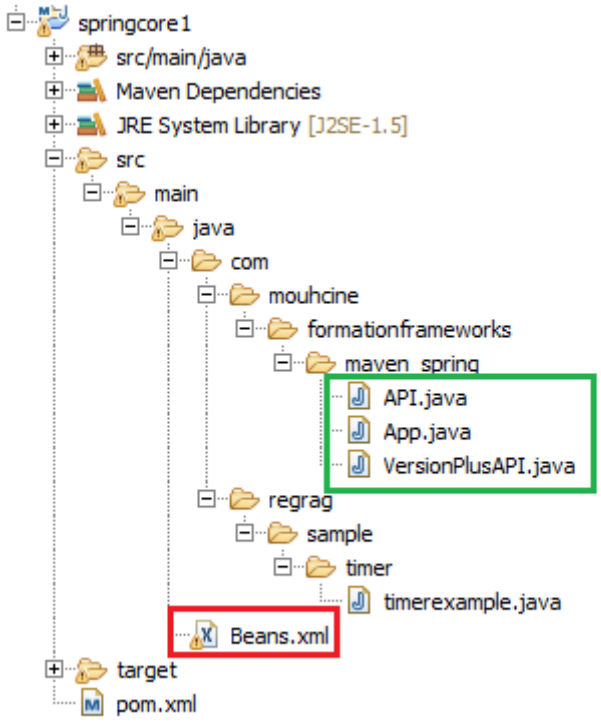
« La version 4 du spring »

Maven7 (Core)

Table des matières

I- Description de l'arborescence du projet :

Après avoir importé le fichier .ZIP springcore.zip dans le workspace :
La structure standard d'un projet springcore doit être comme suit :



Elle contient :

- Un fichier Bean.xml, ou on peut déclarer nos entités (objets) suivant un modèle de donnée (sur l'exemple la classe modèle API).
- Le fichier API.java qui contient le modèle suivant lequel on déclare nos Beans.
- Les Fichiers App.java et VersionPlusAPI.java, qui contienne le code exemple du démo.
- Le fichier timerexample.java est un la plus simple façon d'écrire un programme java avec JEE sans Bean (une classe avec main).
- Dans le POM du projet plusieurs configurations sont requises dans la balise Build.
- Sur le POM faire Exécuter Run→Build… et la commande clean install.
- Sur Les fichiers qui contiennent le main : run→2 application Java.
- Le résultat s'affiche après l'exécution sur la console.

II-Objectif du projet :

Cet exemple vise à faire exécuter un code qui exploite la bibliothèque standard (de base) de JEE, nommé core.
Le premier programme timerexemple.java : est la façon la plus simple de coder avec maven.
Cependant, ce qui est usuel c'est la création des fichiers Bean, des fichiers des modèles de données et les fichiers sources java (ce qui est le cas dans le deuxième exemple)

1- Le premier démo :

Illustre un timer qui permet d'afficher pendant 1000 ms (10s), chaque seconde un message.

On rappelle que :

a- La structure principale d'un programme doit contenir :

import java…. // appel aux autres classes

b- La fonction main :

```
public static void main(String args[])// function principal main

{
// do somethings : corp du programme principale
}
```

c- Le nom d'une classe est le même que le nom du fichier.java.

d- Une classe peut implémenter une interface et hériter d'une autre classe mère :

Dans ce cas notre classe hérite de la classe mère TimerTask :

```
public class timerexample extends TimerTask {

@Override
public void run() {

// do somethings

}
```

La fonction run() résulte de l'héritage du la classe TimerTask invoqué par l'appel :

```
import java.util.TimerTask;
```

2- Le deuxième démo :

Il s'agit d'un modèle du code le plus simple qui utilise les Beans :

a- Un fichier Bean(en format xml) pour décrire un ensemble des entités selon un modèle de données passé en attribut de la balise bean :

```
<bean id="api1" class="Nom de la classe modéle
        <property name="n1" value="v1" />
                ….
        <property name="n2" value="v2" />
</bean>
```

b- Le fichier API.java qui contient la classe modèle de donnée a passé comme attribut (avec class) à la balise Bean.

c- Les deux fichiers VersionPlusAPI.java et App.java qui permet d'exploiter le modèle par billet des Beans.

d- La syntaxe utilisée est la suivante :

```
 // Faire appel au fichier xml : déclaration d'un context
ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

Et

//déclaration d'objet instancié par l'appel au Bean par id= api1
API obj1 = (API) context.getBean("api1");
```

e- Importation des deux nouvelles classes :

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

f- Dans le POM on doit ajouter à la balise Build : l'emplacement du Beans.xml

```
<resources>
     <resource>
          <directory>src/main/java</directory>
          <includes>
               <include>Beans.xml</include>
          </includes>
     </resource>
</resources>
```

III- Fichier du projet :

1- API.java :

```
package com.mouhcine.formationframeworks.maven_spring;

public class API {
	private String key;
	private String name;
	private String year;
	private String version;
	private String technologie;

	public String getkey() {
		return key;
	}

	public void setkey(String key) {
		this.key = key;
	}

	public String getname() {
		return name;
	}

	public void setname(String name) {
		this.name = name;
	}

	public String getyear() {
		return year;
	}

	public void setyear(String year) {
		this.year = year;
	}

	public String getversion() {
		return version;
	}

	public void setversion(String version) {
		this.version = version;
	}

	public String gettechnologie() {
		return technologie;
	}

	public void settechnologie(String technologie) {
		this.technologie = technologie;
	}

		public String getShortDescription() {
		return String.format("%s %s", getkey(), getname());
	}
```

```
	public String getLongDescription() {
		return String.format("%s %s %s %s", getversion(), getyear(),
				getkey(), gettechnologie());
	}
}
```

