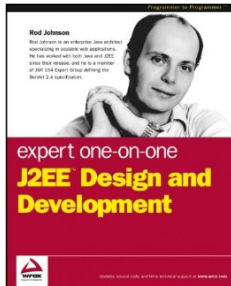


Formation Spring Framework

Introduction

Bref historique



2002

- Rod Johnson publie un livre "J2EE Design and Development" dans lequel il propose du code qui va devenir le Framework Spring



2004

- Sortie du projet open source Spring 1.0
- Rod Johnson publie un livre "J2EE Development without EJB" qui explique les limites des EJB et pourquoi utiliser Spring

2005

- Support des annotations avec Spring 2.5

2013













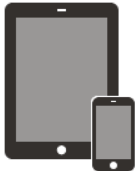





- Spring 4.0 qui inclut le support de Java 8, Groovy 2, WebSocket, ...

Spring

- Spring est construit comme un ensemble d'outils flexibles structurant et facilitant le développement d'applications basées sur la JVM.
- Les applications développées avec Spring peuvent aussi bien être des applications Web que des applications Desktop ou encore des briques systèmes permettant de traiter de grand volumes de données.
- Spring c'est plus de 20 projets open source.
 - Sources : <https://github.com/spring-projects/>
- Spring edite aussi un IDE : Spring Tool Suite (une distribution d'Eclipse avec des plugins Spring)
- **Spring est le framework Java N°1 en Entreprise.**



Vue d'ensemble

 Spring IO	 Spring Boot	 Spring Framework	 Spring XD	 Spring Cloud	 Spring Data
 Spring Integration	 Spring Batch	 Spring Security	 Spring HATEOAS	 Spring Social	 Spring AMQP
 Spring Mobile	 Spring Android	 Spring Web Flow	 Spring Web Services	 Spring LDAP	 Spring Session

Projets Spring (1)

- **Spring IO Platform**
 - Regroupe les différents projets au sein d'une plateforme versionnée.
- **Spring Framework**
 - Fournit les fonctionnalités cœurs de Spring tel que l'injection de dépendance, la gestion des transactions, l'accès aux données, le développement d'applications web, etc.
- **Spring Boot**
 - Fournit un ensemble de configuration par défaut d'un projet Spring pour permettre un développement rapide.
- **Spring XD**
 - Simplifie le développement des applications "Big Data" en offrant une plateforme unifiée et extensible.
- **Spring Cloud**
 - Fournit un ensemble d'outils adressant les problématiques récurrentes des systèmes distribués (gestion de configuration, déployer des microservices, ...).
- **Spring Data**
 - Propose une approche "moderne" d'accès aux données relationnelles, non-relationnelles, MapReduce, ...

Projets Spring (2)

- **Spring Integration**
 - Fournit des implémentations de plusieurs patterns d'intégration d'Entreprise (EndPoint, Channel, Aggregator, Filter, ...) et aussi une intégration aisée avec des systèmes externes (REST/HTTP, FTP/SFTP, JMS, RabbitMQ, ...)
- **Spring Batch**
 - Simplifie la mise en œuvre de batch applicatif.
- **Spring Security**
 - Offre la possibilité de protéger différentes parties d'une application avec un support des autorisations et des authentifications.
- **Spring HATEOAS**
 - Simplifie la création des représentations REST qui suivent le principe de HATEOAS.
- **Spring Social**
 - Simplifie les interactions d'une application avec les réseaux sociaux (Facebook, Twitter, LinkedIn, ...)
- **Spring AMQP**
 - Permet de développer des solutions de messageries basées sur AMQP

Projets Spring (3)

- **Spring Mobile**
 - Extension de Spring MVC pour détecter l'appareil connecté à l'application et fournir des vues adaptées.
- **Spring for Android**
 - Offre un client REST et un support de l'authentification pour des applications Android.
- **Spring Web Flow**
 - Offre un contrôle de la navigation Web via une configuration.
- **Spring Web Services**
 - Facilite le développement de services SOAP.
- **Spring LDAP**
 - Brique d'outil permettant de communiquer avec un LDAP.
- **Spring Session**
 - API/Implémentation pour la gestion des sessions utilisateurs

