

# SPRING FRAMEWORK



# PLAN DU COURS

- Introduction au Framework SPRING CORE
- IoC : Inversion de Control (Injection de Dépendances)
- AOP : Aspect Oriented Programmation
- Abstraction des Services

# INTRODUCTION AU FRAMEWORK SPRING



# INTRODUCTION

- C'est quoi un **Framework** :
  - C'est un **cadre de développement**
  - Contient des «bonnes pratiques»
  - Permet d'éviter de recoder des classes utilitaires
  - Permet de se focaliser sur le métier
- Spring fournit la «plomberie» : le socle technique
- Les développeurs peuvent se concentrer sur le code métier (le vrai travail)

# INTRODUCTION

- Le Framework Spring est un socle pour le développement d'applications.
- Il fournit de nombreuses fonctionnalités.
- C'est l'un des Frameworks les plus répandus dans le monde Java.
- Framework Open Source
- **Framework Spring (Spring Core)** : <http://projects.spring.io/spring-framework>
- **Plate-forme Spring** : <http://www.spring.io>

# INTRODUCTION

- Rapport du paysage des Outils Java et des Technologies utilisées en 2016 :

<https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016>

- Voici, dans le slide suivant, les 12 outils et Frameworks Java les plus utilisés en 2016 :
- Spring y figure deux fois :

# 12 OUTILS ET FRAMEWORKS JAVA LES PLUS UTILISÉS EN 2016

## maven

Over **two in three (68%)** devs use Maven as their main build tool



Over **two in three (68%)** devs use Git as their version control



Almost **two in three (62%)** devs use Java 8 in production



**Three in five (60%)** devs use Jenkins for CI



## IntelliJ IDEA

Almost **one in two (46%)** devs use IntelliJ, the most popular IDE in the survey.



Over **two in five (43%)** devs use Spring MVC



Over **two in five (42%)** devs use Tomcat server in production



Almost **two in five (39%)** devs use Oracle DB in production



## Microservices

Over **one in three (34%)** devs have adopted a microservices architecture



Almost **one in three (32%)** devs use Docker in production



Over **three in ten (31%)** devs use Java EE 7



Almost **three in ten (29%)** devs use Spring Boot



Based on the 2016 Developer Productivity Report by RebelLabs

# BUT ET FONCTIONNALITÉS DE SPRING

- **Spring Boot** : Framework Spring qui vous permet d'embarquer un serveur Tomcat dans votre livrable, et simplifier la livraison et le test de votre application.



- **Spring MVC** : C'est un Framework concurrent de Struts. Il permet de manière flexible de créer des interfaces web évoluées, tout en bénéficiant des fonctionnalités de Spring Core. (les voici : )





# BUT ET FONCTIONNALITÉS DE SPRING

- Le but de Spring est de faciliter et de rendre productif le développement d'applications.
- Il fournit beaucoup de fonctionnalités :
  - **Injection de Dépendances** (IoC : Inversion de Control).
  - **AOP** : Aspect Oriented Programming : Injection de Code en RunTime.
  - **Abstraction** des Services : Permet l'accès à des Services sans entrer le détails de leur fonctionnement (DAO, ORM, MVC, Bean, ...).
  - **Testabilité** de l'application facilitée
  - **Web** : Permet le développement d'interfaces web évoluées.
  - **Intégration** : Spring offre un ESB, qui permet l'intégration et la communication entre les applications (EAI).

# BUT ET FONCTIONNALITÉS DE SPRING

- Une application typique utilisant Spring est généralement structurée en trois couches :
  - Couche **Présentation** : interface homme machine
  - Couche **Service** : interface métier avec mise en œuvre de certaines fonctionnalités.
  - Couche **Accès aux Données** : recherche et persistance des objets.
- Spring est utilisé pour créer et injecter les objets requis pour communiquer entre les différentes couches.

# HISTORIQUE DE SPRING

- **2002** : Rod Johnson publie son livre «Expert One-on-One J2EE Design and Development», dans lequel il propose du code, qui va devenir plus tard le Framework Spring
- **2004** : Spring 1.0, licence Apache 2.0
- **2005** : Spring devient populaire, en particulier en réaction par rapport aux EJBs 2.x
- **2006** : Spring 2.0
- **2007** : Spring 2.5, avec support des annotations
- **2009** : Spring 3.0
- **2013** : Spring 4.0
- **2017** : Spring 4.3.7, Version courante
- **2017** : Spring 5.0 en cours de développement

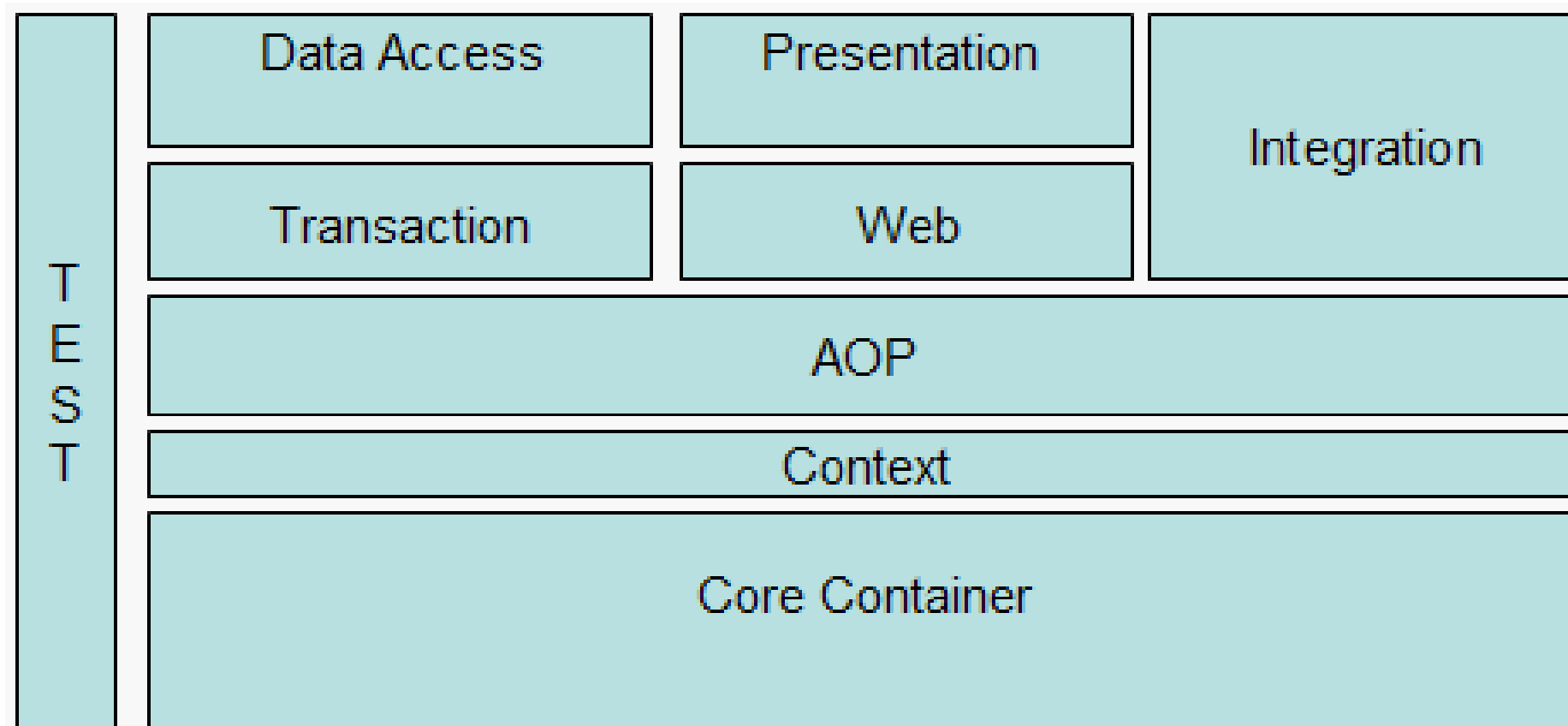
# SPRING CORE

- C'est un Framework Java stable et très populaire.
- Le but premier de Spring est de réduire les dépendances. C'est en partie pour cette raison que ce Framework a été rapidement adopté.
- Il prend en charge la création d'objets et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets.
- Il est considéré comme une alternative aux EJB.
- C'est une solution pour simplifier les parties de développement lourde (JDBC, Web Services, ...).

