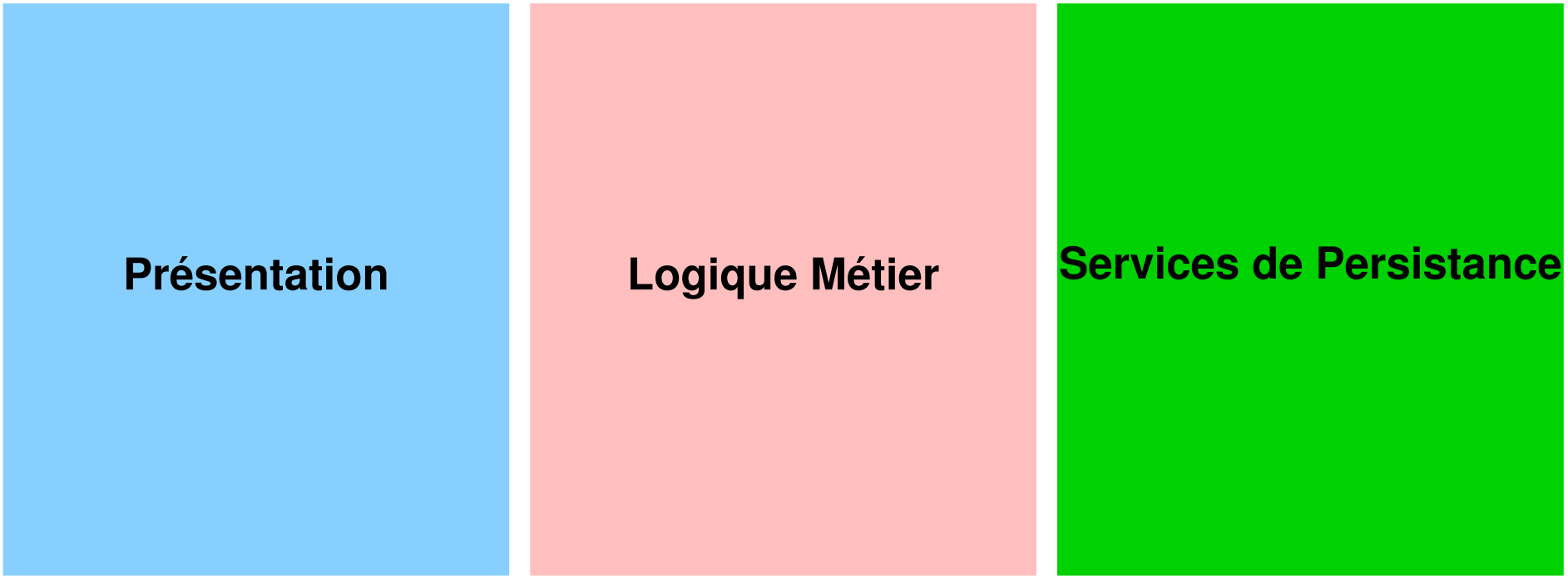
- *Java, plus rapide, plus léger*, Bruce Tate, Justin Gehtland, Philippe Ensarguet , Frédéric Laurent, O'Reilly (Novembre 2004), 250 pages, ISBN 2841773124
- *Les cahiers du programmeur J2EE*, Jérôme Molière, Eyrolles (Janvier 2005), 219 pages, ISBN 221211541
- *Java Enterprise : in a Nutshell*, Jim Farley, William Crawford, Prakash Malani, 3eme édition, O'Reilly (Novembre 2005)
- *Architectures réparties en Java*, Annick Fron, Dunod (2007), 209 pages, ISBN978-2-10-051141-9.
- *Les cahiers du programmeur Java EE 5*, Antonio Goncalves, Eyrolles (Mai 2007), 330 pages, ISBN 978-2-212-12038-7.
- *J2EE 1.4*, James Weaver, Kevin Mukhar, Jim Crume, Eyrolles (2004), 640 pages, ISBN 2-212-11484-2