Projets Spring (4)

- Et bien d'autres :
 - Spring Roo
 - Spring Blazeds Integration
 - Spring Loaded
 - Spring Shell
 - REST Shell

Spring Framework

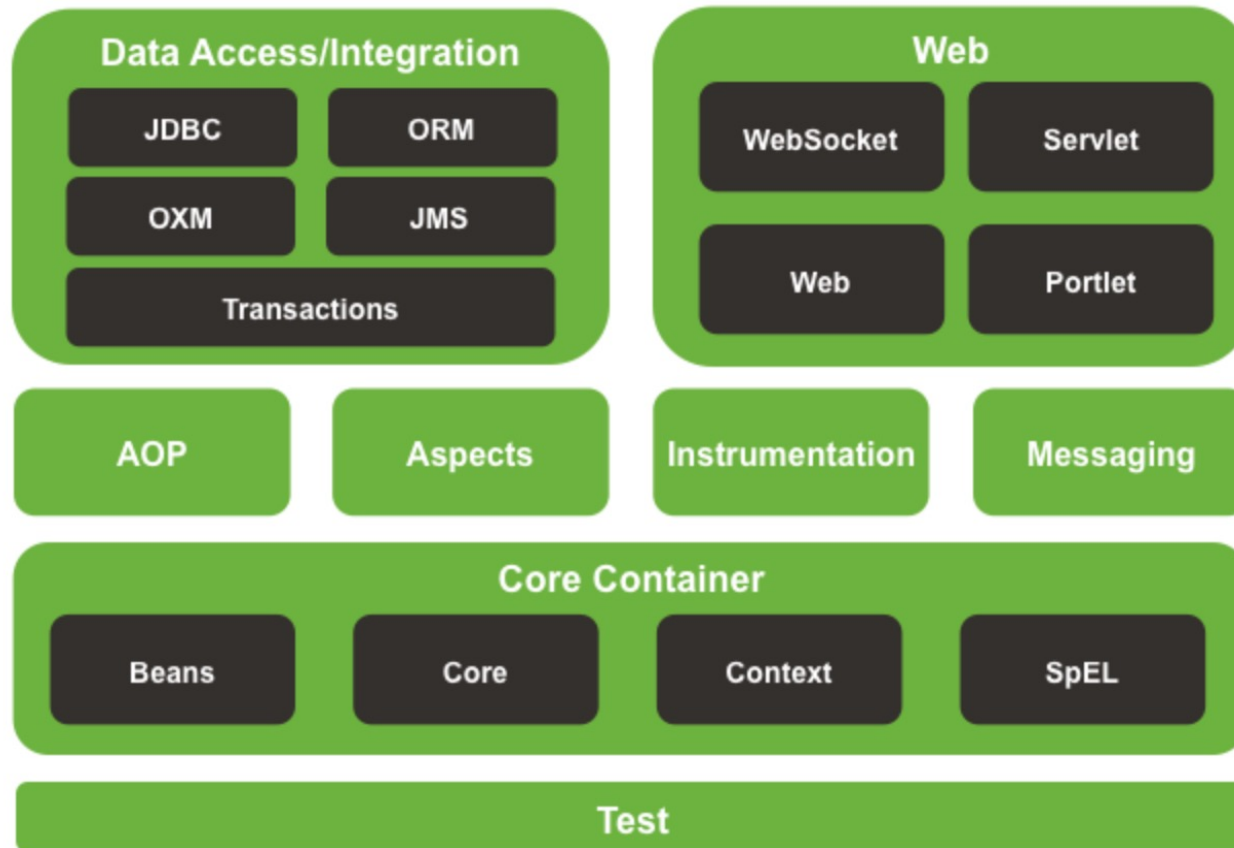
Spring Framework

- Spring Framework est un Framework offrant un cadre de développement d'applications Java. Il est composé de plusieurs modules permettant d'appliquer les bonnes pratiques de développement pour les besoins usuels.
- Spring Framework offre un mécanisme d'**injection de dépendance** qui permet :
 - d'améliorer la qualité du code
 - de coupler faiblement les composants
 - de faciliter l'écriture des tests
- Spring framework possède offre des similaires à la stack Java EE dans un conteneur dit "léger".