# SPRING CORE

- Spring Framework est composé de :
  - **Spring Core Container** : modules de base
  - **AOP** : permet de mettre en œuvre l'AOP
  - **Data Acces/Integration** : modules d'accès aux données
  - **Web** : modules pour le développement d'applications web
  - **Test** : fonctionnalités pour les tests automatisés avec Spring

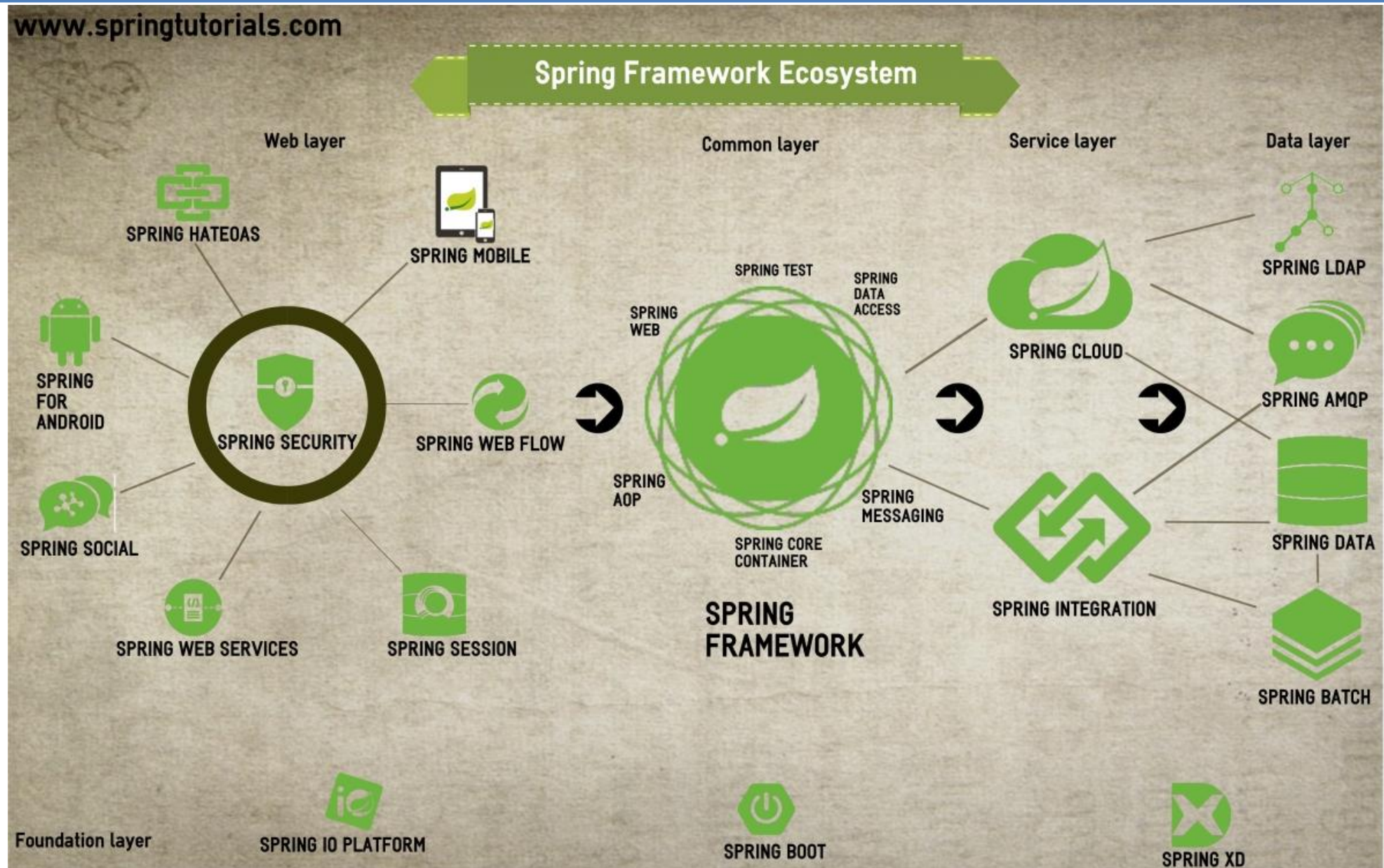
# SPRING CORE



# LES PROJETS SPRING

- <https://spring.io/projects> :
- **Spring Framework** : contient les fonctionnalités de base de Spring
- **Spring Web Services** : permet de développer des services web de type SOAP
- **Spring Security** : permet de gérer l'authentification et les habilitations d'une application web
- **Spring Batch** : permet le développement des applications de type batch qui peuvent gérer de gros volumes de données
- **Spring Integration** : il s'agit d'un ESB (Enterprise Service Bus) pour interconnecter les application d'une entreprise
- **Spring Android** : a pour but de faciliter le développement d'applications Android
- ...

# LES PROJETS SPRING





# TP1 : INSTALLATION **STS** (SPRING TOOL SUITE)



# TP1 : INSTALLATION **STS** (SPRING TOOL SUITE)

- Prérequis - Installation de **Java 8**
- Voir Document : JAVA 8 (PPT)



# TP1 : INSTALLATION **STS** (SPRING TOOL SUITE)

- Téléchargement STS 3.8.4 : <http://spring.io/tools/sts/all>

The screenshot shows the 'Spring Tool Suite™ Downloads' page. At the top, it says 'TOOLS' and 'Spring Tool Suite™ Downloads'. Below this, it instructs users to use links to download an all-in-one distribution for their platform or check previous versions. The main content area features three platform-specific download cards: Windows, Mac, and Linux. The Windows card is expanded, showing a dropdown menu with options for 'WIN, 32BIT' and 'WIN, 64BIT', each with a 'zip' download link and a size of '381MB'. The Mac and Linux cards also show 'Based on Eclipse 4.6.3' and a menu icon. On the left, a large dark card displays the Spring Tool Suite logo and the text 'STS 3.8.4.RELEASE' with a 'New & Noteworthy' link.

TOOLS

## Spring Tool Suite™ Downloads

Use one of the links below to download an all-in-one distribution for your platform.  
Or check the list of [previous Spring Tool Suite™ versions](#).

**STS 3.8.4.RELEASE**  
[New & Noteworthy](#)

**Windows**

Based on Eclipse 4.6.3

WIN, 32BIT	WIN, 64BIT
<a href="#">zip</a> 381MB	<a href="#">zip</a> 381MB

**Mac**

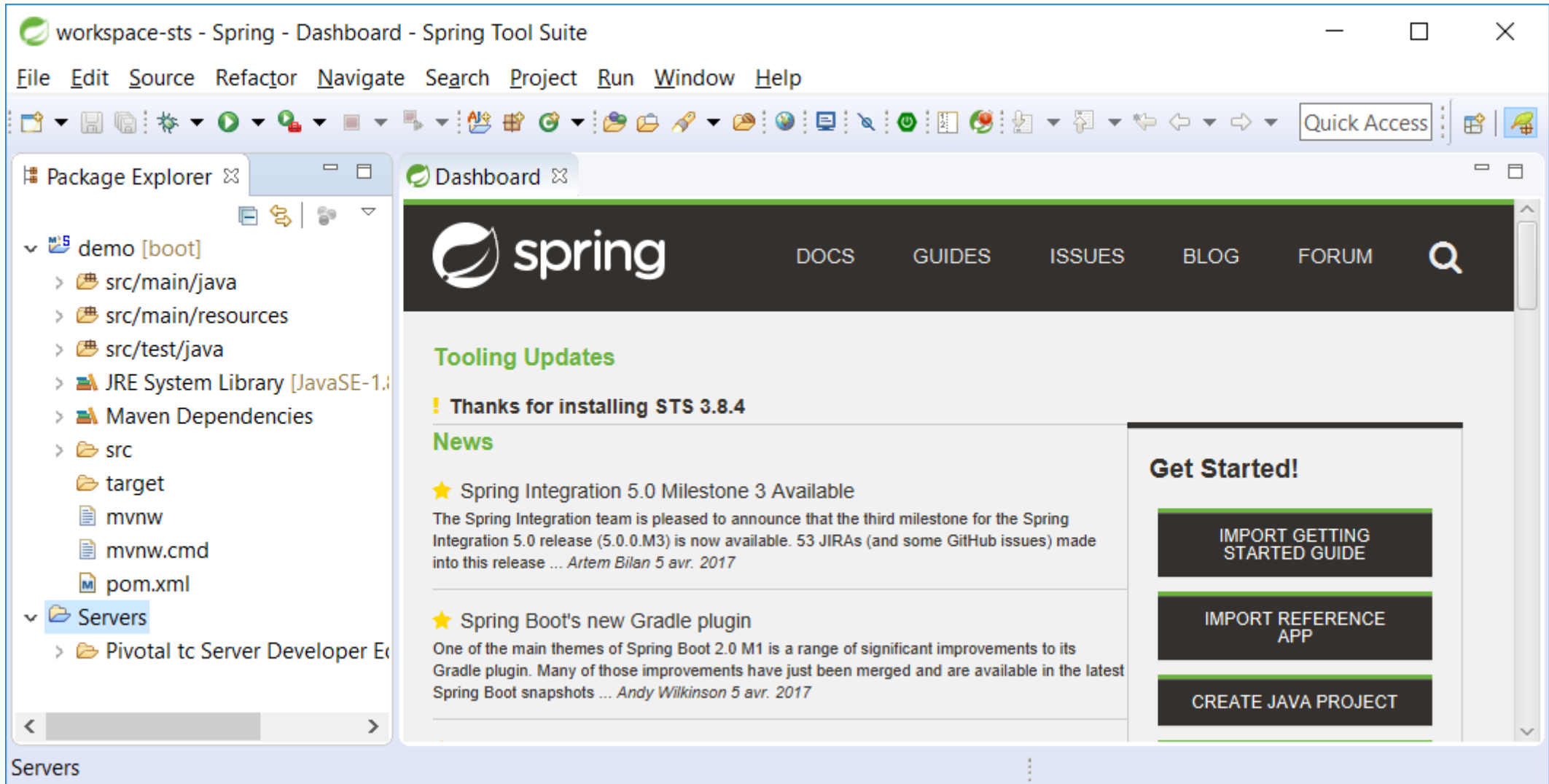
Based on Eclipse 4.6.3

**Linux**

Based on Eclipse 4.6.3

# TP1 : INSTALLATION STS (SPRING TOOL SUITE)

- Installation & Lancement : Dézippez le fichier téléchargé, et Lancer STS.exe



# FONCTIONNALITÉS SPRING : IOC



# IOC & INJECTION DE DÉPENDANCES

- L'IOC (**Inversion de Control**) est un design pattern. Il fonctionne selon le principe que l'exécution de l'application n'est plus sous le contrôle direct de l'application elle-même mais du Framework sous-jacent.
- Dans notre cas, nos applications web seront contrôlés par le Framework Spring. Comment? :
- L'**Injection de Dépendance** est la forme principale de l'loC. Elle implémente le principe de l'loC.

