université de BORDEAUX

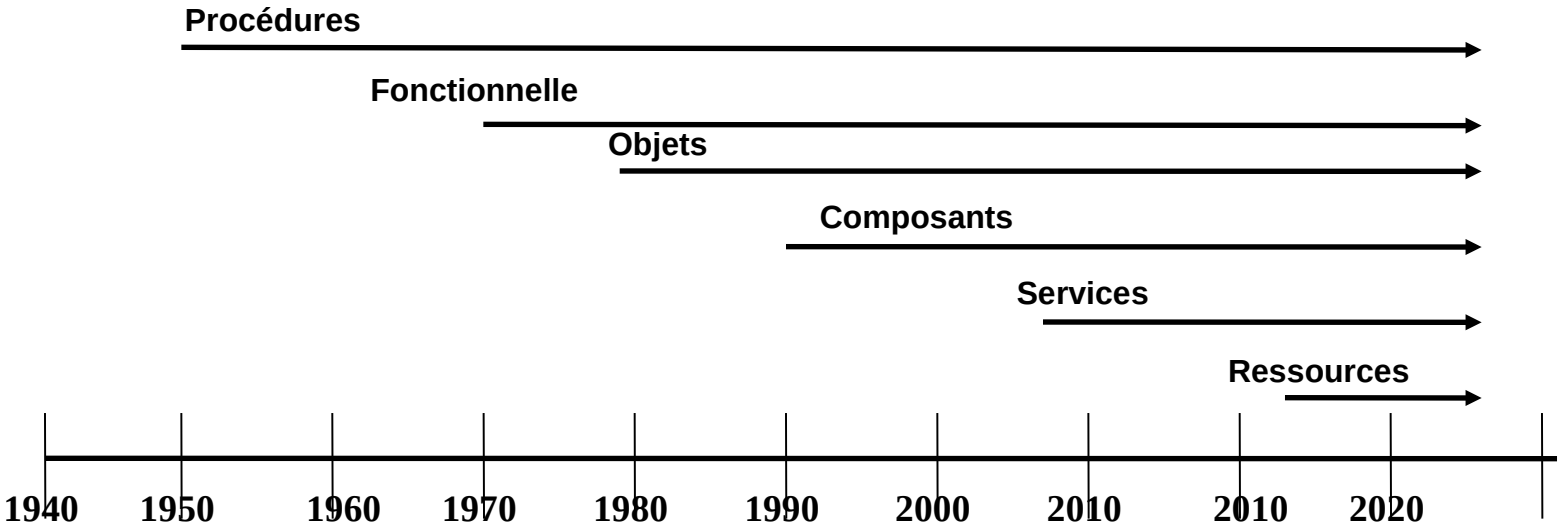
Cours 1 : Les Web Services et Architectures Orientées Services

Février 2024

- Version 1.13 -

université de BORDEAUX

Différents Types de programmation



1. Programmation procédurale
2. Programmation Objet
3. Émergence du modèle composant
4. Orientation Service
5. Orientation Ressource
6. Programmation fonctionelle

université de BORDEAUX

Programmation procédurale – Définition

- **La programmation procédurale est un paradigme qui se fonde sur le concept d'appel procédural.**
- **Une procédure, aussi appelée *routine*, *sous-routine* ou *fonction* (à ne pas confondre avec les fonctions de la programmation fonctionnelle reposant sur des fonctions mathématiques), contient simplement une série d'étapes à réaliser. N'importe quelle procédure peut être appelée à n'importe quelle étape de l'exécution du programme, y compris à l'intérieur d'autres procédures, voire dans la procédure elle-même (récursivité).**

Source wikipedia

Programmation procédurale – Caractéristiques

- **La programmation procédurale permet:**
  - **la réutilisation de code à différents emplacements dans le programme sans avoir à le retaper (factorisation) et donc une amélioration de la maintenabilité de celui-ci**
  - **Une simplification de la compréhension d'un programme; cependant, elle peut induire des « effets de bord », c'est-à-dire la possibilité pour une procédure qui prend des arguments de modifier des variables extérieures à elle même (variables de contexte plus global que la procédure).**

Programmation procédurale – Caractéristiques

- La programmation procédurale est une avancée majeure dans « l'informatique » car elle introduit la notion de modularité permettant :
  - De segmenter une application en module en théorie sans effet de bord externes aux données manipulées
  - De « distribuer » les développements entre différents programmeur
  - De faciliter les tests des applications
  - De faciliter la maintenance

Programmation Objet - Définition

- **La programmation orientée objet (POO), ou programmation par objet, est un paradigme de programmation informatique**
- **Il consiste en la définition et l'interaction de briques logicielles appelées objets ; un objet représente un concept, une idée ou toute entité du monde physique.**
- **Un Objet possède une structure interne et un comportement, et il sait interagir avec ses pairs.**

Concevoir une application informatique consiste donc à représenter ces objets et leurs relations

Source wikipédia

Programmation Objet - Définition

- **Concrètement, un objet est une structure de données valuées et cachées qui répond à un ensemble de messages. Cette structure de données définit son *état* tandis que l'ensemble des messages qu'il comprend décrit son *comportement*.**

Programmation Objet – Caractéristiques

- La programmation objet permet :
  - L'héritage
  - Le typage/polymorphisme
  - La redéfinition
  - L'encapsulation
  - …

Les caractéristiques d'un langage objet étant source de débat « sans fin » se rapporter à votre cours de POO.

