Introduction à Spring

Pourquoi Spring ?

Livres

- **Livres**
  - « Spring in Action »
    - https://www.amazon.fr/Spring-Action-Craig-Walls/dp/161729120X
  - « Better, faster, lighter Java »
    - https://www.amazon.fr/Better-Faster-Lighter-Java-Bruce-ebook/dp/B0028N4WHE
  - « Head first design patterns »
    - https://www.amazon.fr/Head-First-Design-Patterns-Freeman-ebook/dp/B00AA36RZY/
  - « Patterns of enterprise application architecture »
    - https://www.amazon.fr/Patterns-Enterprise-Application-Architecture-Martin-ebook/dp/B008OHVDFM/
  - « Engineering software as a service : an agile approach using cloud computing »
    - https://www.amazon.fr/Engineering-Software-Service-Approach-Computing/dp/0984881247/

Aperçu historique (1)

- **Corba (ONG 1991)**
  - Common object broker architecture
  - Traitement distribué orienté objet
  - RMI / JNDI (Remote Method Invocation)
  - Notion de proxy / stub, d'annuaire, etc…
- **CGI (NCSA 1993)**
  - Common Gateway Interface
  - Remplacer le contenu statique des serveurs web par du contenu dynamique généré à la volée par du code / processus système fork
- **Fallacies of distributed computing (Sun Microsystem 1994)**
  - Le réseau est fiable
  - La latence est nulle
  - La bande passante est infinie
  - Le réseau est sécurisé
  - La topologie ne change pas
  - Il y a un seul administrateur
  - Le transport ne coûte rien
  - Le réseau est homogène

Aperçu historique (2)

- **Design patterns (Gang of Four 1994)**
  - Formaliser les bonnes pratiques de conception des logiciels à travers des « design pattern »
- **PHP (Lerdorf 1994)**
  - Langage interpété adapté au développement de scripts CGI
- **Apache httpd (1995)**
  - Basé sur le serveur Web NCSA
  - Remplacement de la CGI par des modules et une architecture multi-threads
- **Java (Sun Microsystem 1995)**
  - Orienté objet
  - Garbage collection
  - Code once / run many
  - Langage côté client
- **Weblogic (Weblogic 1995)**
  - Premier serveur d'applications JEE (1995-1998)

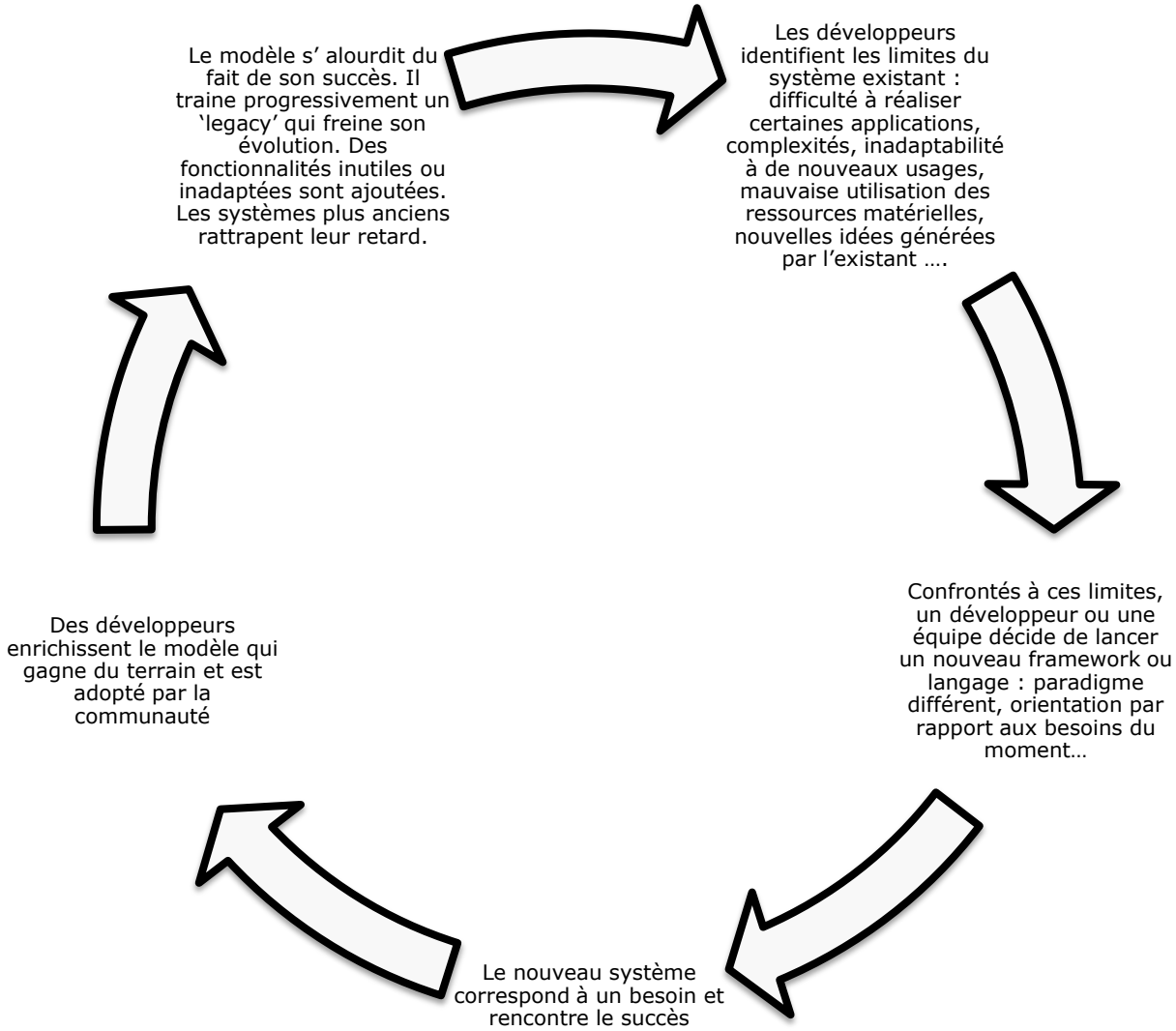
Aperçu historique (3)