Bibliographie UML et systèmes d'information

- *Advanced Systems Design With Java, UML And MDA*, Kevin Lano, Butterworth-Heinemann (Avril 2005), 416 pages, ISBN 0750664967
- *Developing Enterprise Java Applications with J2EE and UML*, Khawar Zaman Ahmed, Cary E. Umrysh, 288 pages, Addison-Wesley Professional, (Decembre 2001), ISBN 0201738295
- *Building Web Applications with UML (2nd Édition)*, Jim Conallen, 496 pages, Addison-Wesley (Octobre 2002), ISBN 0201730383
- *UML : Modéliser un site e-commerce*, Pascal Roques, Eyrolles, 2002, Les cahiers du programmeur, 152 pages, ISBN 2212110707

Les applications des SI

Une première approche conduit à un découpage simpliste des différentes fonctionnalités (côté programmeur).



Le modèle MVC

Définition :

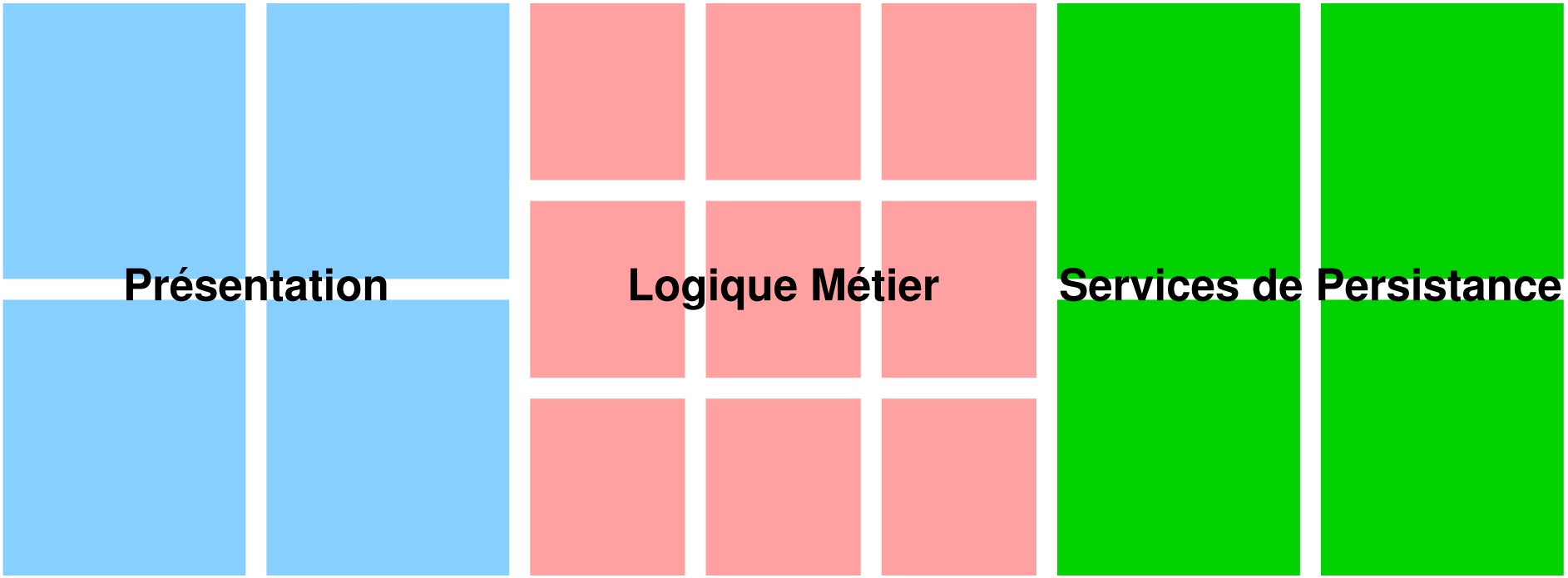
Le modèle de développement MVC (Modèle-Vue-Contrôleur/ Model-View-Controller) est un pattern de conception (architecture) qui consite à diviser une application en :

- *un modèle (modèle de données et traitements associés),*
- *une vue (interface utilisateur ou couche de présentation)*
- *un contrôleur (logique de contrôle ou logique applicative)*