# IOC & INJECTION DE DÉPENDANCES

- L'**Injection de Dépendance** est une méthode pour créer les dépendances nécessaires d'un objet donné, sans avoir besoin de coder cela nous même.
- Elle permet un couplage faible entre les classes.
- Ceci permet de réduire le code relatif aux dépendances. Ce qui permet une lisibilité du code, et permet de gagner en temps de développement.

# INJECTION DE DÉPENDANCES

- Injection de dépendance par instanciation **statique** (new):
- Injection de dépendances entre la couche métier, et la couche accès aux données :

```
ScrumTeam scrumTeam = new ScrumTeam();
```

```
TeamMember developer = new TeamMember();
```

```
TeamMember productOwner = new TeamMember();
```

```
TeamMember scrumMaster = new TeamMember();
```

```
scrumTeam.setDeveloper(developer);
```

```
scrumTeam.setProductOwner(productOwner);
```

```
scrumTeam.setScrumMaster(scrumMaster);
```



# INJECTION DE DÉPENDANCES

- Injection de dépendance par instanciation **dynamique** :
- Pas besoin de recompiler, de re-packager.

```
Scanner scanner = new Scanner(new File("config.txt"));
String memberClassName = scanner.nextLine();
Class cmember = Class.forName(memberClassName);
ITeamMember member = (ITeamMember) cmember.newInstance();
```

# INJECTION DE DÉPENDANCES

- Injection de dépendance par instanciation **dynamique**, en utilisant **Spring** :



# C'EST QUOI UN BEAN?

- C'est un composant Java spécifique avec un identifiant unique.
- Plusieurs types de Bean (scope):
  - **singleton** (Une seule instance du Bean créée, et référencée à chaque invocation).
  - **prototype** (Nouvelle instance créée à chaque appel du Bean = `New ClassName()`)
  - ...

# COMMENT DÉFINIR UN BEAN?

- On définit un Bean dans un fichier XML pour que Spring puisse le gérer :
- Les attributs :
  - **class** : fully qualified java class name
  - **id** : the unique identifier for this bean
  - **configuration** : (scope, init-method, etc.)
  - **property** : arguments to pass to the bean setters at creation time

# EXEMPLE DE DÉFINITION D'UN BEAN

- Dans applicationContext.xml :

```
<bean id="memberService"
      class="com.m2i.formation.service.MemberService"
      scope="prototype">
  <property name="role" value="developper" />
</bean>
```

- Et dans le code java :

```
package com.m2i.formation.service;

public class MemberService {
    private String role;
    public void setRole(String role) {
        this.role = role;
    }
}
```

# FICHIERS DE CONFIGURATION DES BEAN

- Trois manières pour charger les fichiers de configurations des Beans :
  - A partir de web.xml :

```
<init-param>  
  <param-name>contextConfigLocation</param-name>  
  <param-value>/WEB-INF/spring/applicationContext.xml</param-value>  
</init-param>
```

- En utilisant le tag d'Import :

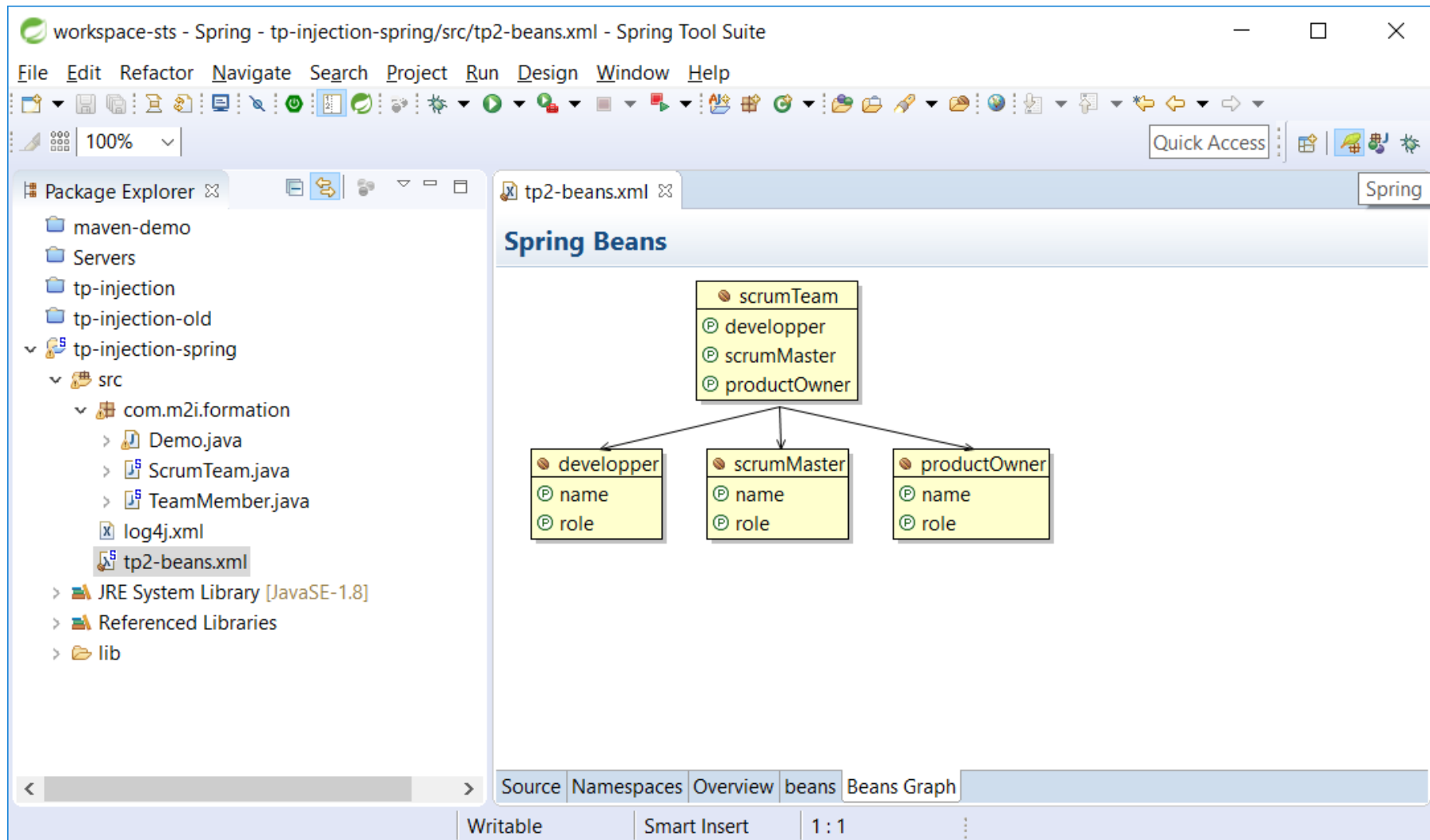
```
<import resource="classpath:spring-config.xml" />
```

- En utilisant le Contexte de l'Application :

```
ApplicationContext context = new  
ClassPathXmlApplicationContext("applicationContext.xml");
```

# TP2 : INJECTION DE DÉPENDANCES

- Les étapes seront décrites en cours ensemble.



# TP3 : INJECTION DE DÉPENDANCES AVEC MAVEN

- Refaire le même Projet (TP2) mais en utilisant Maven.
- Pour cela on commencera par une présentation de Maven.



