Éléments de Spring

Présentation du framework
SPRING et quelques nouveautés
de la version 2.0

Jean-Pierre PAWLAK
Jp.pawlak@wanadoo.fr

Notre plan

- Les grands principes
  - L'inversion de contrôle
  - L'intégration de technologies
  - La programmation orientée Aspect
- La structure générale
- Principales nouveautés de la version 2.0
- Les produits complémentaires
- Ressources
- Questions / réponses

Les objectifs de Spring

- Les objectifs initiaux (2002) étaient la simplification du développement, notamment sous J2EE
  - Simplifier sans réduire les possibilités
  - Permettre d'utiliser les meilleures techniques du développement orienté objet
  - Partir de l'expérience concrète de Rod Johnson et d'autres développeurs/architectes

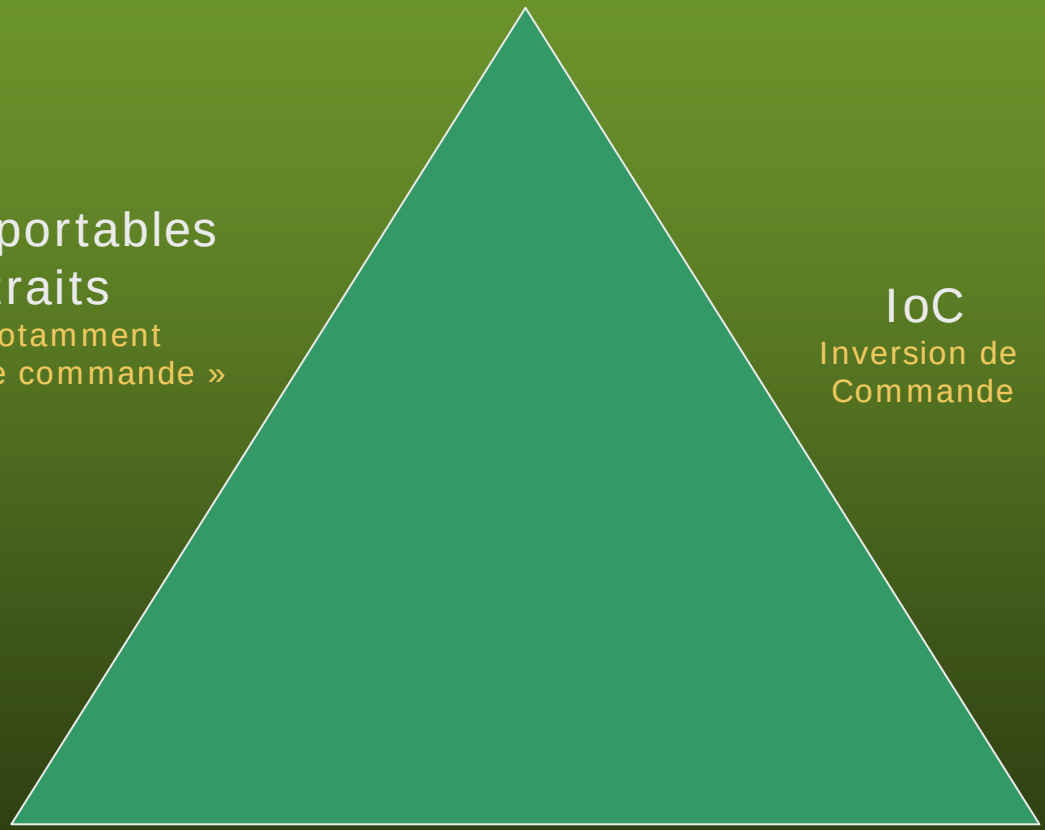
> Simple things should be simple and complex things should be possible
>
> *- Alan Kay*

> Unless simple things are simple, complex things are impossible
>
> *- Rod Johnson*

Les moyens : Les Pojos

- Réhabilitation des POJOs (« Plain Old Java Objects »)
  - Offrir une configuration sophistiquée adaptée à la complexité des applications réelles
  - Permettre l'application des services d'entreprise à ces POJOs de manière déclarative et non envahissante
- Exemples
  - Les POJOS peuvent devenir transactionnels sans la moindre allusion aux API transactionnelles
  - Les POJOs peuvent être managés au travers de JMX sans hériter de l'interface « MBean »