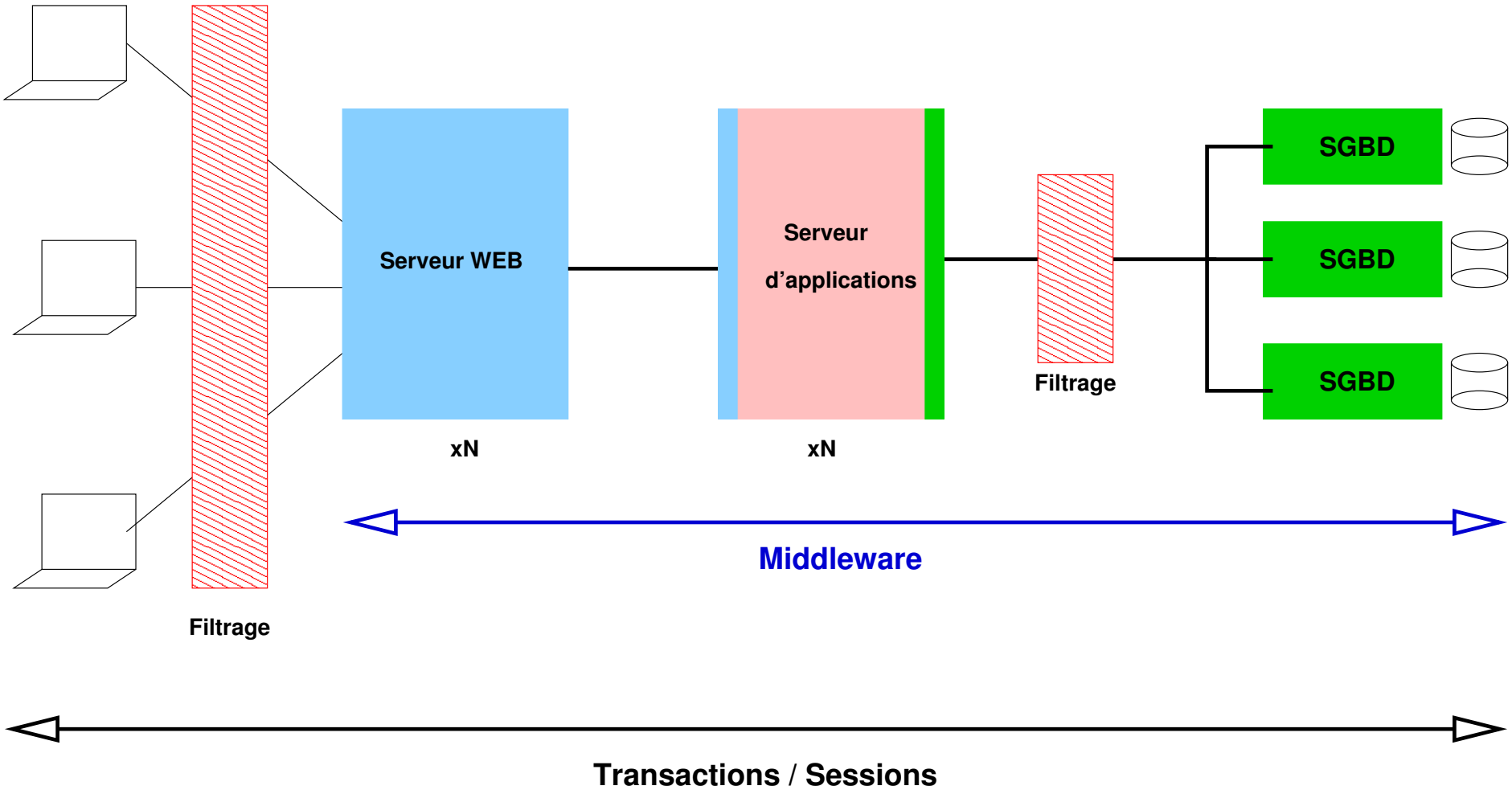
À l'origine le modèle MVC à été développé à la fin des année 70 pour le langage smalltalk.
Le modèle MVC en imposant la séparation entre les données, la présentation et les traitements, réduit le **couplage** entre les couches et permet une meilleurs évolutivité de l'application.