# MAVEN























- Voir le document document MAVEN (PPT)



# TP3 : INJECTION DE DÉPENDANCES AVEC MAVEN

- Maven étant vu : Refaire le même Projet (TP2), en utilisant Maven et la déclaration des beans dans un fichier de configuration XML :
- Créer un projet de type **Maven** : tp3-injection-maven
- Ajouter **spring** : ajouter dépendance dans **pom.xml**
- mettre **log4j.xml** dans \src\main\resources et dans src\test\resources
- Créer le **package** : com.m2i.formation.service dans \src\main\java
- Créer les **bean** TeamMemberService et ScrumTeamService.java
- Créer le fichier de configuration **Spring** : **applicationContext.xml** dans \src\main\resources
- Ajouter les bean scrumTeam, developper, productOwner et scrumMaster dans applicationContext.xml (les trois derniers beans sont de type TeamMember).
- Créer la classe Demo.java, qui appelle le Service ScrumTeamService

# TP3 : INJECTION DE DÉPENDANCES AVEC MAVEN

- ▼  tp-injection-spring-maven
  - ▼  src/main/java
    - ▼  com.m2i.formation
      - >  Demo.java
      - >  ScrumTeam.java
      - >  TeamMember.java
  - ▼  src/main/resources
    -  log4j.xml
    -  tp2-beans.xml
  -  src/test/java
  -  src/test/resources
  - >  JRE System Library [J2SE-1.5]
  - ▼  Maven Dependencies
    - >  spring-context-4.3.7.RELEASE.jar - C:\Users\Mourad HASSINI\.m2\
    - >  spring-aop-4.3.7.RELEASE.jar - C:\Users\Mourad HASSINI\.m2\rep
    - >  spring-beans-4.3.7.RELEASE.jar - C:\Users\Mourad HASSINI\.m2\re
    - >  spring-core-4.3.7.RELEASE.jar - C:\Users\Mourad HASSINI\.m2\rep
    - >  commons-logging-1.2.jar - C:\Users\Mourad HASSINI\.m2\reposit
    - >  spring-expression-4.3.7.RELEASE.jar - C:\Users\Mourad HASSINI\.r
  - >  src
  -  target
  -  pom.xml

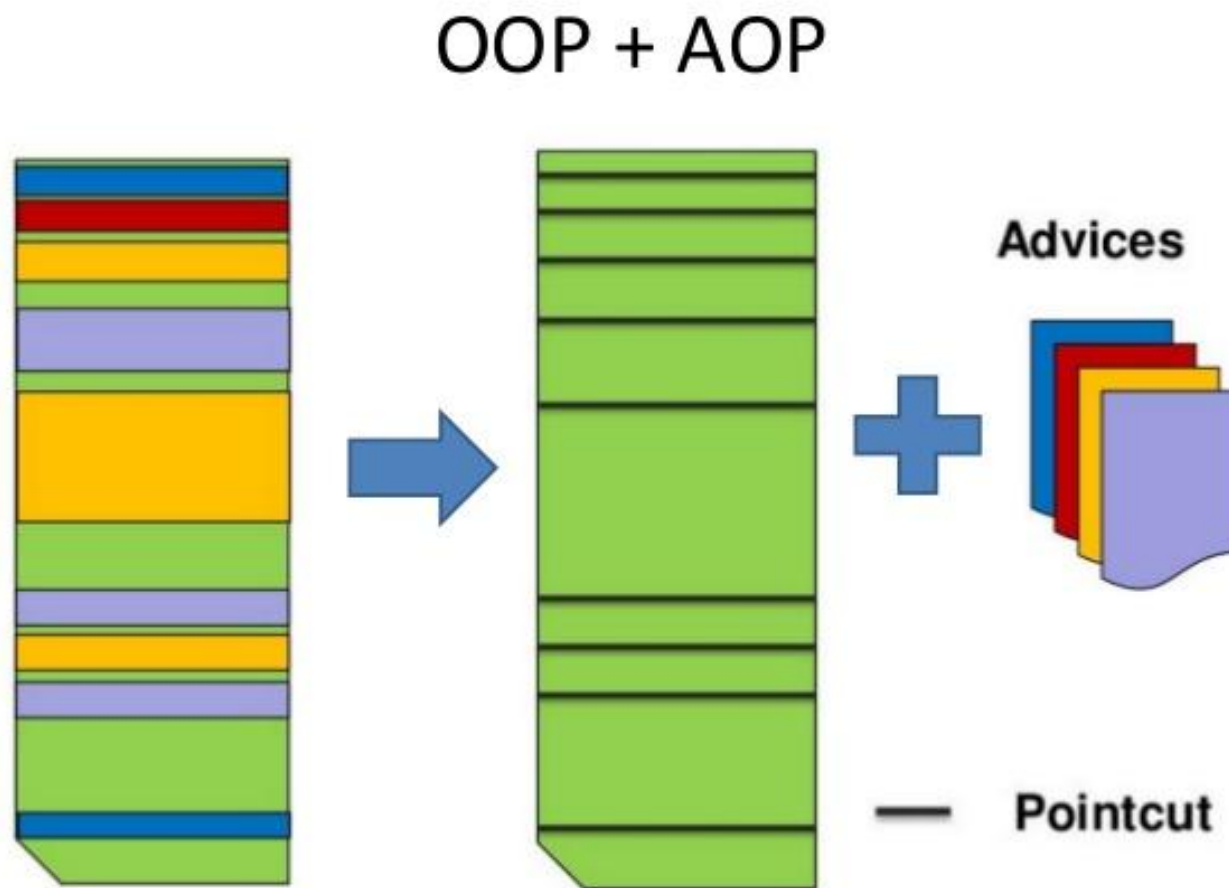
# FONCTIONNALITÉ SPRING : AOP



# AOP : ASPECT ORIENTED PROGRAMMATION

- **AOP** : Aspect Oriented Programming, ou Programmation Orientée Aspect en Français
- L'un des deux concepts principaux de Spring (avec l'Inversion de Contrôle)
- Permet de rajouter des comportements à des classes ou des méthodes existantes
  - Ajouter des traces (logs)
  - Ajouter la gestion des transactions
  - Ajouter du monitoring, ...
- Il s'agit de problématiques transverses, en général, techniques.

# AOP – OOP



# AOP : ASPECT ORIENTED PROGRAMMATION

- Separation of Concerns (**SoC**) : Au lieu d'avoir un appel direct à un module technique depuis un module métier, en AOP, le code du module en cours de développement est concentré sur le but poursuivi (la logique métier). Exemple : Ajout de logs dans une application existante.
- Don't Repeat Yourself (**DRY**) : Cela évite la duplication de code.

# AOP : LES CONCEPTS

- **Join point** : l'endroit où l'on veut qu'un aspect s'applique. Avec Spring AOP, il s'agit toujours d'une méthode
- **Pointcut** : une expression, utilisant la syntaxe AspectJ, qui permet de sélectionner plusieurs Join points. Par exemple, «toutes les méthodes qui se nomment find()».
- **Advice** : le code que l'on veut rajouter. On peut ajouter ce code avant, après, autour de la méthode...



# AOP : ADVICE

- Il est possible de définir 5 types d'advices
- **Before advice** : s'exécute avant le Join point. S'il lance une Exception, le Join point ne sera pas appelé
- **After returning advice** : s'exécute après le Join point, si celui-ci s' est bien exécuté (s'il n'y a pas ed'Exception)
- **After throwing advice** : s'exécute si une Exception a été lancée pendant l'exécution du Join point
- **After advice** : s'exécute après le Join point, qu'il y ait eu une Exception ou non
- **Around advice** : s'exécute autour du Join point. C'est l'advice le plus puissant.

# AOP : ASPECTJ

- **ASPECTJ** : C'est une extension du langage Java pour la Programmation Orientée Aspect.

## TP5 : AOP

- Comprendre le projet tp5-aop.
- Recréer le même projet sur vos postes, mais avec un projet « mavenisé », nom du projet : tp6-aop

# FONCTIONNALITÉS SPRING : ABSTRACTION DES SERVICES



# ABSTRACTION DES SERVICES

- Nous allons voir la 3ème (et dernière) grande fonctionnalité de Spring :
- Les deux premières : Inversion de Contrôle (Injection de Dépendance) et Programmation Orientée Aspect
- La 3ème : la mise à disposition d'abstractions, qui simplifient des problématiques techniques courantes
- L'abstraction utilise l'AOP (les annotations). De la même façon que l'AOP utilise l'IoC.

# ABSTRACTION DES SERVICES

- Exemple : L'accès aux données (bases de données)
- Cette action est facilitée par des classes fournies par Spring (JdbcTemplate, ....) :
  - Diminution de la taille du code, grâce à des classes fournies par Spring.
  - Gestion des exceptions réalisées par Spring pour vous.

```
1 jdbcTemplate.update("INSERT INTO...");  
2  
3 List<T> results = jdbcTemplate.query("SELECT * FROM...", new RowMapper<T>() {...});
```

# ABSTRACTION DES SERVICES

- Spring JDBC va nous aider à faire des requêtes en base de données :
- Gestion des accès à la base de données
- Aide à l'exécution du SQL
- Gestion des Exceptions

# ABSTRACTION DES SERVICES

- Voici une configuration Spring typique pour l'accès à la base de données (data source) :

```
<bean id="datasource2" destroy-method="close"  
class="org.apache.commons.dbcp.BasicDataSource">  
  <property name="driverClassName" value="${db.driver}" />  
  <property name="url" value="${db.url}" />  
  <property name="username" value="${db.username}" />  
  <property name="password" value="${db.password}" />  
</bean>
```



# ABSTRACTION DES SERVICES

- La Data Source est gérée par Spring
- C'est Spring JDBC qui va gérer le fait d'ouvrir une connexion JDBC et de la refermer
- Il n'y a plus à coder ces parties techniques
- Cela élimine les risques d'erreurs : en cas d'Exception, c'est Spring qui se charge de correctement fermer la connexion
- Cela reprend ce que nous avons vu au début
- Spring se charge de la plomberie : ouvrir/fermer les connexions, gérer les Exceptions
- Au développeur de coder le métier : la requête SQL