- **Servlets API (Sun Microsystem 1997)**
  - Java côté serveur + adapté au Web
  - Servlet container = Tomcat
- **Théorème CAP (Brewer 1998)**
  - Introduit des limites théoriques aux transactions ACID dans un environnement distribué
  - ACID = Atomique, Cohérente, Indépendante, Durable
- **Tomcat Server (Apache 1999)**
  - Container de servlet
- **JEE (Sun Microsystem 1999)**
  - Java Enterprise Edition
  - Java beans
- **Jboss (Fleury 1999)**
  - Serveur d'applications compatible JEE

Aperçu historique (4)

- **Hibernate framework (King 2001)**
  - ORM / interface entre la couche objet et les bases de données
  - Simplification des développements + bonne pratique
- **Django (Holovaty-Willison 2003)**
  - Serveur web en python implémentant le MVC
- **Spring framework (Johnson 2003)**
  - Container léger
  - Implémentation des principaux design patterns de la JEE sans la complexité de la JEE
  - Retour aux sources (POJO)

L'évolution des frameworks et langages



Pourquoi Spring?

- **Les leçons tirées de la JEE, de ses qualités et de ses échecs**
- **Un écho à des initiatives non-java qui ont apporté de la simplicité et de la flexibilité aux développements Web**
  - RoR
  - Python/Django
  - ASP.Net
- **Une approche plus mature du développement**
  - Design pattern
  - Retour d'expérience sur les systèmes distribués
  - …

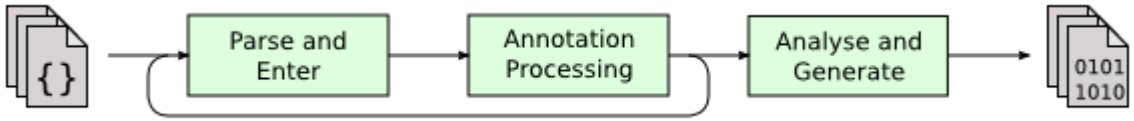
Questions ?

Introduction à Spring

Spring Framework

Qu'est ce que Java ?