Modules Spring Framework



Spring Framework Runtime

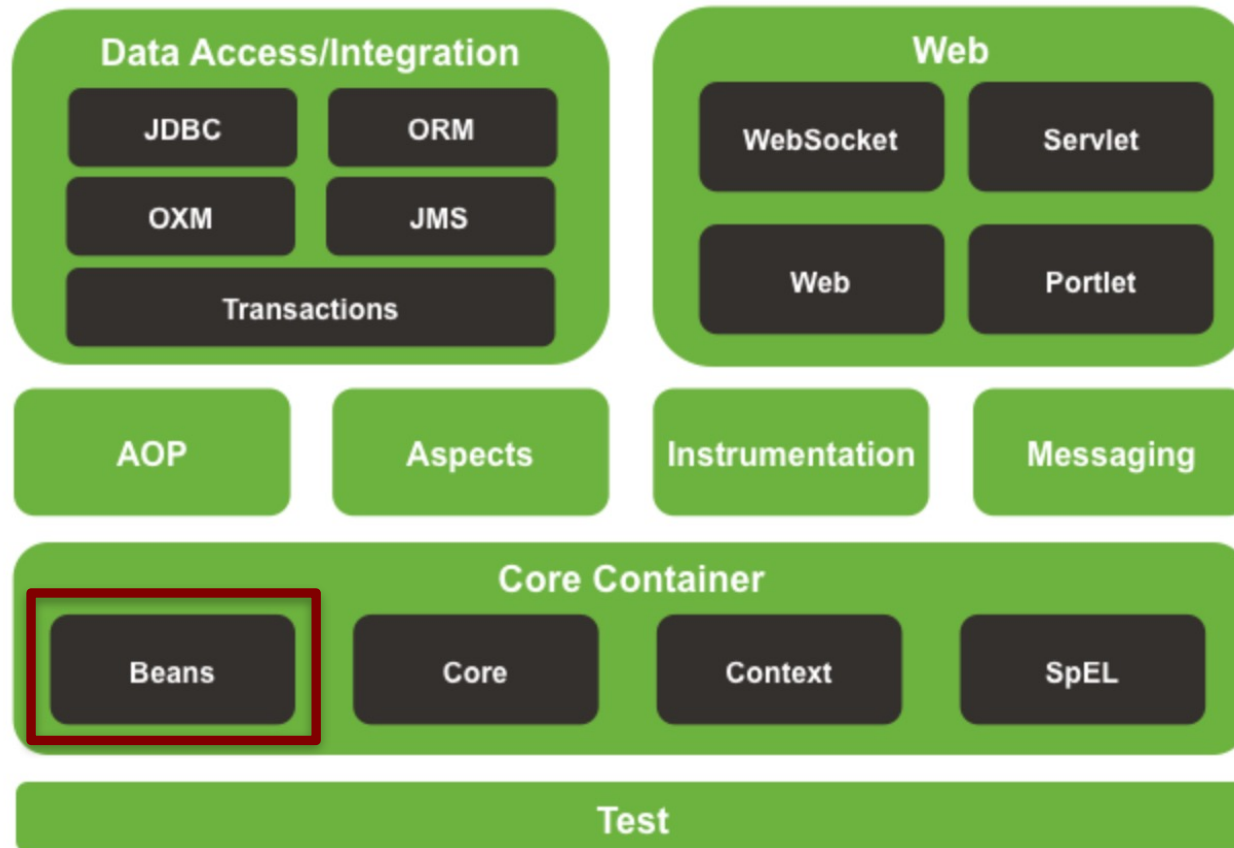


Beans

Beans



Spring Framework Runtime



Beans Spring

- Un Bean est un objet Java simple
 - POJO (Plain Old Java Object)
- Un Bean Spring est un objet Java simple dont le cycle de vie est géré par Spring
- Un Bean Spring bénéficie du mécanisme d'injection de dépendance.

```
public class PizzaService {  
  
    // du code  
  
    public List<Pizza> findAllPizzas() {  
        // du code  
    }  
  
    public Pizza findById(Long id) {  
        // du code  
    }  
  
    // du code  
}
```