La puissance aux Pojos



IoC : l'inversion de contrôle ou principe de Hollywood

- SPRING utilise abondamment l'injection de dépendances au travers de son conteneur IoC
  - L'objet ne cherche plus ses collaborateurs
    - Il n'est plus dépendant de leur implémentation
    - Il n'est plus dépendant des stratégies de recherche
    - Il est testable isolément
    - Il est réutilisable
      - dans différents environnements
      - avec différentes implémentations de collaborateurs
  - En lieu et place, il définit des moyens d'injection de ses collaborateurs
    - En définissant des « setters »
    - En définissant des constructeurs paramétrés

IoC : Exemple « setter »

```
public class ExampleBean {
    private AnotherBean beanOne;
    private YetAnotherBean beanTwo;
    private int i;

    public void setBeanOne(AnotherBean beanOne) {
        this.beanOne = beanOne;
    }

    public void setBeanTwo(YetAnotherBean beanTwo) {
        this.beanTwo = beanTwo;
    }

    public void setIntegerProperty(int i) {
        this.i = i;
    }
}
```

```
<bean id="exampleBean" class="examples.ExampleBean">

    <!-- setter injection using the nested <ref/> element -->
    <property name="beanOne"><ref bean="anotherExampleBean"/></property>

    <!-- setter injection using the neater 'ref' attribute -->
    <property name="beanTwo" ref="yetAnotherBean"/>
    <property name="integerProperty" value="1"/>

</bean>

<bean id="anotherExampleBean" class="examples.AnotherBean"/>
<bean id="yetAnotherBean" class="examples.YetAnotherBean"/>
```

IoC : Exemple « constructeur »

```
public class ExampleBean {
    private AnotherBean beanOne;
    private YetAnotherBean beanTwo;
    private int i;

    public ExampleBean(
        AnotherBean anotherBean, YetAnotherBean yetAnotherBean, int i) {

        this.beanOne = anotherBean;
        this.beanTwo = yetAnotherBean;
        this.i = i;

    }
}
```

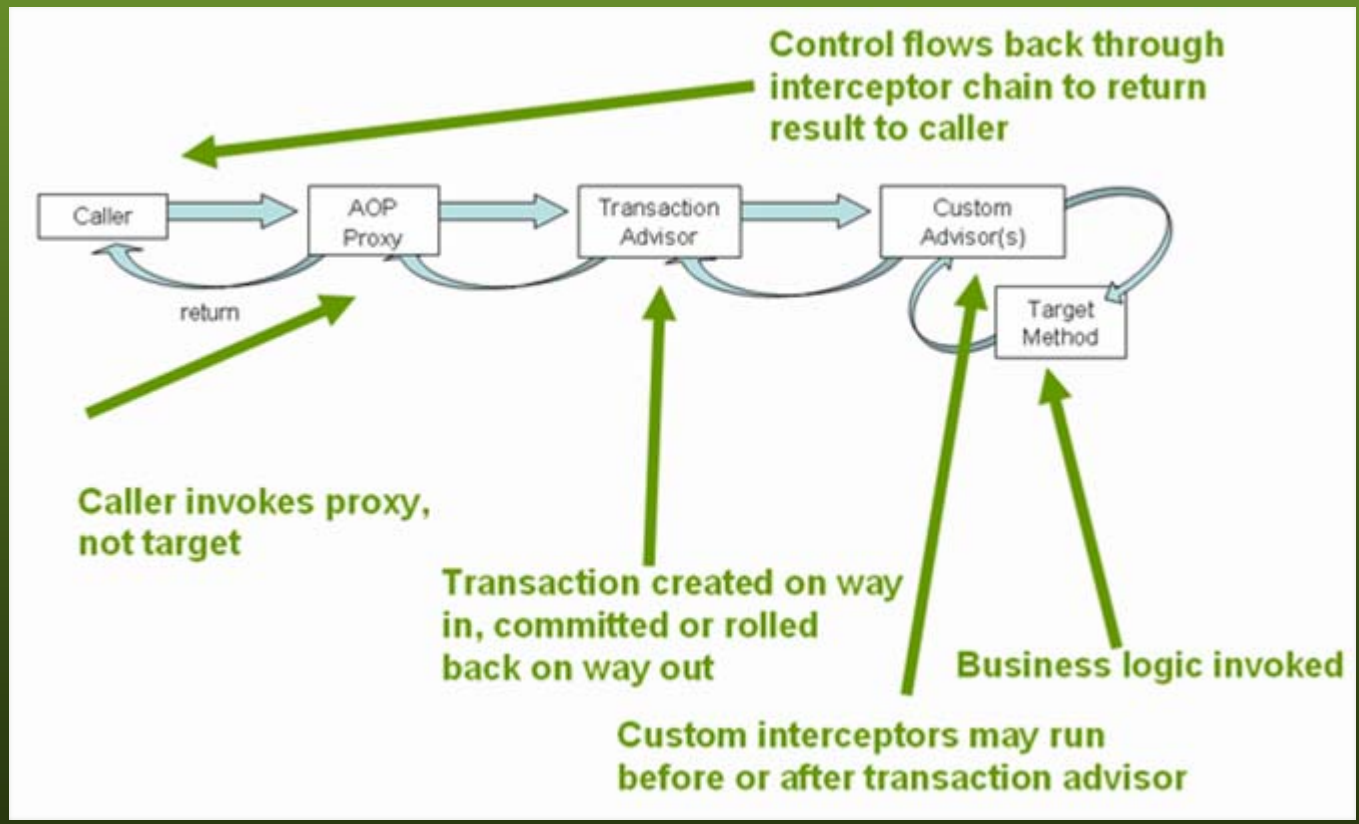
```
<bean id="exampleBean" class="examples.ExampleBean">

  <!-- constructor injection using the nested <ref/> element -->
  <constructor-arg><ref bean="anotherExampleBean"/></constructor-arg>

  <!-- constructor injection using the neater 'ref' attribute -->
  <constructor-arg ref="yetAnotherBean"/>

  <constructor-arg type="int" value="1"/>
</bean>

<bean id="anotherExampleBean" class="examples.AnotherBean"/>
<bean id="yetAnotherBean" class="examples.YetAnotherBean"/>
```