# ABSTRACTION DES SERVICES

- JdbcTemplate permet de faire des requêtes JDBC (CRUD) en une seule ligne,

```
/**
 * Enregistrement d'un Employé avec Spring JdbcTemplate
 * Avantages : Spring gère en interne Statements SQL, Connection DB, Exceptions,
 *
 */
public static void saveEmployeeJdbcTemplate(Employe employe) {

    final String EMPLOYE_INSERT =
        "insert into employe (id, login, password, prenom, nom, email, role) " +
        "values (?, ?, ?, ?, ?, ?, ?)";

    ClassPathXmlApplicationContext appContext = new ClassPathXmlApplicationContext("spring-data.xml");
    JdbcTemplate jdbcTemplate = (JdbcTemplate) appContext.getBean("jdbcTemplate");

    jdbcTemplate.update(EMPLOYE_INSERT,
        new Object[] { employe.getId(), employe.getLogin(), employe.getPassword(),
            employe.getPrenom(), employe.getNom(), employe.getEmail(),
            employe.getRole() });
}
```

- C'est plus pratique que la méthode «manuelle» :

# ABSTRACTION DES SERVICES

```
public static Employee getEmployeebyId(Integer id) {
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;

    try {
        final String EMPLOYE_QUERY =
            "select id, login, email, password, prenom, nom, role " +
            " from employee where id=?";

        ClassPathXmlApplicationContext appContext = new ClassPathXmlApplicationContext("spring-data.xml");
        DataSource dataSource = (DataSource) appContext.getBean("datasource2");

        conn = dataSource.getConnection();
        stmt = conn.prepareStatement(EMPLOYE_QUERY);

        stmt.setInt(1, id);
        rs = stmt.executeQuery();
        Employee employee = null;
        if(rs.next()) {
            employee = new Employee();
            employee.setId(rs.getInt("id"));
            employee.setEmail(rs.getString("email"));
            employee.setPassword(rs.getString("password"));
            employee.setLogin(rs.getString("login"));
            employee.setNom(rs.getString("nom"));
            employee.setPrenom(rs.getString("prenom"));
            employee.setRole(rs.getString("role"));
        }

        return employee;

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

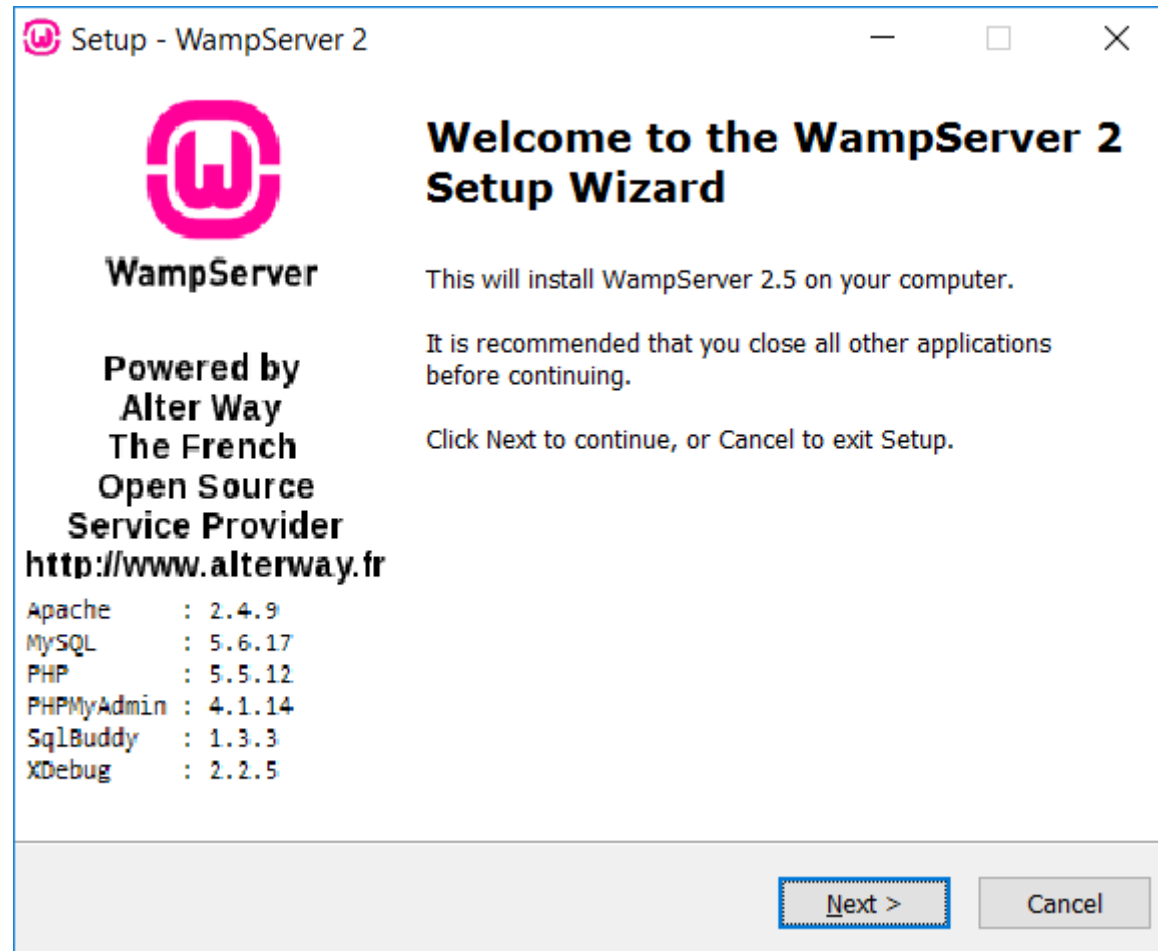
## TP6 : ABSTRACTION DES SERVICES

- Développer un projet qui permet d'interagir avec une base de données en utilisant Spring JDBC :
- Les étapes à suivre sont décrites dans les slides suivants :

# TP6 : ABSTRACTION DES SERVICES

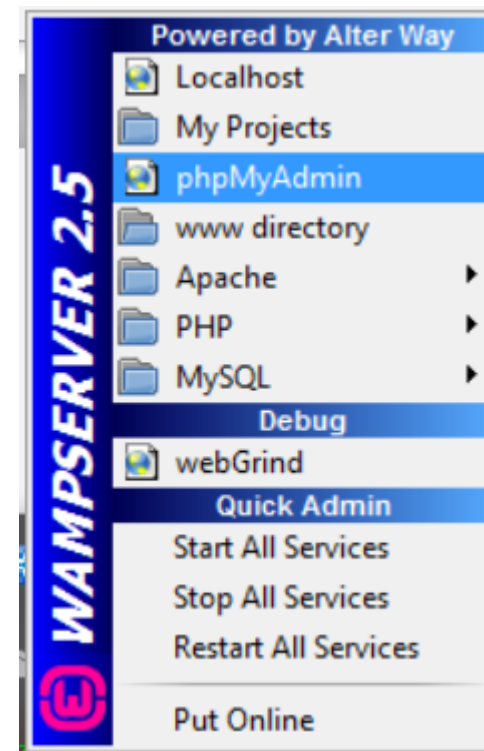
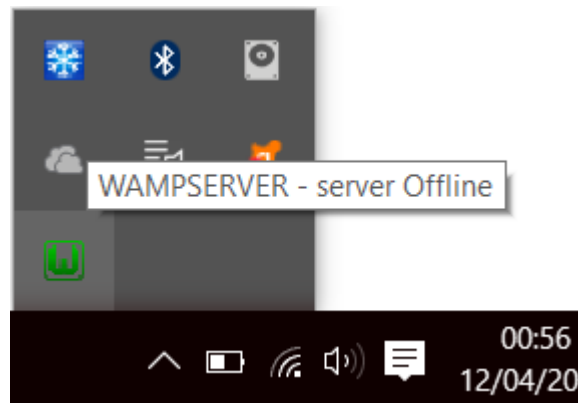
- Création et Alimentation base de données
- Installation **WAMP** : Serveur **A**pache2/**P**HP5/**M**ySQL5 sous **W**indows
- Téléchargez la dernière version de WAMP : <https://wamp-server-wamp5.fr.uptodown.com/windows> ,  
Lancez l'installation Installez WAMP5 en double-cliquant sur le fichier téléchargé.

# TP6 : ABSTRACTION DES SERVICES



# TP6 : ABSTRACTION DES SERVICES

- Lancer l'interface de l'administration de la base de données **phpMyAdmin** :



# TP6 : ABSTRACTION DES SERVICES

- Créez une base de données nommée 'formation' :

Créer une base de données ?

formation utf8\_bin Créer

Note: L'activation des statistiques peut causer un tra

✓ La base de données formation a été créée.

Base de données	Interclassement
<input type="checkbox"/> formation	utf8_bin Vérifier les privilèges
<input type="checkbox"/> information_schema	utf8_general_ci Vérifier les privilèges
<input type="checkbox"/> mysql	latin1_swedish_ci Vérifier les privilèges
<input type="checkbox"/> performance_schema	utf8_general_ci Vérifier les privilèges
<input type="checkbox"/> test	latin1_swedish_ci Vérifier les privilèges
<b>Total: 5</b>	latin1_swedish_ci



# TP6 : ABSTRACTION DES SERVICES

- Importer le fichier SQL : formation.sql



## Importation dans la base de données «formation»

### Fichier à importer :

Le fichier peut être comprimé (gzip, bzip2, zip) ou non.

Le nom du fichier comprimé doit se terminer par **[format].[compression]**. Exemple: **.sql.zip**

Parcourir :  formation.sql (Taille maximum: 128Mio)

Jeu de caractères du fichier :

# TP6 : ABSTRACTION DES SERVICES

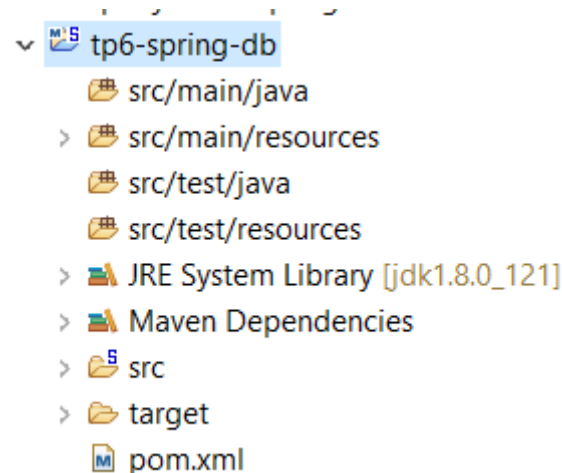
```
-- formation.sql : Base de données: `formation`