Influences de la distribution

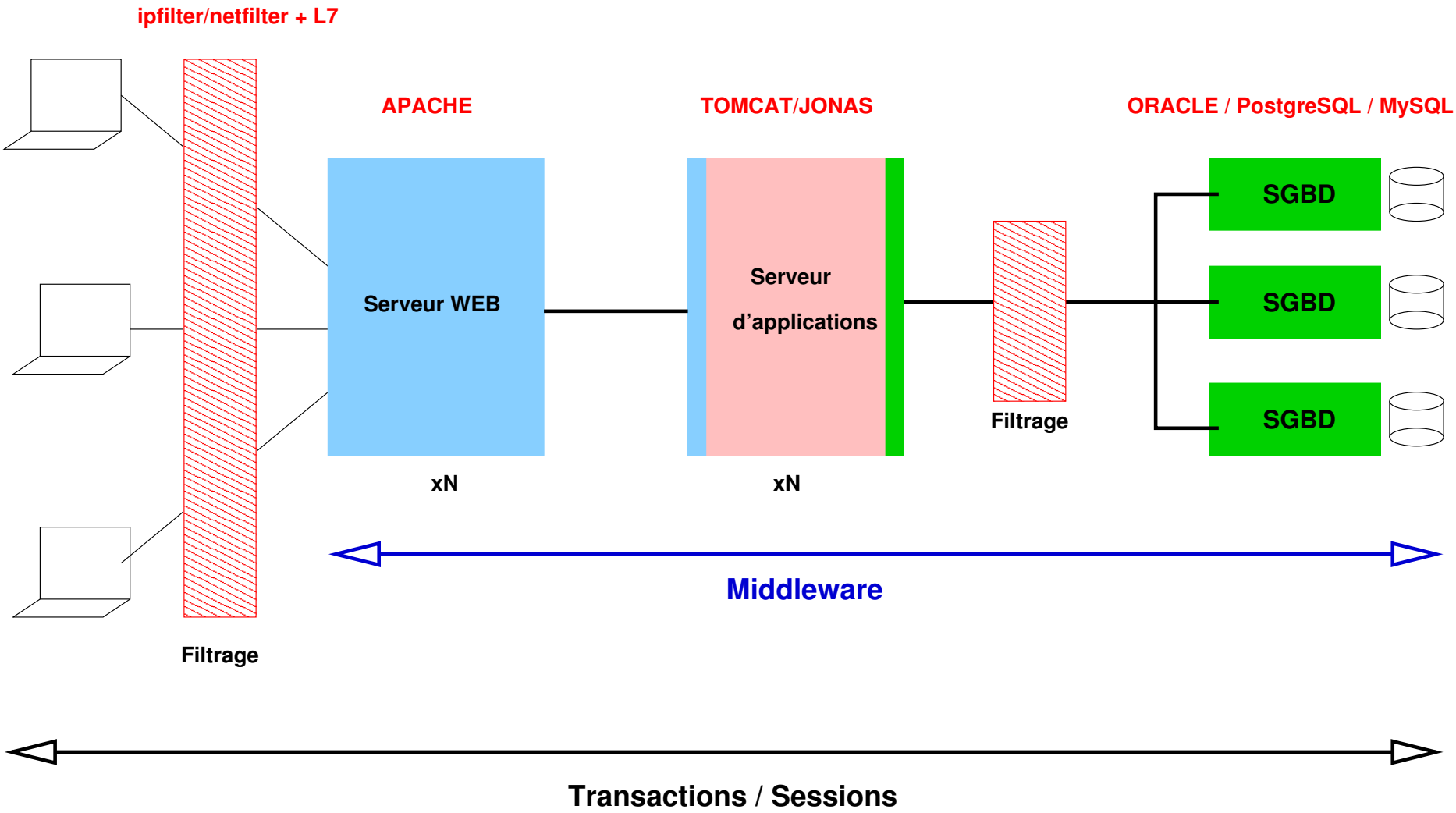
Le développement des réseaux et des technologies distribuées influe sur chaque des couches.