Services portables

- SPRING fournit une excellente solution d'accès aux données aussi bien pour JDBC que pour les principaux ORM
  - JPA, Hibernate, TopLink, JDO, OJB, iBatis
- SPRING intègre un nombre impressionnant de technologies
  - JMS, JMX, JCA, Remoting (RMI, Burlap, Hessian, Web services), Email, ordonnanceurs de tâches, ...
- Pour le WEB, SPRING propose un MVC particulièrement flexible, un autre pour les Portlets et intègre tous les principaux frameworks de présentation

AOP : La transaction (1)

- Par configuration un proxy va être intercalé entre la façade de service et ses clients



AOP : La transaction (2)

- La façade à rendre transactionnelle

```
<bean id="bizFacadeTarget" class="….MyBizFacade" lazy-init="true">
            <property name="dataSource" ref="dataSource"/>
</bean>
<bean id="dataSource"  . . . />
```

- Le gestionnaire de transaction

```
<!-- Transaction manager for a single JDBC DataSource -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
            <property name="dataSource" ref="dataSource"/>
</bean>
```

```
<!-- Transaction manager that delegates to JTA (for a transactional JNDI DataSource) -->
<bean id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionManager"/>
```

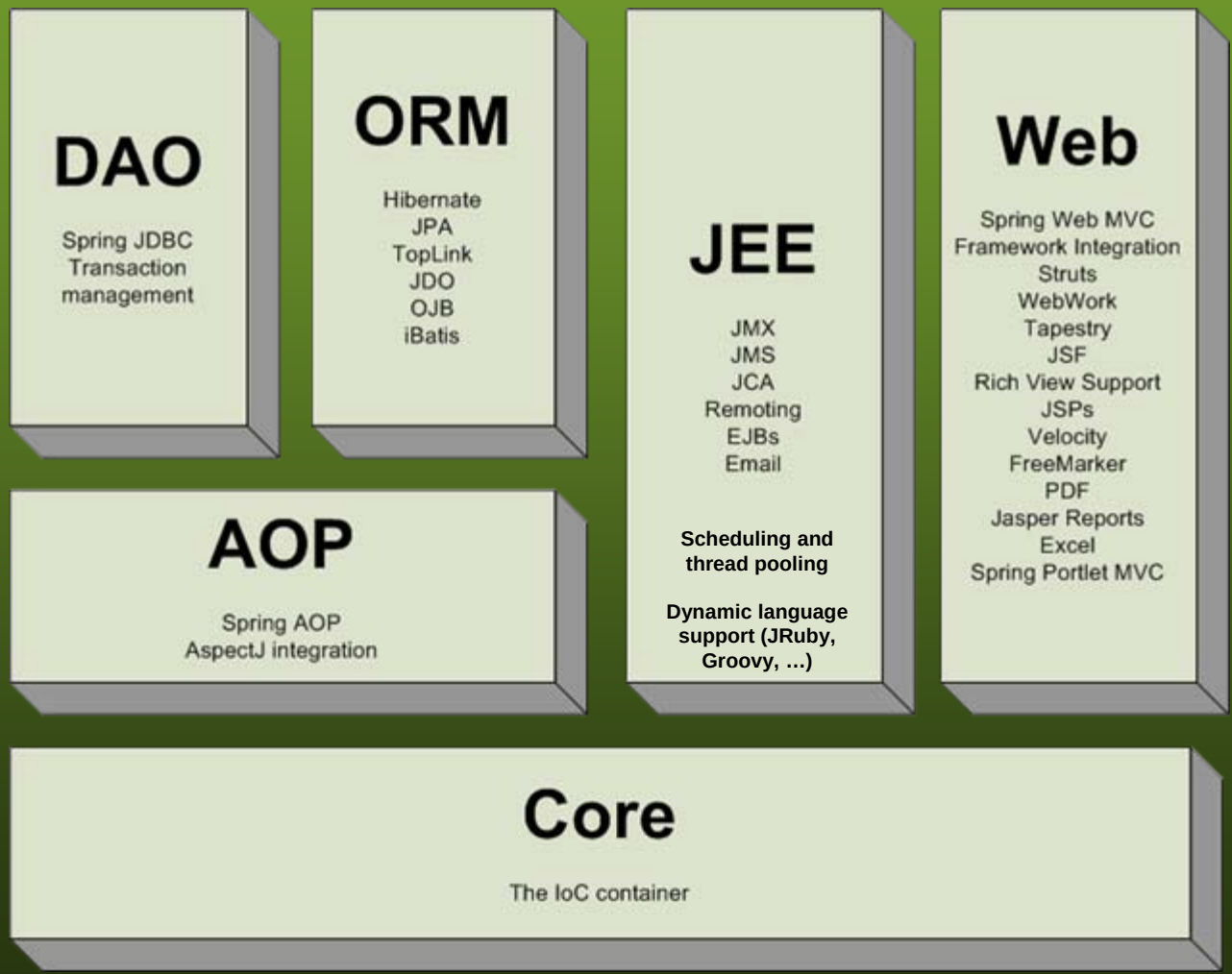
- Le proxy transactionnel AOP

```
<!-- Transactional proxy. -->
<bean id="bizFacade" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
            <property name="transactionManager" ref="transactionManager"/>
            <property name="target" ref="bizFacadeTarget"/>
            <property name="transactionAttributes">
                        <props>
                                    <prop key="find*">PROPAGATION_REQUIRED,readOnly</prop>
                                    <prop key="*">PROPAGATION_REQUIRED</prop>
                        </props>
            </property>
</bean>
```