-- Structure de la table `employe`
CREATE TABLE `employe` (
  `ID` int(10) unsigned NOT NULL auto_increment,
  `login` varchar(25) NOT NULL default "",
  `password` varchar(10) NOT NULL default "",
  `nom` varchar(30) NOT NULL default "",
  `prenom` varchar(30) NOT NULL default "",
  `email` varchar(30) default NULL,
  `role` varchar(10) NOT NULL default "",
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

-- Contenu de la table `employe`
INSERT INTO `employe` VALUES (2, 'CDE', 'm2i', 'DEVOT', 'Corinne', 'corinne.devot@m2i.com', 'developper');
INSERT INTO `employe` VALUES (3, 'NIN', 'm2i', 'ING', 'Nathalie', 'nathalie.ing@m2i.com', 'productOwner');
INSERT INTO `employe` VALUES (1, 'PMA', 'm2i', 'HASSINI', 'Mourad', 'mourad.hassini@m2i.com', 'scumMaster');
```

# TP6 : ABSTRACTION DES SERVICES

- Projet Eclipse 'tp6-spring-db' :
- Création du projet Maven (avec skip archetype)  
File/new Maven Project/  
Group Id : com.m2i.formation  
Artefact Id & Nom : tp6-spring-db. Puis Finish



# TP6 : ABSTRACTION DES SERVICES

- Rajouter la dépendance vers spring et spring-jdbc dans pom.xml

```
<dependency>  
<groupId>org.springframework</groupId>  
<artifactId>spring-context</artifactId>  
<version>4.3.7.RELEASE</version>  
</dependency>
```

```
<dependency>  
<groupId>org.springframework</groupId>  
<artifactId>spring-jdbc</artifactId>  
<version>4.3.7.RELEASE</version>  
</dependency>
```

# TP6 : ABSTRACTION DES SERVICES

- Installation librairie driver JDBC : Ajouter dans ma ven la dépendance vers **mysql-connector-java- 5.1.41.jar** (à partir de <https://mvnrepository.com> )
- Installation libairies Pool de **connection : commons-dbcp-1.4.jar** (ce jar dépend de **commons-pool-1.5.4.jar**).
- Configuration Log4j : Ajouter la dépendance vers **log4j.1.2.17**. Ajouter le fichier log4j.xml dans src/main/resources.

# TP6 : ABSTRACTION DES SERVICES

```
<dependency>  
<groupId>mysql</groupId>  
<artifactId>mysql-connector-java</artifactId>  
<version>5.1.41</version>  
</dependency>
```

```
<dependency>  
<groupId>commons-dbcp</groupId>  
<artifactId>commons-dbcp</artifactId>  
<version>1.4</version>  
</dependency>
```

```
<dependency>  
<groupId>log4j</groupId>  
<artifactId>log4j</artifactId>  
<version>1.2.17</version>  
</dependency>
```

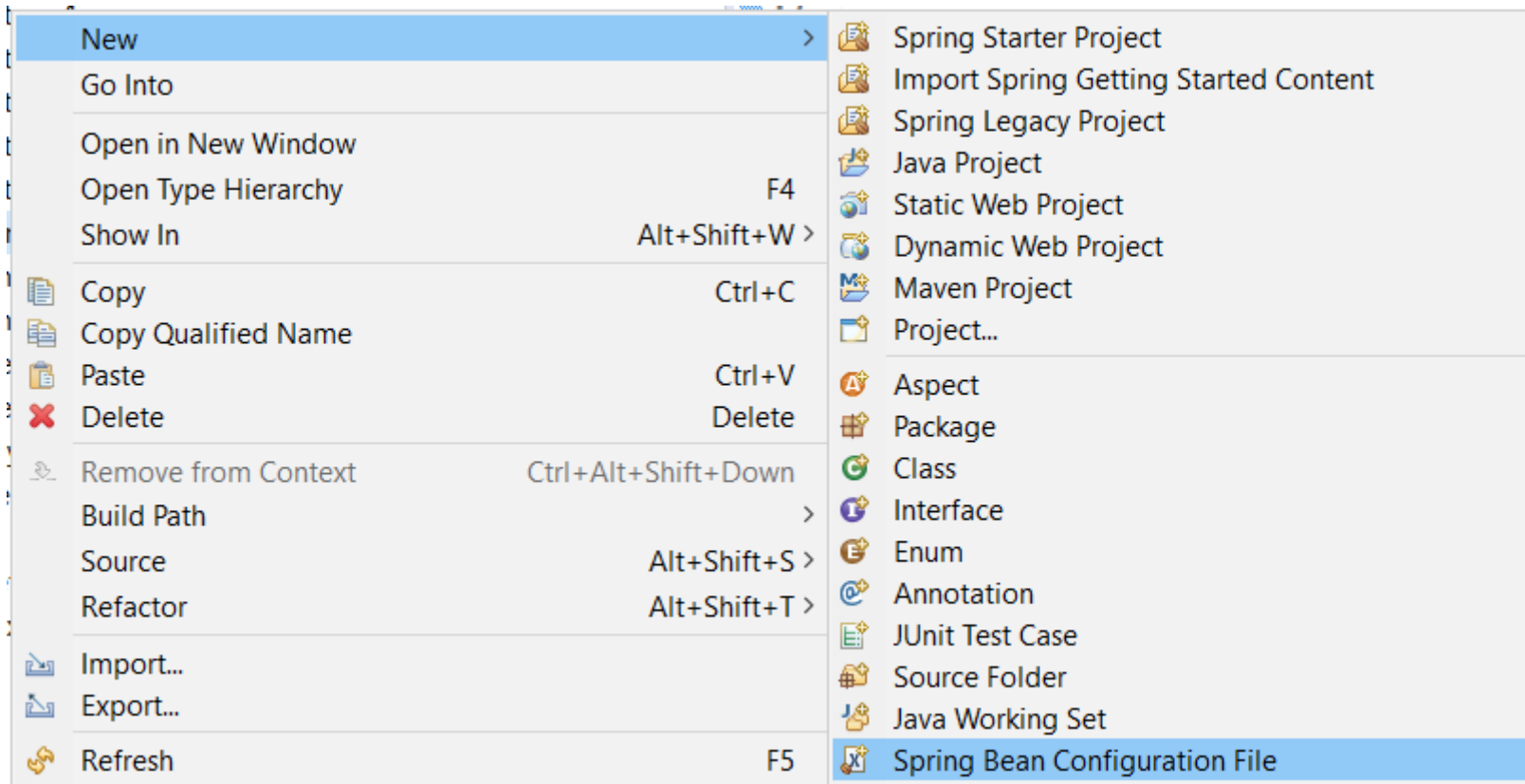
# TP6 : ABSTRACTION DES SERVICES

- Exemple de log4j.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration PUBLIC
"http://logging.apache.org/log4j/docs/api/org/apache/log4j/xml/log4j.dtd"
"http://logging.apache.org/log4j/docs/api/org/apache/log4j/xml/log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
  <appender name="console" class="org.apache.Log4j.ConsoleAppender">
    <layout class="org.apache.Log4j.SimpleLayout" />
  </appender>
  <root>
    <level value="info" />
    <appender-ref ref="console" />
  </root>
</log4j:configuration>
```

# TP6 : ABSTRACTION DES SERVICES

- Fichier de déclaration des beans : En utilisant l'assistant de création de bean Spring, créez dans 'src/main/resources' du projet 'tp6-spring-db', un fichier **spring-data.xml** qui sera dédié à la définition de beans relatifs à la base de données.