2- VersionPlusAPI.java :

```
package com.mouhcine.formationframeworks.maven_spring;

public class VersionPlusAPI {

   public void ChangeVersionAPI(API api) {
          System.out.printf("%s in progression ...\n", api.getname());
          api.setversion("Nouvelle version");
   }

   public void ChangeYearAPI(API api) {
          System.out.printf("%s in commercialisation ...\n", api.getname());
          api.setyear("new year");
   }
}
```

3- App.java :

```
package com.mouhcine.formationframeworks.maven_spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {
   public static void main(String[] args) {

          @SuppressWarnings("resource")
ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

VersionPlusAPI maintain = new VersionPlusAPI();

          // api1 bean
          API obj1 = (API) context.getBean("api1");
          System.out.printf("I build a %s.\n", obj1.getLongDescription());
          System.out.printf("Is my : %s , version:
%s\n",obj1.getShortDescription(), obj1.getversion());
          maintain.ChangeVersionAPI(obj1);
          System.out.printf("Is my : %s new version: %s\n\n",
obj1.getkey(),obj1.getversion());

          // api2 bean
          API obj2 = (API) context.getBean("api2");
          System.out.printf("My wife build a %s.\n",
obj2.getLongDescription());
          System.out.printf("Is her %s in progression: %s\n",
obj2.getShortDescription(),obj2.getyear());
          maintain.ChangeYearAPI(obj2);
          System.out.printf("Is her %s start commercialisation now : %s\n\n",
obj2.getkey(),obj2.getyear());

          // api3 bean
          API obj3 = (API) context.getBean("api3");
          System.out.printf("I build a API technologies : %s\n",
obj3.gettechnologie().toLowerCase());

   }
}
```

4- timerexemple.java :

```
package com.regrag.sample.timer;

import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;


public class timerexample extends TimerTask {

@Override
public void run() {
   System.out.println("Start of recording at :" + "--" + new Date() + "--");
   // do save
   System.out.println("-Save in one seconde Lignes with succes!");
    doSomeWork();
    System.out.println("");
}
// simulate a time consuming task
private void doSomeWork() {
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

}


public static void main(String args[]) {


   System.out.println("");
   System.out.println("Thread write by REGRAG mouhcine  programmer/analyst JEE
: 12/07/2017");
   System.out.println("");

    TimerTask timerTask = new timerexample();
    // running timer task as daemon thread
    Timer timer = new Timer(true);
    timer.scheduleAtFixedRate(timerTask,0,1);
    // cancel after sometime
    try {
        Thread.sleep(10000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    timer.cancel();
    System.out.println("");
    System.out.println("End of recording in '10 s' at :" + "--" + new Date() +
"--");

}
}
```

5- Beans.xml :

```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-3.0.xsd">

   <bean id="api1" class="com.mouhcine.formationframeworks.maven_spring.API">
         <property name="key" value="API-01-2010" />
         <property name="name" value="SpringSecurité" />
         <property name="year" value="2010" />
         <property name="version" value="1.1.0" />
         <property name="technologie" value="JAVA" />
   </bean>

   <bean id="api2" class="com.mouhcine.formationframeworks.maven_spring.API">
         <property name="key" value="API-04-2015" />
         <property name="name" value="SpringMVC" />
         <property name="year" value="2015" />
         <property name="version" value="1.0.3" />
         <property name="technologie" value="JAVA" />
   </bean>

   <bean id="api3" class="com.mouhcine.formationframeworks.maven_spring.API">
         <property name="key" value="API-15-2013" />
         <property name="name" value="Plugin" />
         <property name="year" value="2013" />
         <property name="version" value="1.2.2" />
         <property name="technologie" value="PHP" />
   </bean>

</beans>
```

6- POM :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>com.regrag.formationframeworks</groupId>
   <artifactId>springcore1</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <packaging>jar</packaging>

   <properties>
         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   </properties>

   <dependencies>
         <dependency>
               <groupId>junit</groupId>
               <artifactId>junit</artifactId>
               <version>3.8.1</version>
               <scope>test</scope>
         </dependency>
         <dependency>
               <groupId>org.springframework</groupId>
               <artifactId>spring-core</artifactId>
               <version>3.2.4.RELEASE</version>
         </dependency>
         <dependency>
               <groupId>org.springframework</groupId>
               <artifactId>spring-context</artifactId>
               <version>3.2.4.RELEASE</version>
         </dependency>
   </dependencies>

   <build>

         <resources>
               <resource>
                     <directory>src/main/java</directory>
                     <includes>
                           <include>Beans.xml</include>
                     </includes>
               </resource>
         </resources>

         <plugins>
               <plugin>
                     <groupId>org.apache.maven.plugins</groupId>
                     <artifactId>maven-dependency-plugin</artifactId>
                     <version>2.5.1</version>
                     <executions>
                           <execution>
                                 <id>copy-dependencies</id>
                                 <phase>package</phase>
                                 <goals>
                                       <goal>copy-dependencies</goal>
                                 </goals>
```

```
                <configuration>
<includeGroupIds>org.springframework,commons-logging</includeGroupIds>
<outputDirectory>${project.build.directory}/dependency-jars/</outputDirectory>
                </configuration>
                </execution>
                </executions>
                </plugin>
                <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>2.4</version>
                <configuration>
                <archive>
                <manifest>
                <addClasspath>true</addClasspath>

  <mainClass>com.regrag.formationframeworks.maven_spring.App</mainClass>
  <classpathPrefix>dependency-jars/</classpathPrefix>
                </manifest>
                </archive>
                </configuration>
                </plugin>
        </plugins>

  </build>
 </project>
```