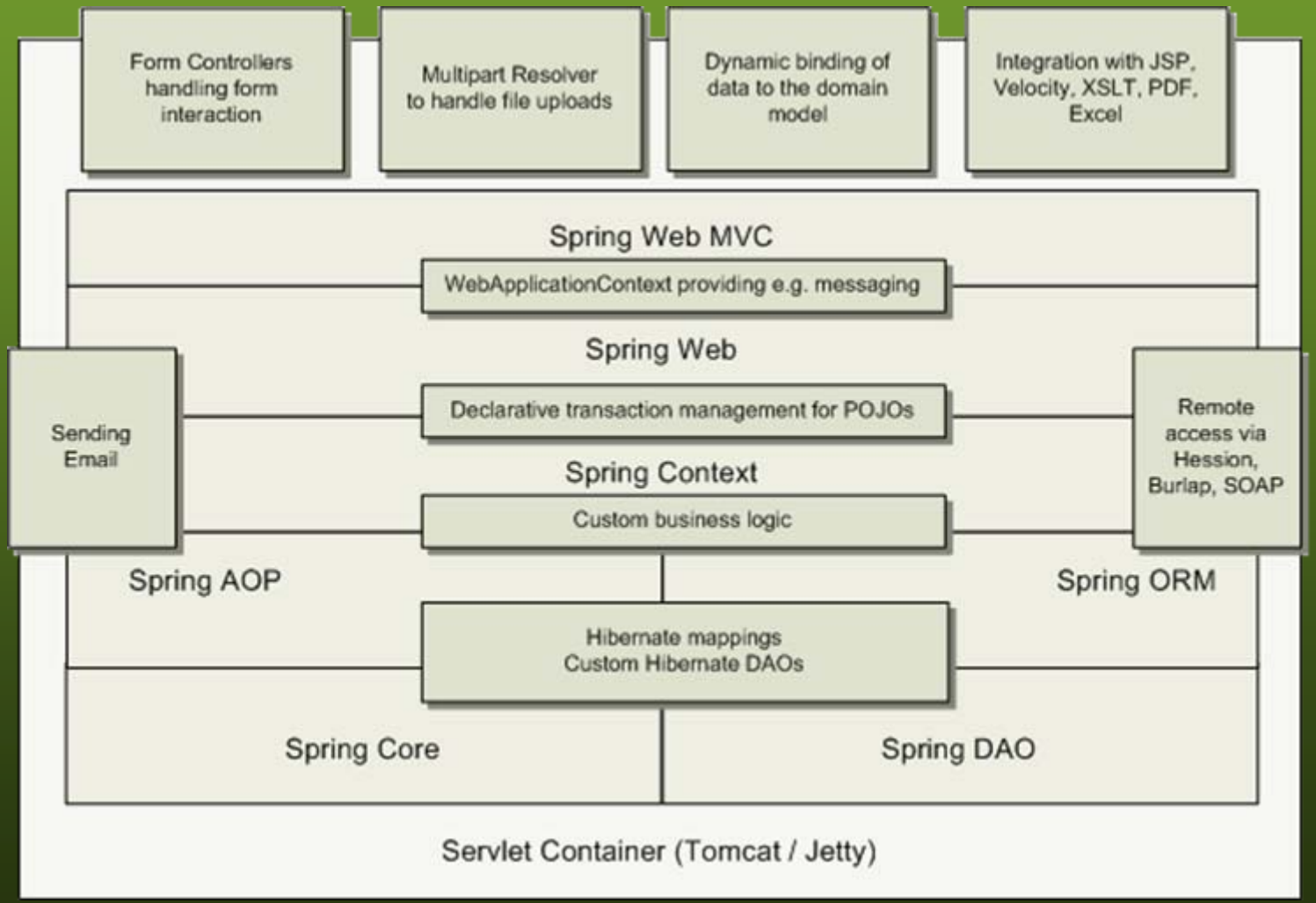
AOP : conclusion

- En résumé du cas d'utilisation de l'AOP :
  - Le code du service ne gère pas l'aspect « transaction »
  - Le code du proxy ne gère que son « aspect »
  - Le client appelle la méthode du proxy en croyant appeler celle du service
  - Plusieurs aspects peuvent être chaînés, en bout de course la méthode du service est appelée
  - La construction peut être déclarative ou codée
- SPRING fournit nombre de proxies de ce type
- SPRING fournit tous les éléments pour construire ses propres aspects Spring-AOP et AspectJ

Structure de Spring



Utilisation complète de Spring



Principales nouveautés v2.0

- Configuration plus concise et extensible
- Intégration de JPA
- Réception JMS asynchrone (JMS maintenant complet)
  - Message Driven POJOs
- SimpleJdbcTemplate pour Java5
- Portlet MVC
- MVC: nouvelle librairie de Tags
- Langages scripts (JRuby, Groovy, ...)

- Objectif v2.1
  - Compatibilité avec l'environnement OSGI

Compléments

- SPRING Security (ACEGI)
- SPRING Web Flow
- SPRING Bean Doc
- SPRING IDE For Eclipse
- SPRING Rich Client
- SPRING LDAP
- SPRING Modules
- JSF-SPRING
- Aurora
- AppFuse (https://appfuse.dev.java.net)
- SPRING Framework .Net

Ressources

- Site officiel
  - http://www.springframework.org
- Forum
  - http://forum.springframework.org
- Annuaire relatif à SPRING
  - http://www.springhub.com
- Livre en Français et de qualité: SPRING par la pratique
  - http://www.eyrolles.com/Accueil/Livre/9782212117103
  - http://www.amazon.fr/Spring-par-pratique-d%E9velopper-applications/dp/2212117108

Par où commencer ?

- SPRING MVC step-by-step
  - Dans la distribution et sur le site (Documentation/Tutorials/Step-by-Step)
- Exemples de la distribution
  - Petclinic
  - Showcases
  - Countries
  - Jpetstore
  - Imagedb
- Manuel de référence
  - Dans la distribution et sur le site (Documentation/Reference and API/2.0/PDF)
- JavaDoc
  - Dans la distribution et sur le site (Documentation/Reference and API/2.0/API)

Vos questions

Et merci pour votre attention