- **Un langage**
  - Syntaxe
  - Sémantique
- **Un modèle de compilation**



- Source : http://openjdk.java.net/groups/compiler/doc/compilation-overview/index.html

Rappel concernant les annotations

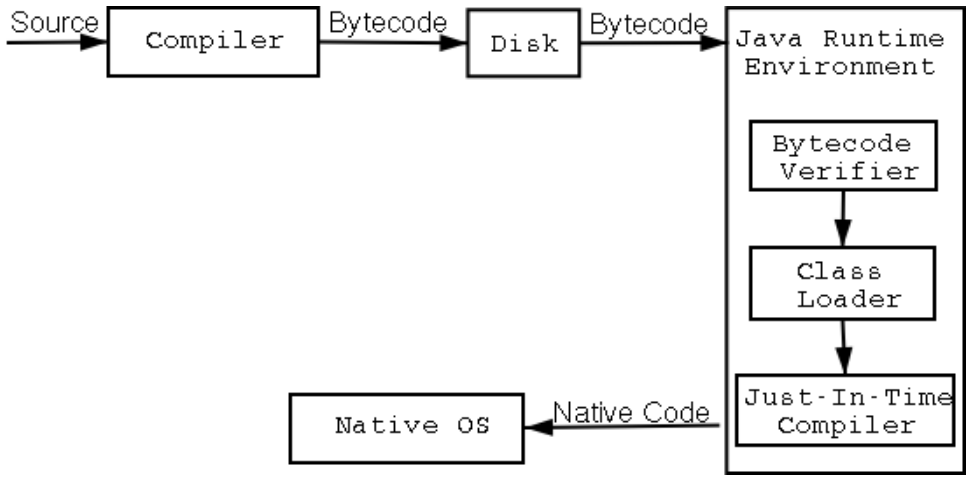
- **Rôle des annotations**
  - Transmises au compilateur pour modifier son comportement (suppression des warnings, …)
  - Utilisé par le compilateur ou des outils logiciels complémentaires pour générer du code java, des fichiers XML, de la documentation, etc…
  - Certaines annotations peuvent être accessibles au moment de l'exécution du code

- **Des annotations sont prédéfinies dans le langage Java**

- **Les développeurs et éditeurs de logiciel peuvent également définir leurs propres annotations**

Qu'est ce que Java ?

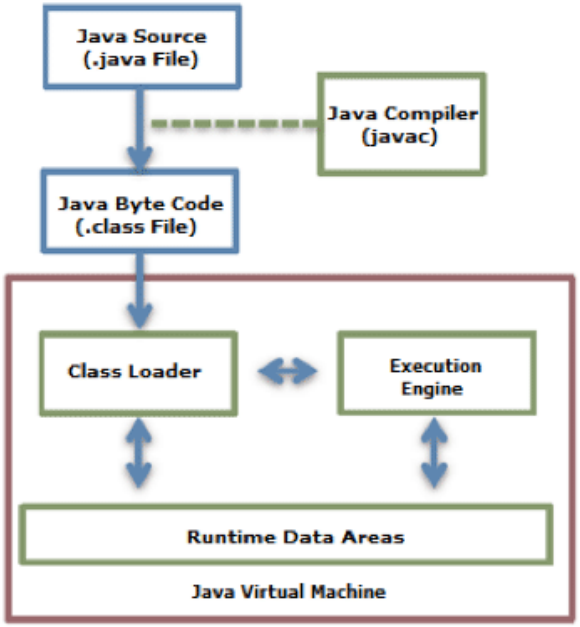
- **Un modèle d'exécution : byte code et JVM**



- Source : http://althing.cs.dartmouth.edu/local/www.acm.uiuc.edu/sigmil/RevEng/ch02.html

Qu'est ce que Java ?

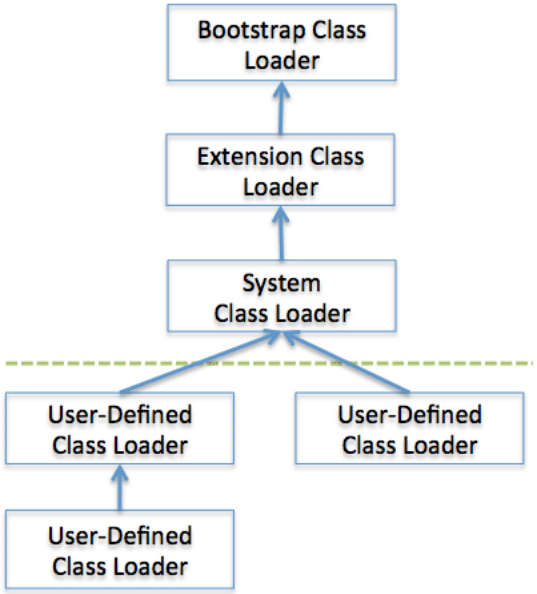
- **Un modèle d'exécution : byte code et JVM**



- http://www.cubrid.org/blog/dev-platform/understanding-jvm-internals/

Qu'est ce que Java ?