Définition d'un bean

Déclaration XML

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/
schema/beans/spring-beans.xsd">

    <bean id="pizzaService" class="fr.pizzeria.spring.PizzaService"></bean>

</beans>
```

Equivalent Java

```
PizzaService service = new PizzaService();
```

Bean avec constructeur

Déclaration XML

```
<bean id="pizzaService" class="fr.pizzeria.spring.PizzaService">  
    <constructor-arg ref="pizzaRepository" />  
</bean>  
  
<bean id="pizzaRepository" class="fr.pizzeria.spring.PizzaRepository" />
```

Equivalent Java

```
PizzaRepository pr = new PizzaRepository();  
PizzaService service = new PizzaService(pr);
```


Bean avec propriétés

Déclaration XML

```
<bean id="pizzaService" class="fr.pizzeria.spring.PizzaService">  
  <property name="repository" ref="pizzaRepository" />  
  <property name="version" value="2.0"></property>  
</bean>  
  
<bean id="pizzaRepository" class="fr.pizzeria.spring.PizzaRepository" />
```

Equivalent Java

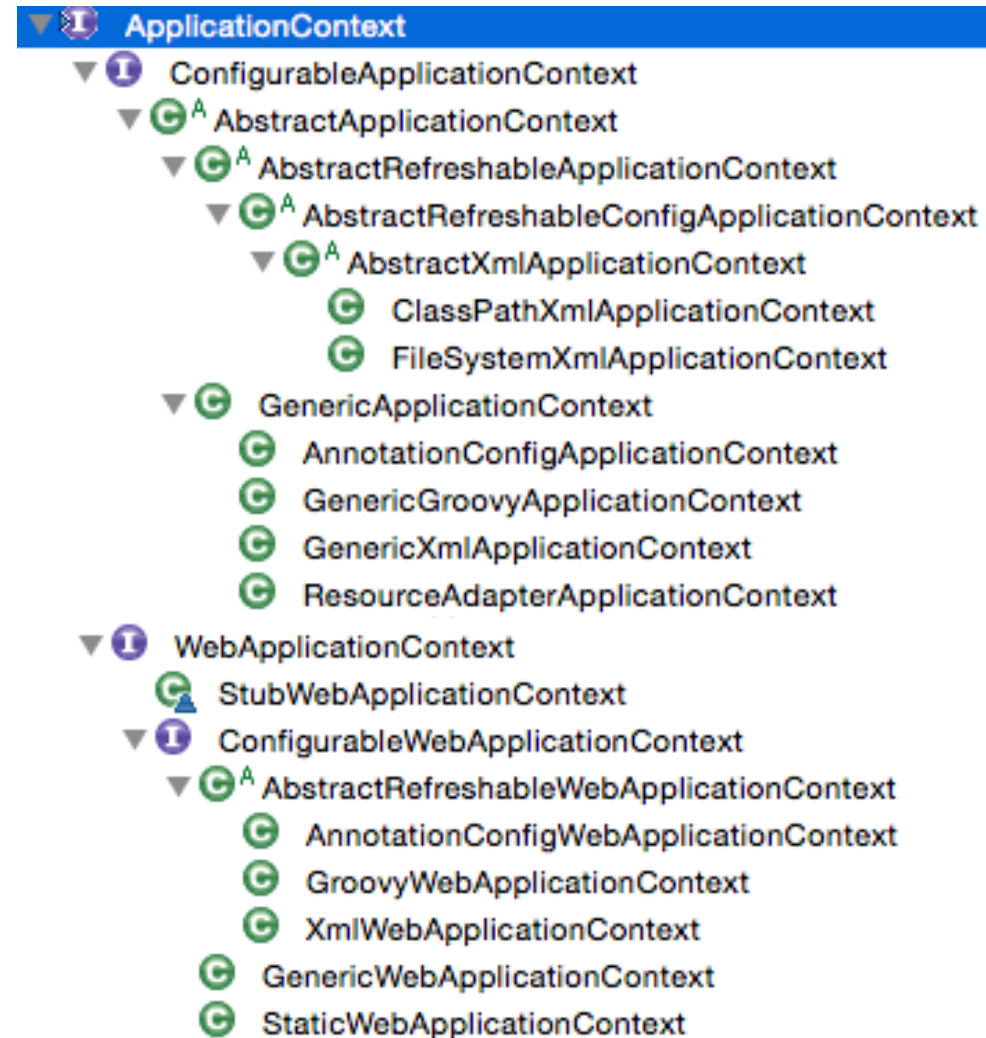
```
PizzaRepository pr = new PizzaRepository();  
PizzaService service = new PizzaService();  
service.setRepository(pr);  
service.setVersion("2.0");
```

Contexte Spring

- Un contexte Spring peut être initialisé depuis n'importe quel environnement :
 - Tests unitaires Junit
 - Application Web
 - EJB
 - Application autonome (Console, Desktop, Batch, ...)
- Un contexte est initialisé à partir d'un fichier de configuration.
- Un contexte est une implémentation de l'interface `org.springframework.context.ApplicationContext`.

Interface ApplicationContext

- Spring met à disposition une interface ApplicationContext
- De nombreuses implementations :
 - **ClassPathXmlApplicationContext**
 - Chargement du contexte depuis une source XML présente dans le classpath
 - **FileSystemXmlApplicationContext**
 - Chargement du contexte depuis une source XML depuis le système de fichier
 - **XmlWebApplicationContext**
 - Chargement du contexte depuis une source XML dans le contexte d'une application web
 - **AnnotationConfigApplicationContext**
 - Chargement du contexte à partir de classes annotées
 - **AnnotationConfigWebApplicationContext**
 - Chargement de contexte à partir de classes annotées au sein d'une application Web.
 - ...



Exemple de Création de bean

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="pizzaService" class="fr.pizzeria.spring.PizzaService"></bean>
</beans>
```