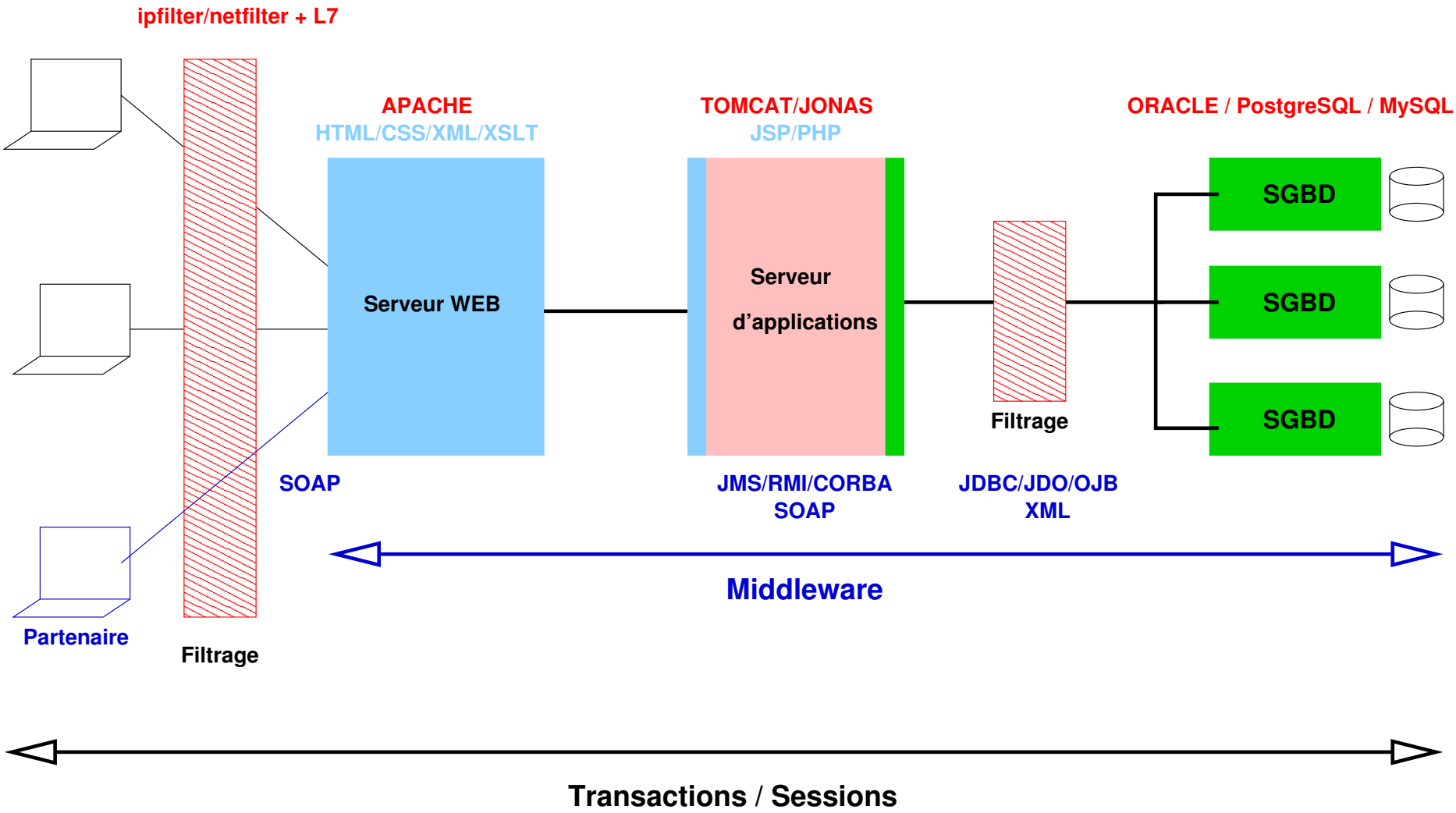
Architecture de SI modernes



Architecture de SI modernes



Architecture de SI modernes

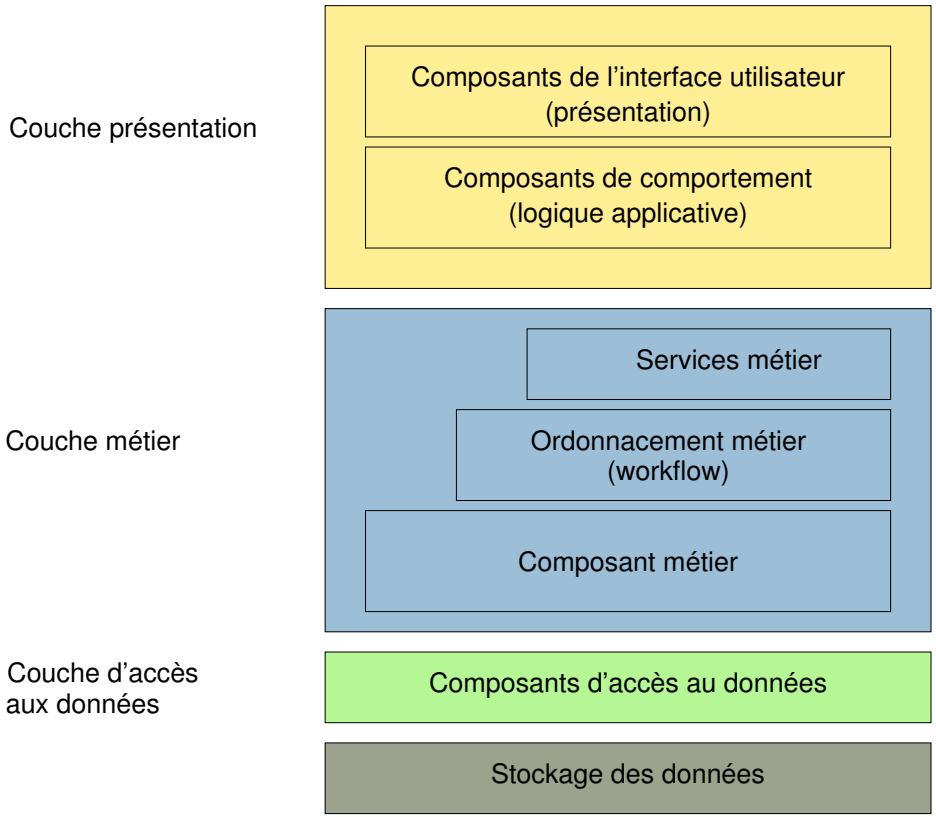


Critiques et évolutions du modèle MVC

- problème des contrôleurs multiples (couplage)
- rôle du contrôleur mal défini = logique métier, applicative ?
- architectures multi-tiers non prises en compte

Pour réduire le couplage entre les technologies et le code (problème d'intrusion des technologies dans le code métier) on utilise les principes d'injection de dépendance ou d'inversion de contrôle.

Critiques et évolutions du modèle MVC



À l'issue du cours vous devrez être capable de citer plusieurs technologies utilisées dans chacune des couches.

Les différents types d'interactions C/S

Rappels :

- C/S de données ;
- C/S de procédures ;
- C/S d'affichage ;
- C/S mixte.

Exercice :

Pour chaque type d'interaction C/S citer plusieurs protocoles de couche 7 basés sur TCP/IP.

Abstractions des interactions C/S

Une première classification des différents modes d'interaction C/S selon l'intégration dans le code applicatif :

- Interaction basées sur un protocole standard ou ad-hoc (développement long)
- Utilisation d'une extension de la notion d'appel de procédure (liaison statique)
- Interaction de plus haut niveau basée sur la notion d'objets distribués (intergiciels)

Comment traiter les communication asynchrones ?

Problématique(s) des applications Web

- Gestion des transactions distribuées et des transactions longues
- Prise en compte des évolutions technologiques : indépendance relative du code métier vis à vis des technologies utilisées → interopérabilité et substituabilité des composants logiciels
- Problèmes de couplage entre le code métier et les technologies de communication ou de persistance
- Adéquation des protocoles de communication avec la sécurité réseau
- Intégration de l'architecture logicielle dès la conception