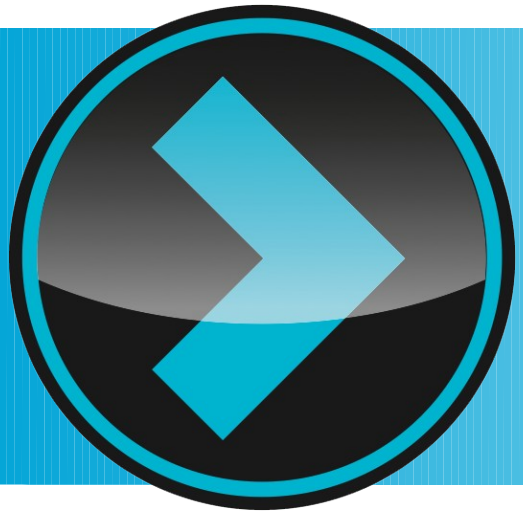


Spring IoC

Java WEB M1

ingésup
L' École d'Experts en Informatique
> Bordeaux/Toulouse

Course Goals

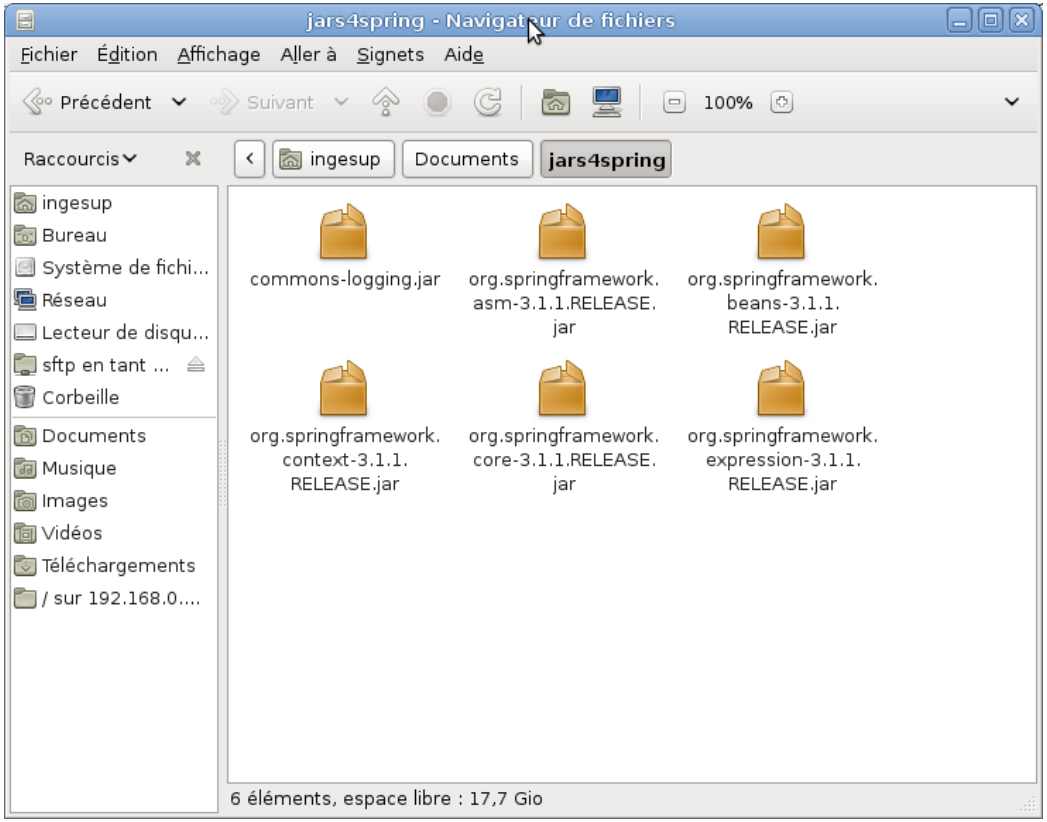
- What is it for ?
- Which needed files ?
- Some examples to use beans
- 3tier example

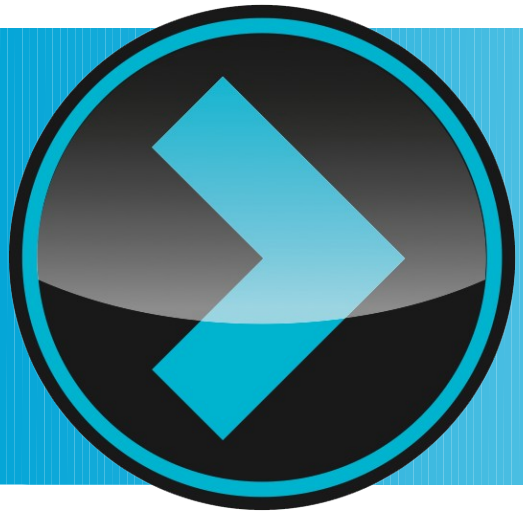
What is it for ?