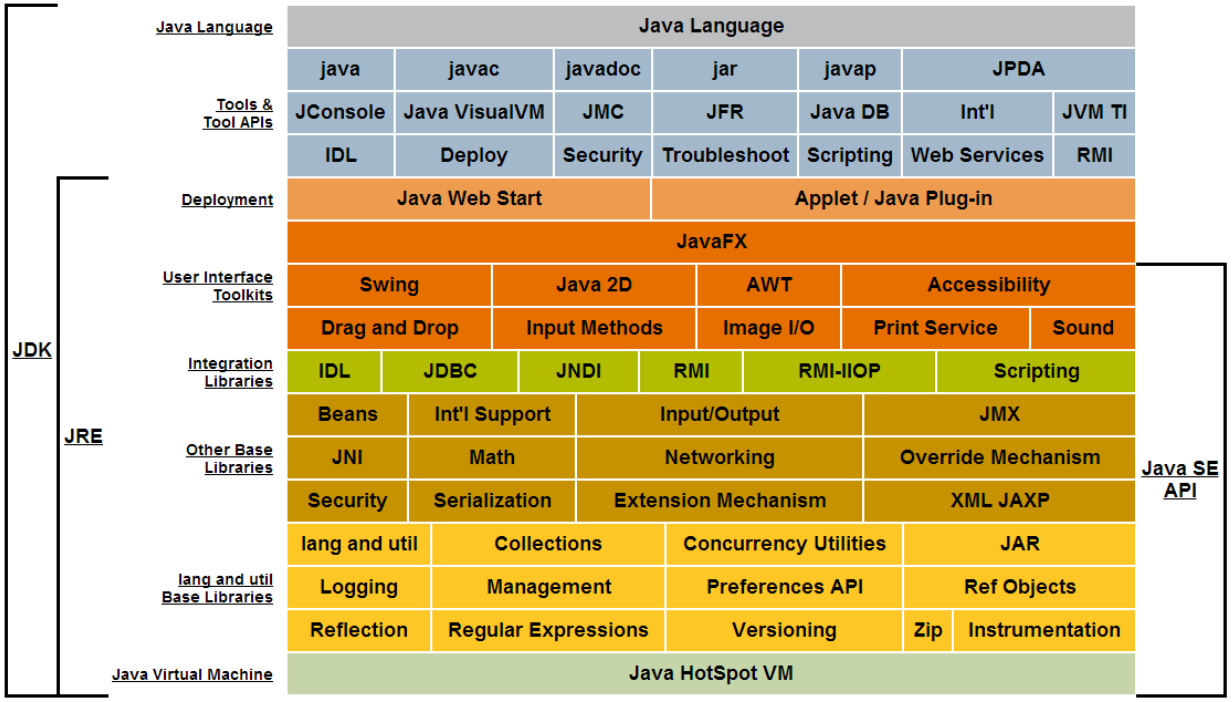
- **Une solution de chargement dynamiques des classes dans la JVM s'appuyant sur des références symboliques aux classes et interfaces**



- http://www.cubrid.org/blog/dev-platform/understanding-jvm-internals/

Qu'est ce que Java ?

- **Un ensemble de librairies et d'outils standards**



  - http://stackoverflow.com/questions/9057396/what-does-api-level-mean

- **La capacité à ajouter des librairies externes**
- **La capacité à construire ses propres librairies**
- **Des règles de gestion des espaces de nommage pour gérer le chargement des librairies**