# TP6 : ABSTRACTION DES SERVICES

Spring-data.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">

</beans>
```

# TP6 : ABSTRACTION DES SERVICES

- Beans d'accès aux données :

Configuration propriétés de la base de données

Créez un fichier **db.properties** (New File) contenant des informations de connexion en base de données, dans `src/main/resources` :

# Configuration propriétés de la base de données

`db.driver=com.mysql.jdbc.Driver`

`db.url=jdbc:mysql://localhost/formation`

`db.login=root`

`db.password=`

# TP6 : ABSTRACTION DES SERVICES

- Déclarez dans spring-data.xml un bean de type **PropertyPlaceholderConfigurer**, qui assure la récupération dynamique des informations présentes dans le fichier de propriété **db.properties** :

```
<!-- Déclaration du PropertyPlaceholderConfigurer -->
<bean
    class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
        <list><value>classpath:db.properties</value></list>
    </property>
</bean>
```

## TP6 : ABSTRACTION DES SERVICES

- Déclaration beans '**datasource1**' et '**datasource2**'  
Dans le fichier spring-data.xml, déclarez 2 beans 'datasource1' et 'datasource2' correspondant à 2 types de configuration de la base de données
  - 'datasource1' : configuration via la classe `org.springframework.jdbc.datasource.DriverManagerDataSource`, avec information d'accès à la base (URL, Driver ; login, pwd) écrites 'en dur' dans le fichier XML
  - 'datasource2' : configuration via classe `org.apache.commons.dbcp.BasicDataSource`, (de Apache DBCP DataBase Connexion Pool) avec informations d'accès à la base récupérées d'un fichier de propriété : `db.properties` du package `com.m2i.formation.entity`

# TP6 : ABSTRACTION DES SERVICES

```
<!-- Déclaration de la DATASOURCES -->
```

```
<bean id="datasource1"  
class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
<property name="driverClassName" value="com.mysql.jdbc.Driver" />  
<property name="url" value="jdbc:mysql://localhost/formation" />  
<property name="username" value="root" />  
<property name="password" value="" />  
</bean>
```

```
<bean id="datasource2" destroy-method="close"  
class="org.apache.commons.dbcp.BasicDataSource">  
<property name="driverClassName" value="${db.driver}" />  
<property name="url" value="${db.url}" />  
<property name="username" value="${db.login}" />  
<property name="password" value="${db.password}" />  
</bean>
```

# TP6 : ABSTRACTION DES SERVICES

- Domaine Métier
- Créer le Bean 'Employe', dans un package `com.m2i.formation.business`, possédant les attributs correspondants aux champs de la base de données :
  - id
  - login
  - password
  - nom
  - prenom
  - email
  - rôle

# TP6 : ABSTRACTION DES SERVICES

```
Employee.java
1 package com.m2i.formation.business;
2
3 public class Employe {
4
5     private int id;
6     private String login;
7     private String password;
8     private String nom;
9     private String prenom;
10    private String email;
11    private String role;
12
13    public int getId() {
14        return id;
15    }
16    public void setId(int id) {
17        this.id = id;
18    }
19 }
```

# TP6 : ABSTRACTION DES SERVICES

- Enregistrement d'un employé en Base de données  
Dans Eclipse, créez le package 'business'  
Dans le package domaine, et en utilisant assistant Eclipse (Cliquez droit sur le code + Source/Generate...) créez pour le bean Employe,
- Ainsi que :
  - un constructeur sans paramètre : Employe()
  - un constructeur avec les 7 paramètres : Employe(int,String,String,String,String,String)
  - Getters/Setters pour chaque propriété de la classe Employe



# TP6 : ABSTRACTION DES SERVICES

```
Employee.java x
1 package com.m2i.formation.business;
2
3 public class Employee {
4
5     private int id;
6     private String login;
7     private String password;
8     private String nom;
9     private String prenom;
10    private String email;
11    private String role;
12
13    public Employee () {
14
15    }
16
17    public Employee (int id, String login, String password, String nom, String prenom, String email, String role) {
18        this.id = id;
19        this.login = login;
20        this.password = password;
21        this.nom = nom;
22        this.prenom = prenom;
23        this.email = email;
24        this.role = role;
25    }
26
27    public int getId() {
28        return id;
29    }
30 }
```

# TP6 : ABSTRACTION DES SERVICES

- Mise en œuvre JDBC pure dans Spring
- Dans cette partie vous analysez l'utilisation de Spring avec JDBC pure.

importez dans le package `com.m2i.formation.entity` le fichier **SpringJDBC.java**

- SpringJDBC implémente les méthodes :
  - `saveEmploye(Employe employe)` : Enregistrement d'un employé en base, avec méthode JDBC pure
  - `getEmployeById()` : récupération d'un objet `Employe` à partir de son id :

# TP6 : ABSTRACTION DES SERVICES

- Ci-dessous méthode getEmployeById() :

```
package com.m2i.formation.entity;
import ...
import com.m2i.formation.business.Employe;
public class SpringJDBC {
    public Employe getEmployeById (int id) {
        java.sql.Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        Employe employe = new Employe();
        try {
            final String SELECT_QUERY = "SELECT ID, LOGIN, EMAIL, PASSWORD, PRENOM, NOM, ROLE FROM EMPLOYE WHERE ID = ?";
            ApplicationContext appContext = (ApplicationContext) new ClassPathXmlApplicationContext("spring-data.xml");
            DataSource datasource = (DataSource) appContext.getBean("datasource2");
            conn = datasource.getConnection();
            stmt = conn.prepareStatement(SELECT_QUERY);
            stmt.setInt(1, id);
            rs = stmt.executeQuery();
            if (rs.next()) {
                employe.setId(rs.getInt("ID")); employe.setLogin(rs.getString("LOGIN")); employe.setPrenom(rs.getString("PRENOM"));
                employe.setNom(rs.getString("NOM")); employe.setPassword(rs.getString("PASSWORD")); employe.setEmail(rs.getString("EMAIL"));
                employe.setRole(rs.getString("ROLE"));
            }
        } catch (SQLException e) { e.printStackTrace(); }
        return employe;
    }
}
```

# TP6 : ABSTRACTION DES SERVICES

```
public void saveEmployeById (Employee employe) {

    java.sql.Connection conn = null;
    PreparedStatement stmt = null;
    int i = 0;

    try {
        final String INSERT_QUERY = "INSERT INTO employe (login, password, nom, prenom, email, role) VALUES (?, ?, ?, ?, ?, ?)";
        ApplicationContext appContext = (ApplicationContext) new ClassPathXmlApplicationContext("spring-data.xml");
        DataSource datasource = (DataSource) appContext.getBean("datasource2");
        conn = datasource.getConnection();
        stmt = conn.prepareStatement(INSERT_QUERY);

        stmt.setString(1, employe.getLogin());
        stmt.setString(2, employe.getPassword());
        stmt.setString(3, employe.getNom());
        stmt.setString(4, employe.getPrenom());
        stmt.setString(5, employe.getEmail());
        stmt.setString(6, employe.getRole());

        i = stmt.executeUpdate();
        System.out.println("i : " + i);

    } catch (SQLException e) { e.printStackTrace(); }
```

## TP6 : ABSTRACTION DES SERVICES

- Créer la classe de test (avec un main() ou Junit) pour tester notre code :

```
SpringJDBC springJDBC = new SpringJDBC();
```

```
Employe employe = springJDBC.getEmployeById(1);  
System.out.println("employe" + employe.getLogin());
```

```
Employe employe = new Employe(1111, 'MHA', 'm2i',  
    'HASSINI', 'Mourad', 'mourad.hassini@m2i.com',  
    'scrumMaster');  
springJDBC.saveEmploye (employe);
```

## TP6 : ABSTRACTION DES SERVICES

- Mise en œuvre JdbcTemplate :
- Spring propose la classe **JdbcTemplate** pour vous simplifier l'accès à une base de données en Java.
- Déclarez un bean JdbcTemplate dans le fichier de configuration de Spring :

```
<bean id="jdbcTemplate"  
    class="org.springframework.jdbc.core.JdbcTemplate">  
<property name="dataSource" ref="datasource2" />  
</bean>
```

# TP6 : ABSTRACTION DES SERVICES

- Implémentez dans SpringJDBC la `saveEmployeJdbcTemplate()` : Enregistrement d'un employé en base, avec Template `jdbcTemplate`, fourni par SPRING :

```
public void saveEmployebyIdJdbcTemplate (Employe employe) {  
  
    final String INSERT_QUERY = "INSERT INTO employe (id, login, password, nom,  
        prenom, email, role) VALUES (?, ?, ?, ?, ?, ?, ?)";  
  
    ApplicationContext appContext = (ApplicationContext) new  
        ClassPathXmlApplicationContext("spring-data.xml");  
  
    JdbcTemplate jdbcTemplate = (JdbcTemplate) appContext.getBean("jdbcTemplate");  
  
    jdbcTemplate.update(INSERT_QUERY, new Object[] {employe.getId(),  
        employe.getLogin(), employe.getPassword(),  
        employe.getNom(), employe.getPrenom(), employe.getEmail(),  
        employe.getRole()});  
}
```

# TP6 : ABSTRACTION DES SERVICES

- Implémentez dans SpringJDBC la `getEmployeJdbcTemplateById()` :  
Récupération des informations d'un employé de la base, avec `JdbcTemplate`, fourni par SPRING :