- La programmation objet étend les concepts apportés par la programmation procédurale en limitant encore les effets de bord.
  - Simplification de la compréhension du code
  - Factorisation du code et réutilisation
  - Facilitation de la maintenance
  - Testabilité
  - Segmentation des développements

Elle induit toutefois des charges de conception non négligeable car indispensable à la qualité des

Programmation orientée composant - Définition

- **La programmation orientée composant (POC) consiste à utiliser une approche modulaire de l'architecture d'un projet informatique, ce qui permet d'assurer au logiciel une meilleure lisibilité et une meilleure maintenance. Les développeurs, au lieu de créer un exécutable monolithique, se servent de composants réutilisables.**

Source wikipedia

Modèle Composant - Caractéristiques

- **Intérêt des composants**
  - **Indépendance des composants à la Compilation**
  - **Granularité: Faciliter la création de grandes applications**
  - **Réutilisation: Composants = boîtes noires indépendantes**
  - **Programmation: Facilitée (parfois !)**
  - **Extensibilité: Compléter un composant sans effets de bord**
  - **Exécution: Maîtrise du cycle de vie**

université de BORDEAUX

Modèle Composant – Les apports

- Même si l'approche composant est une avancée majeure dans les architectures logicielles, elle est toujours très proche de « l'informatique technique ».
- L'approche « Service » implique une vision globale.
- L'informatique devient une fonction support aux autres métiers de l'entreprise. Il est donc nécessaire de comprendre ces métiers et leurs fonctionnements.
- La mise en place d'une telle architecture nécessite l'implication de tous les services.

Modèle SOA - Définition

- L'architecture orientée services (calque de l'anglais Service Oriented Architecture, ou SOA) est un modèle d'interaction applicative qui met en œuvre des services (composants logiciels) :
  - avec une forte cohérence interne (par l'utilisation d'un format d'échange pivot, le plus souvent XML),
  - et des couplages externes « laches » (par l'utilisation d'une couche d'interface interopérable, souvent un service web WS-*).
- Le service est une action exécutée par un « fournisseur » (ou « producteur ») à l'attention d'un « client » (ou « consommateur »), cependant l'interaction entre consommateur et producteur est faite par le biais d'un médiateur (qui peut être un bus) responsable de la mise en relation des composants.

université de BORDEAUX

Modèle SOA - Caractéristiques

- **L'objectif d'une architecture orientée services est donc de décomposer une application en un ensemble de fonctions basiques, appelées services, fournies par des composants et de décrire finement le schéma d'interaction entre ces services.**
- **Ces services doivent être conçus pour ne pas être limités à un usage pour une seule application.**

Modèle SOA - - Caractéristiques

- **Une architecture orientée services permet d'obtenir tous les avantages d'une architecture client-serveur et notamment :**
  - **Une modularité permettant de remplacer facilement un service par un autre .**
  - **Une réutilisabilité possible des services.**
  - **De meilleures possibilités d'évolution (il suffit de faire évoluer un service ou d'ajouter un nouveau service) .**
  - **Une plus grande tolérance aux pannes.**
  - **Une maintenance facilitée .**

- **La SOA est un concept d'architecture, la WSOA (*WebService Oriented Architecture*) en son implémentation avec des WebServices.**

- **Il est possible d'implémenter un service logique (concept SOA) en un ou plusieurs Services Web.**

- Dans une architecture orientée service, il est possible d'implémenter tout ou partie des services sous formes de Web Services.
- Une application se découpe en plusieurs processus métier.
- Il convient donc de coordonner ces différents processus pour effectuer correctement les traitements souhaités.
- Cet coordination s'appelle « orchestration »

- La coordination des services se fait à travers BPEL (Business Process Execution Language).
- La version 2.0 a été normalisé par l'OASIS en mars 2007.
- BPEL est un langage XML permettant de modéliser les processus métier.
- BPEL ne se limite pas au Web Service mais permet d'orchestrer des processus définis dans plusieurs technologies.

Modèle Ressource - Définition

- **Même si l'orientation service a été un grand pas en avant dans la « simplification » des modèles architecturaux, la complexité d'implémentation des différentes « normes » et l'importante consommation de ressources « machines » liée aux traitement des documents XML à conduit à repenser le modèle.**
- **Tout naturellement les modèles actuels tendent à se rapprocher des fondamentaux du Web : un objet du Web est accessible par son URL à travers le protocole HTTP**

Modèle Ressource - Définition

- **Selon le Gartner :**

  ROA = SOA + WWW + REST

- **En conséquence, une architecture orientée ressource est une architecture de service accessible à travers le Web et utilisant les paradigmes REST.**

Modèle Ressource - Définition

- L’acronyme REST signifie REpresentional State Transfert.