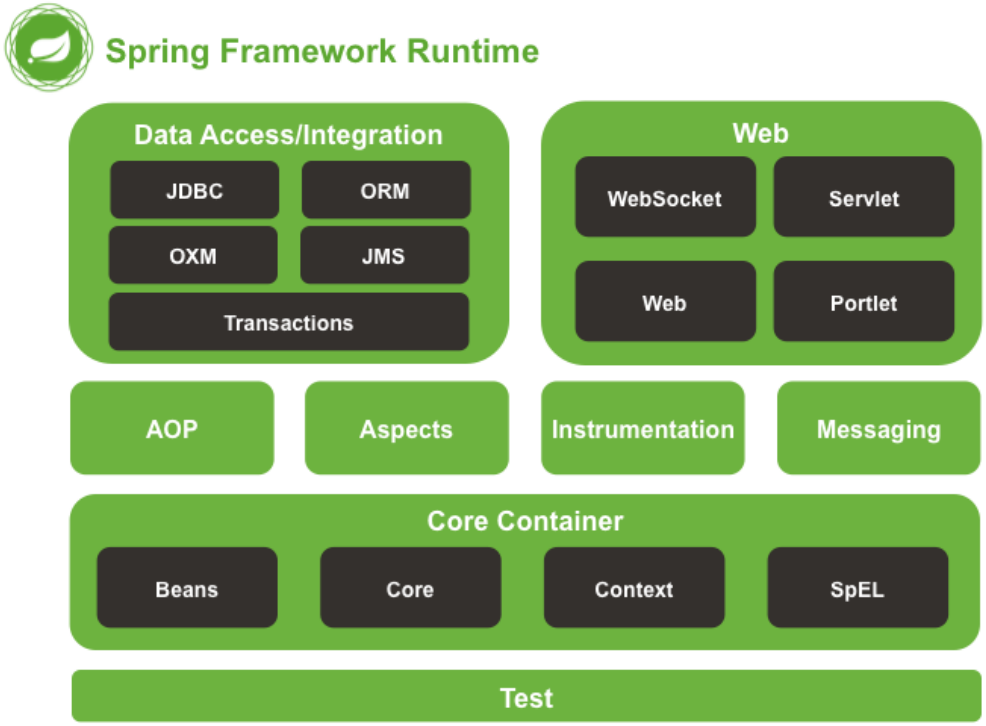
Qu'est ce que Spring ?

- Un ensemble de librairies Java
- Des fichiers XML de gestion des dépendances / de définition de contexte
- Des annotations qui peuvent se substituer à la configuration XML
- Des class loaders customisés permettant un chargement dynamique des classes Java en fonction du contexte
- Une solution de packaging automatique des développements pour construire une application: spring-boot

Qu'est ce que Spring ?

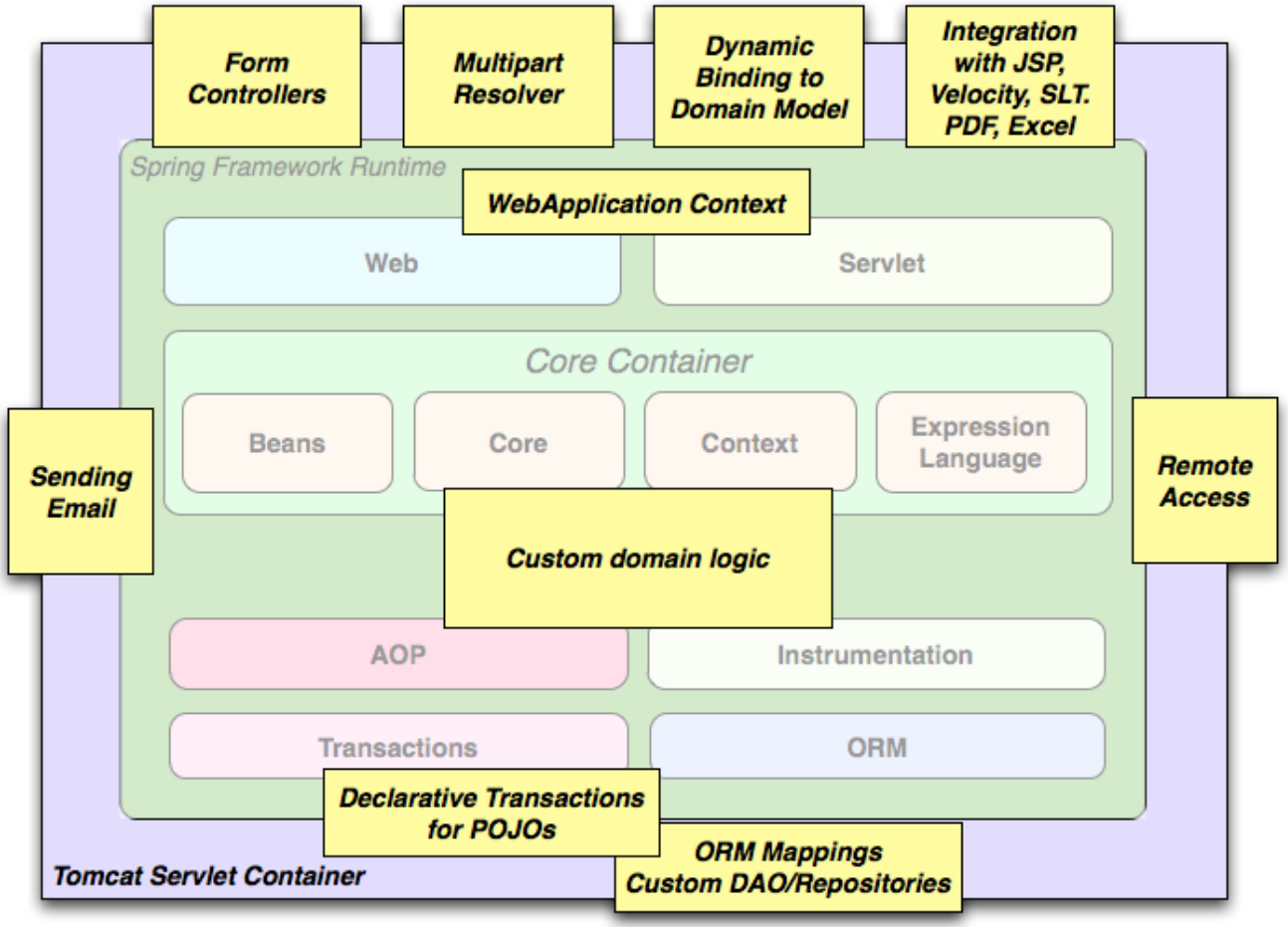
- **L'association de ces différents éléments simplifie le développement des applications en automatisant la création de design patterns fréquemment utilisés :**
  - MVC
  - IOC
  - AOP
  - …
- **à partir des objets Java standards (POJO = Plain Java Object) écrits par le développeur…**
- **en surchargeant le linkage de ces objets à travers des class loaders customisés s'appuyant sur le contexte contenu dans les fichiers XML ou les annotations**