```
public Employee getEmployebyIdJdbcTemplate (int id) {  
  
    final String SELECT_QUERY = "SELECT ID, LOGIN, EMAIL, PASSWORD,  
        PRENOM, NOM, ROLE FROM EMPLOYE WHERE ID = ? ";  
  
    ApplicationContext appContext = (ApplicationContext) new  
        ClassPathXmlApplicationContext("spring-data.xml");  
  
    JdbcTemplate jdbcTemplate = (JdbcTemplate)  
        appContext.getBean("jdbcTemplate");  
  
    return jdbcTemplate.queryForObject(SELECT_QUERY,  
        BeanPropertyRowMapper.newInstance(Employee.class), id);  
}
```



## TP6 : ABSTRACTION DES SERVICES

- Ecrivez une méthode de test associée à la méthode `saveEmployeeJdbcTemplate()`
- Ecrivez une méthode de test associée à la méthode `getEmployeeJdbcTemplate()`

# TP6 : ABSTRACTION DES SERVICES

- Mise en œuvre DAO :

Proposez dans un package `com.m2i.formation.dao` une mise en œuvre de DAO implémentant une interface `IEmployeDao` avec méthodes :

- `Employe getEmployeById(int id) ;`
- `Employe getEmployeByLogin(String login) ;`
- `void saveEmploye(Employe employe) ;`
- `List getAllEmployes() ;`
- `int getEmployesCount() ;`

# LES ANNOTATIONS

- Java a introduit la notion d'annotations en 2006, avec Java 5 .
- Spring a introduit les annotations depuis Spring 2.5 (2007), avec un début d'intégration des annotation avec Spring 2.0 (2006).

# LES ANNOTATIONS

- Les annotations Java permettent de simplifier les fichiers de configuration de Spring.
- Elles permettent de séparer plus nettement les beans techniques (DataSource,...), qui restent configurés en fichier XML, des beans fonctionnels pour lesquels les annotations apportent des simplifications.
- Pour que Spring Framework charge les beans via les annotations, il faut lui indiquer, dans son fichier de configuration, les paquetages à parcourir, Spring parcourra tous les sous-packages de celui qui est indiqué : **<context:component-scan base-package="com.m2i.formation"/>**

# LES ANNOTATIONS

- Une première façon de déclarer les beans : Pour déclarer un bean gérés par Spring, on devait ajouter une **balise** **<bean id= class=... >** dans le fichier xml de configuration Spring.
- Une deuxième façon de déclarer les bean : La balise **<context:component-scan>** indique à Spring qu'il doit rechercher dans le code certaines annotations que voici : @Repository, @Service, @Controller, @Component :

# LES ANNOTATIONS

- **@Repository** : Cette annotation existe depuis Spring 2.0 et sert à identifier un bean de type DAO.
- **@Service** : Celle-ci identifie le bean comme un service
- **@Controller** : Utile uniquement si l'on utilise SpringMVC, cette annotation indique un contrôleur Spring MVC.
- **@Component** : Cette dernière est l'annotation générique pouvant fonctionner pour n'importe quel bean

# LES ANNOTATIONS

Configuration par Annotation :

```
@Service("userService")
public class UserService {
    private String name = "Corinne DEVOT";
}
```

Configuration XML d'une bean :

```
<context:component-scan base-package="com.m2i.formation" />

<bean id="userService" class="com.m2i.formation.UserService">
    <property name="name" value="Corinne DEVOT"></property>
</bean>
```

# LES ANNOTATIONS

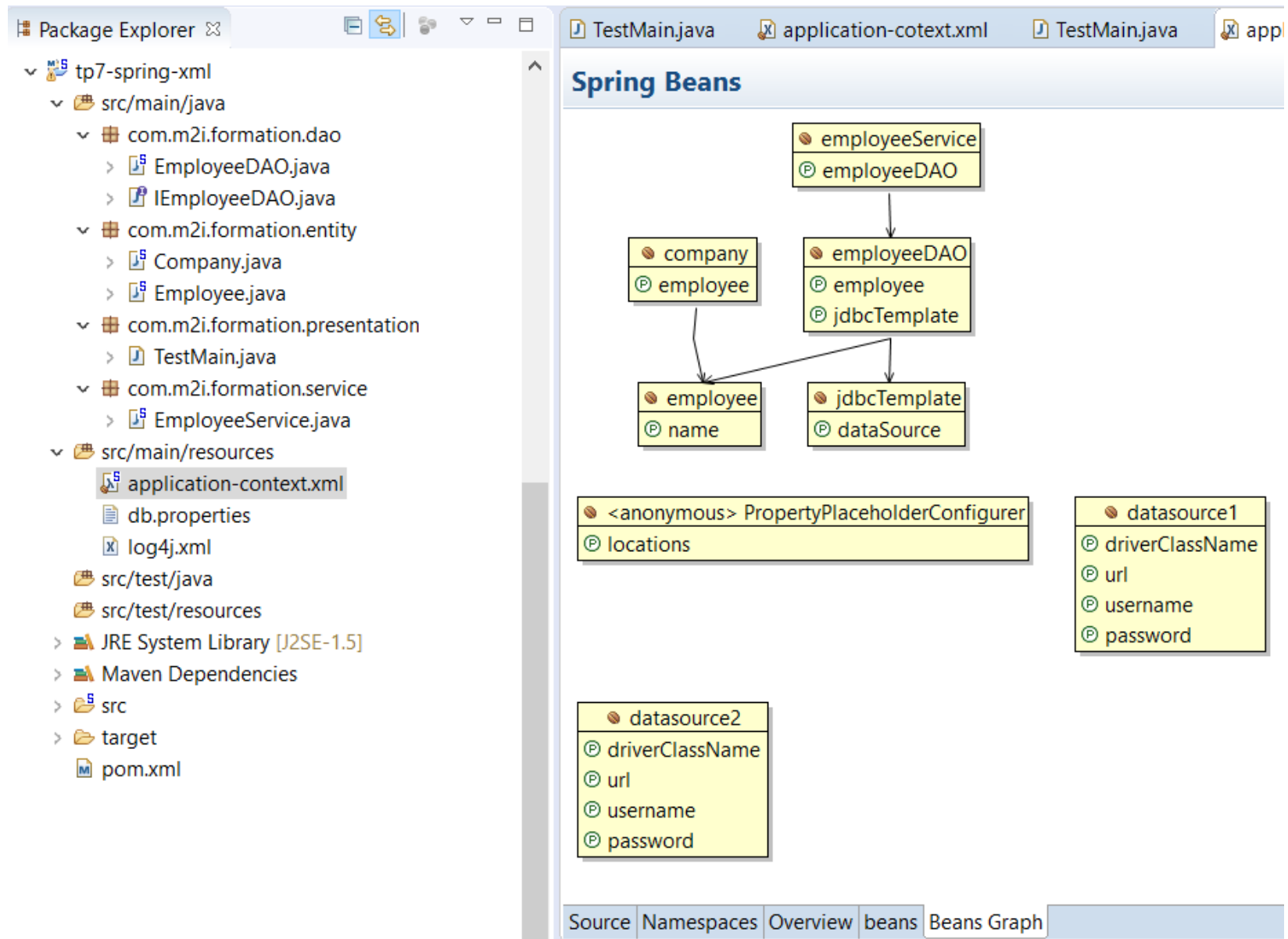
- Avec l'introduction des annotations, a-t-on toujours besoin du fichier XML de configuration des beans?



# TP7 - LES ANNOTATIONS

- Projet Spring Simple, utilisant la déclaration des beans dans le fichier de configuration XML : tp7-spring-xml
- Recréer le même projet, mais en utilisant les annotations : tp7-spring-annotation .
- Il s'agit de créer les beans : Employee.java (propriété : name, firstName, nbYearExp) et Company.java (propriétés : name, developer, techLead, manager de type employee)

# TP7 - LES ANNOTATIONS



# TP7 - LES ANNOTATIONS

- Dans Eclipse, créer un Projet de type Maven :
- Group Id : com.m2i.formation
- Nom = ArtefactId = Description : tp7-spring-xml
- Skip Archetype (à cocher).
- Ajouter la dépendance vers le jar spring-context, spring-jdbc, connecteur mysql, cmmons-dbcnp et log4j (voir TP6).
- Créer le fichier log4j.xml
- Créer le fichier application-context.xml , en utilisant l'utilitaire offert par STS.

# TP7 - LES ANNOTATIONS

- Beans métiers :
- Déclarer les beans company, employee, EmployeeService, EmployeeDAO le fichier XML
- Créer les beans Java : Company.java et Employee.java dans le package com.m2i.formation.entity
- Créer EmployeeService.java dans com.m2i.formation.service
- Créer EmployeeDAO.java dans com.m2i.formation.dao

# TP7 - LES ANNOTATIONS

- Beans techniques:
- Déclarer les beans pour l'accès à la base de données, dans le fichier XML de configuration es beans : `datasource` et `jdbcTemplate`, récupérer le fichier `db.properties` (Voir TP6).

# TP7 - LES ANNOTATIONS

- Recréer le même projet en le renommant tp7-spring-annotation
- Remplacer les configurations de bean par des annotations (ne remplacer que les beans métier).

# RÉVISION

- Quelques questions sur les notions abordées (Spring IoC, Spring AOP, Abstractions de Services, Java, Maven, ...).