PizzaService service = new PizzaService();

src/application.xml

```
public class PizzaService {
    public String getVersion() {
        return "1.0";
    }
}
```

src/fr.pizzeria.spring.PizzaService.java

```
public class BeanSpringTest {
    @Test
    public void test_creation_bean_spring() {
        try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("application.xml")) {
            PizzaService pizzaService = context.getBean(PizzaService.class);
            assertNotNull(pizzaService);
            assertEquals("1.0", pizzaService.getVersion());
        }
    }
}
```

test/fr.pizzeria.spring.BeanSpringTest.java

Portée d'un Bean

```
<bean id="pizzaService" class="fr.pizzeria.spring.PizzaService" scope="prototype" ></bean>
```

- Spring définit pour chaque bean sa portée
- Les portées disponibles :
 - **singleton**
 - Une seule instance par contexte. C'est la portée par défaut
 - **prototype**
 - Une nouvelle instance est créée chaque fois que le bean est injecté
 - **session**
 - Une nouvelle instance créée par session utilisateur (environnement Web)
 - **request**
 - Une nouvelle instance créée pour chaque requête (environnement Web)
 - **application**
 - Une nouvelle instance par ServletContext (environnement Web)
 - **custom**
 - Scope personnalisé.

Héritage

```
<bean id="parentBean" class="exemple.ParentBean" abstract="true">
  <property name="firstname" value="value1"/>
</bean>

<bean id="childBean" class="exemple.ChildBean" parent="parentBean">
  <property name="lastname" value="value2"/>
</bean>
```

- Bean parent caractérisé par l'attribut « **abstract** ».
- Si la valeur « abstract » est à « true » alors le bean ne peut être utilisé directement.
- Bean enfant caractérisé par l'attribut « **parent** » pointant vers le bean parent.
- Le bean enfant doit contenir les constructeurs et les propriétés du bean parent (sans lien d'héritage Java entre ces beans)

```
public class ParentBean {

    private String firstName;

}
```

```
public class ChildBean {

    private String firstName;
    private String lastName;

}
```

Héritage

```
<bean id="parentBean" abstract="true">
  <property name="firstname" value="value1"/>
</bean>

<bean id="childBean" class="exemple.ChildBean" parent="parentBean">
  <property name="lastname" value="value2"/>
</bean>
```

- Un bean parent peut ne pas avoir de classe concrète
 - Le bean parent est utilisé comme pure template
- Le bean enfant peut surcharger les attributs Spring du bean parent
- Attention Héritage Java <> Héritage Spring
 - Spring → Héritage de configuration de bean
 - Java → Héritage de classe

p-namespace

- "p-namespace" permet de déclarer les attributs d'un bean sans utiliser le tag "property"

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://
www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://
www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="pizzaSpeciale" class="fr.pizzeria.spring.Pizza" p:pizzaName="nantes" p:id="12"></bean>

</beans>
```


Externalisation de la configuration

src/application.xml

```
<beans>
  <bean id="dataSource" class="com.oracle.jdbc.pool.OracleDataSource">
    <property name="URL" value="${datasource.url}"/>
    <property name="user" value="${datasource.user}"/>
  </bean>
  <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="location" value="/WEB-INF/config/configuration.properties"/>
  </bean>
</beans>
```

/WEB-INF/config/configuration.properties

```
datasource.url=jdbc:oracle:thin@localhost:1521:BANK
datasource.user=monutilisateur
```

c-namespace

- "c-namespace" permet de configurer l'injection par constructeur

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:c="http://
www.springframework.org/schema/c"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://
www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="pizzaSpeciale" class="fr.pizzeria.spring.Pizza" c:pizzaName="nantes" c:id="12"></
bean>

</beans>
```

util-namespace

- "util-namespace" permet d'accès à des configurations utilitaires :
 - Chargement des fichiers
 - Déclarer une liste
 - ...

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:util="http://
www.springframework.org/schema/util"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://
www.springframework.org/schema/beans/spring-beans.xsd">

    <util:properties id="jdbcConfiguration" location="classpath:fr/nantes/app.properties" />

    <util:list id="emails">
        <value>t@gmail.com</value>
        <value>h@yahoo.fr</value>
    </util:list>
</beans>
```




j2ee-namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:jee="http://
www.springframework.org/schema/jee"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://
www.springframework.org/schema/beans/spring-beans.xsd">

    <jee:jndi-lookup id="dataSource" jndi-name="jdbc/PizzaDataSource"/>

    <bean id="pizzaDao" class="fr.PizzaDao">
        <property name="dataSource" ref="dataSource"/>
    </bean>
</beans>
```

Et bien d'autres

<input type="checkbox"/>		aop - http://www.springframework.org/schema/aop
<input checked="" type="checkbox"/>		beans - http://www.springframework.org/schema/beans
<input type="checkbox"/>		c - http://www.springframework.org/schema/c
<input type="checkbox"/>		cache - http://www.springframework.org/schema/cache
<input type="checkbox"/>		context - http://www.springframework.org/schema/context
<input type="checkbox"/>		jdbc - http://www.springframework.org/schema/jdbc
<input checked="" type="checkbox"/>		jee - http://www.springframework.org/schema/jee
<input type="checkbox"/>		jpa - http://www.springframework.org/schema/data/jpa
<input type="checkbox"/>		lang - http://www.springframework.org/schema/lang
<input type="checkbox"/>		mvc - http://www.springframework.org/schema/mvc
<input checked="" type="checkbox"/>		p - http://www.springframework.org/schema/p
<input type="checkbox"/>		repository - http://www.springframework.org/schema/data/repository
<input type="checkbox"/>		task - http://www.springframework.org/schema/task
<input type="checkbox"/>		tx - http://www.springframework.org/schema/tx
<input type="checkbox"/>		util - http://www.springframework.org/schema/util