Qu'est ce que Spring ?



- Source : https://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html

Qu'est ce que Spring ?



- Source : https://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html

Questions ?

Introduction à Spring

Retour sur les méthodes de développement

Au-delà du code (1)

Chiffres extraits d'un Rapport d'étude du SYNTEC sur les projets de mise en œuvre des ERP

62 % sont insatisfaits du résultat de ces projets :

53,5%

METIER

pour 18% les besoins, les attentes les objectifs ont été mal définis

pour 16%, des fonctionnalités inadaptées, sur ou sous dimensionnées

pour 9%, les utilisateurs n'ont pas été suffisamment intégrés au projet

pour 4%, la dimension stratégique du projet a été mal expliquée ou mal perçue

pour 13%, les engagements avec les prestataires ont été mal définis ou mal respectés

22,0%

MOYENS / MOBILISATION

pour 11%, déficit d'implication de la direction et du management

pour 11%, les ressources humaines et / ou budgétaires ont été insuffisantes

14,5%

METHODOLOGIE

pour 8%, le projet a été mal préparé ou mal conduit

pour 13%, les engagements avec les prestataires ont été mal définis ou mal respectés

10,0%

TECHNOLOGIE

pour 10%, les technologies ont été mal maitrisées

Au-delà du code (2)

Un projet de développement, c'est

- **Un travail d'équipe**
- **Une demande d'un client**
- **Une volonté de réaliser**
  - Sponsor
  - Attente des utilisateurs
- **Des contraintes**
  - Budget
  - Temps
  - Contrats
  - Environnement d'exécution
  - Domaine métier
  - Environnement applicatif

Au-delà du code (3)

- **Des matériaux de construction**
  - Langages
  - Librairies
  - Environnement d'exécution

- **Des méthodes pour coordonner l'ensemble**
  - Méthode en V
  - Scrum
  - XP
  - …

Une multitude de facteurs de succès ou d'échec du projet (1)

- **Choix et l'organisation de l'équipe projet**
  - Profils et compétences
  - Hiérarchie
  - Organisation physique
  - Moyens
  - Contrats
  - Relations ou cloisonnement entre les différents domaines de responsabilité (support / développement / production)