- REST est un ensemble de principe définissant comment les standards du Web (HTTP, URL, ..) devraient être utilisés.

Modèle Ressource - Caractéristiques

- **Les 5 principes fondateurs de REST sont :**
  - **Identifier chacun de vos éléments**
  - **Lier vos éléments entre eux**
  - **N'utiliser que les standards**
  - **Une ressource peut avoir plusieurs représentation**
  - **Communiquer entre vos ressources sans état**

Modèle Ressource - Caractéristiques

- **Identifier chaque chose :**
  - **Sur le Web la méthode pour identifier des éléments est d'utiliser une URI pour chaque éléments. Cela permet dans un espace de nommage global d'identifier de manière unique chaque éléments du Web.**
  - **Ce principe est à appliquer pour chacun des élément que vous souhaitez « partager »**

Modèle Ressource - Caractéristiques

- **Lier vos éléments entre eux :**
  - **Le concept d'hyperlien permet la navigation sur un site Web. Toutefois, ce concept n'est pas réservé à la navigation humaine. En ce basant sur le principe précédent chaque élément de votre application doit se représenter sous forme d'URI.**
  - **En conséquence, utiliser des hyperliens pour « représenter » vos éléments dès que possible.**

Modèle Ressource - Caractéristiques

- **N’utiliser que les standards :**
  - **Le protocole sous-jacent au Web est HTTP. Il propose un certain nombre de méthode appelée Verbe dans la nomenclature REST.**
  - **Pour que votre architecture soit RESTful il faut que vous respectiez les significations implicites de chacun de ceux-ci. Cela permettra à votre application d’être complètement intégrée au Web et facilitera les interactions entre elle et ces clients.**

Modèle Ressource - Caractéristiques

- Parmi les différentes spécifications HTTP on peut isoler les verbes suivants :
  - GET permet de récupérer une « représentation » de la ressource
  - PUT/PATCH permet de mettre à jour une ressource
  - DELETE permet de supprimer une ressource
  - POST permet de créer une ressource
- Les trois premières méthodes doivent pouvoir être rejouer N fois et engendrer le même résultat (concept d'idempotence)

Modèle Ressource - Caractéristiques

- **Une ressource doit avoir plusieurs représentation :**
  - **Parmi les éléments offerts par http le header Accept permet de spécifier un type/mime pour le résultat. Ce principe permet pour deux requête identique en terme d'URI, de verbe mais différents en terme de type mime d'obtenir des résultat différent. On peut ainsi imaginer renvoyer pour une réservation de salle une représentation au format Icalc ou XML**

Modèle Ressource - Caractéristiques

- **Communiquer sans etat :**
  - **Un service « REST » ne doit pas conserver d'état Client entre deux requêtes. Cela permet de limiter la charge serveur mais aussi cela facilite aussi la reprise sur incident car un client n'est pas obligé de s'adresser au même serveur tout au long de son processus métier.**

Programmation fonctionnelle – Définition

- La programmation fonctionnelle est un paradigme de programmation qui considère le calcul en tant qu'évaluation de fonctions mathématiques.
- Le paradigme fonctionnel n'utilise pas de machine à états pour décrire un programme, mais un emboîtement de fonctions qui agissent comme des « boîtes noires ». Chaque boîte possédant plusieurs paramètres en entrée mais une seule sortie

Source wikipedia

Programmation fonctionnelle - Caractéristiques

- **Les langages fonctionnels ont comme autre propriété la transparence référentielle. Ce terme recouvre le principe simple selon lequel le résultat du programme ne change pas si on remplace une expression par une expression de *valeur égale.* Ainsi les effets de bord sont impossible dans les langages fonctionnels.**

Programmation fonctionnelle - Caractéristiques

- Du fait de la **transparence référentielle** il est très facile de définir des plans de tests pour des applications utilisant la programmation fonctionnelle
- Les « optimisations » des programmes fonctionnels sont « simple » si les paramètres en entrée sont les même le résultat sera le même donc il n'est pas indispensable d'exécuter la fonction

Théorème dit de « CAP »

- **Le développement de l'exécution des applications sur des infrastructures réparties amène à se poser la question sur le fonctionnement de nos applications.**
- **En particuliers sur les aspects suivants :**
  - **La « Cohérence » (Consistency)**
  - **La Disponibilité (Availability)**
  - **La « Tolérance à la distribution » (Partition Tolérance)**

Théorème dit de « CAP »

- La « Cohérence » signifie qu'une données est identique sur la totalité d'un système.
- La « Disponibilité » signifie que le système répond systématiquement même si certains éléments sont défaillants.
- La tolérance au partitionnement signifie que l'on peut répartir la charge d'exécution de l'application sur plusieurs infrastructures.

Théorème dit de « CAP »

- **Le théorème de CAP précise qu'un système ne peut, EN MEME TEMPS, respecter plus de deux des dimensions précédemment exposées.**