- **Formulation de la demande client**
  - Relations entre le client et les développeurs
  - La manière d'exprimer le besoin
  - L'expression du besoin

isITix
keep IT simple

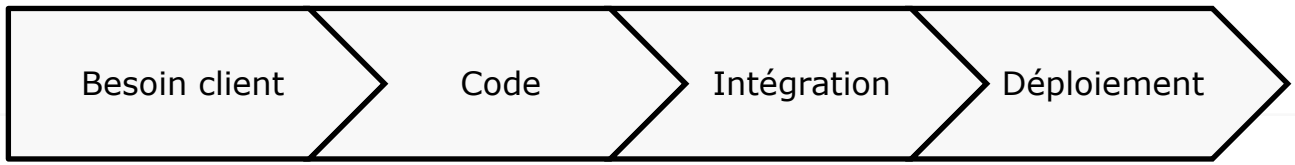
Une multitude de facteurs de succès ou d'échec du projet (2)

- **Compatibilité des contraintes et des objectifs**
  - Wishful thinking
  - Marché de dupes
  - La grenouille et le bœuf
  - Le marteau pilon pour enfoncer des clous
  - …
- **Choix des matériaux**
  - Effets de mode
  - Golden hammer
  - Coût/choix d'apprentissage
  - Oubli d'une partie des matériaux (serveurs, stockage, …)
  - Gestion du legacy
- **Méthode**
  - Tribale
  - Procédurière
  - Informelle
  - Document first
  - …

Une multitude de facteurs de succès ou d'échec du projet (3)

- **Beaucoup de projets sont condamnés avant d'avoir démarré**
  - Absence de volonté de réaliser
  - Choix politiques
  - Climat conflictuel
  - Politique commerciale du faiseur
  - Positionnement prix pour rentrer une affaire
  - Politique d'achat du demandeur
  - …

- **D'autres sont des échecs pour des questions de méthodologie**
  - Méthodologie de développement
  - Organisation de la chaîne de production



Recommandations

- De nombreux ensembles de recommandations ont été rédigés au fil du temps
- Certaines sont spécifiques à un langage ou à une technologie
- D'autres se réfèrent à une méthode particulière de développement (scrum, XP, merise,…)

Exemple 1 : Vision achat

- **Qualité…**
- **Délai…**
- **Budget**

Pick two

Exemple 2 : Java - extrait de better, faster, lighter java (1)

- **Code rules** : le code est la référence (code = documentation)
- **Embrace change** : accepter le changement plutôt que de le combattre
- **Strive for simplicity** : rechercher la solution la plus simple, être économe
- **Feedback** : demander régulièrement au client si le produit correspond à ses attentes

Exemple 2 : Java - extrait de better, faster, lighter java (2)

- **Automate testing : automatiser les tests**
  - Tests unitaires
  - Tests d'intégration
- **Integrate continuously : intégrer en continu le code produit**
- **Refactor : refactoriser le code**
  - define (test) => code => integrate => refactor

Exemple 3 : manifeste agile

We value

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following plan

Exemple 4 : Ruby on rails - Engineering software as as service (1)

DRY development: Don't Repeat Yourself

- Clarity via conciseness
- Refactoriser le code (pas de copier/coller)
- Instrumenter le code (métrique, intégration continue, gestion de version, configuration…)
- Automatiser

Exemple 4 : Ruby on rails - Engineering software as as service (1)

SOFA methods :

- **S**hort
- do **O**ne thing
- **F**ew Arguments
- single **A**bstraction level

Exemple 3 : Ruby on rails - Engineering software as as service (2)

SOLID class

- **Single responsibility**
- **Open/Closed : Open to extension, Closed to modification**
- **Liskov subsitution : une méthode qui fonctionne sur une classe T doit fonctionner dans les classes héritées de T**
- **dependency Injection : si deux classes dépendent l'une de l'autre mais que leur implémentation peut changer de manière indépendant, il est préférable de créer une interface commune, injectée dans les deux classes**
- **Demeter law : « talk to your friends. Don't get intimate with strangers »**
  - Une méthode peut uniquement appeler les autres méthodes de sa classe ou les méthodes des variables d'instance de sa classe. L'accès à d'autres méthodes est interdit

Exemple 3 : Ruby on rails - Engineering software as as service (3)

SMART User Stories :

- **S**pecific,
- **M**easurable,
- **A**chievable,
- **T**imeboxed

FIRST tests :

- **F**ast,
- **I**ndependent,
- **R**epeatable,
- **S**elf-checking,
- **T**imely

Questions ?

MERCI DE VOTRE ATTENTION