- IoC means “Inversion of Control” or “Depency Injection”
- Spring IoC use beans we can instanciate out of code ; including bean depencies...
- FINAL GOAL : We want to separate completely the 3 Layers Web, Business logic, and DAO
- We will learn Spring IoC with some no-web examples...

Needed files

- JARS :
  - Retrieve these jars

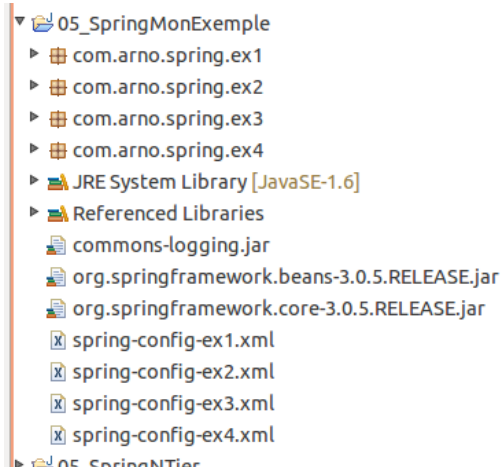


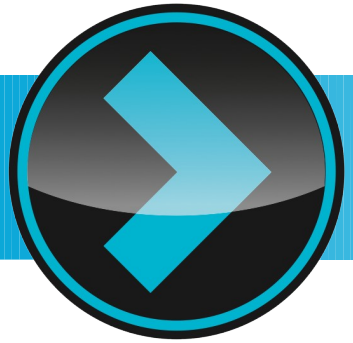


4 examples...

Examples project

- Create a no-web project with its sources under root. Place 3 jars under root and add them to build path.
- We create 4 packages :
  - com.ingesup.spring.ex1..4
- XML Spring configuration files will be seen after...

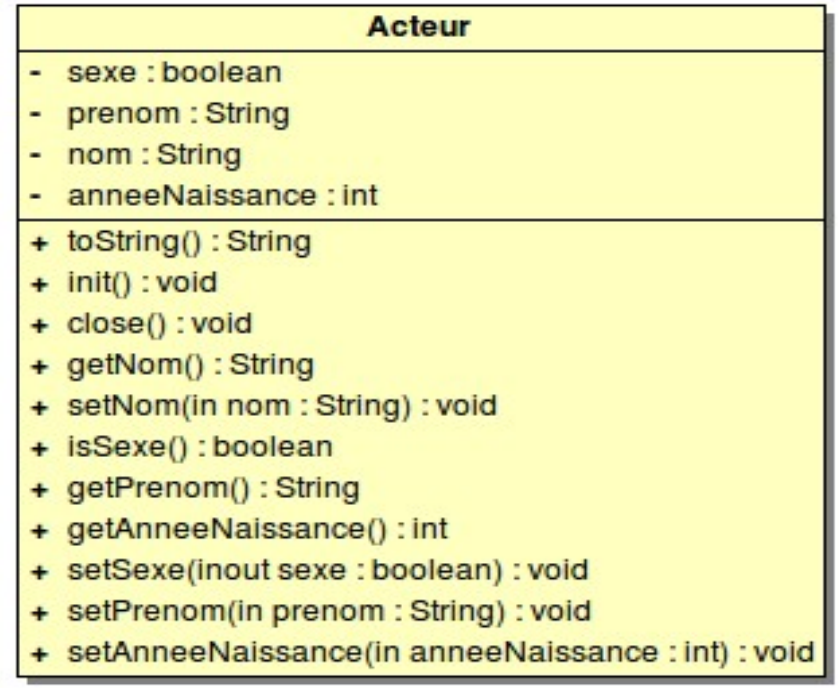




Example 1 : one bean

Example 1 : a simple bean

- Create a “Acteur” bean class :
  - with private attributes (nom, prenom, sexe, and anneeNaissance)
  - With all getters and setters
  - Override toString() to return all values
  - Add a special constructor called for example **init()** containing only a console print information
  - Add a special destructor called for example **close()** with another console print info.



Example 1 : Configuration file

```
<!-- spring-config-ex1.xml : a simple bean usage -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
      <bean id="acteur1" class="com.arno.spring.ex1.Acteur" init-method="init" destroy-method="close">
            <property name="nom" value="FOSTER" />
            <property name="prenom" value="Jodie" />
            <property name="sexe" value="true" />
            <property name="anneeNaissance" value="1962" />
      </bean>
      <bean id="acteur2" class="com.arno.spring.ex1.Acteur" init-method="init" destroy-method="close">
            <property name="nom" value="WILLIS" />
            <property name="prenom" value="Bruce" />
            <property name="sexe" value="false" />
            <property name="anneeNaissance" value="1955" />
      </bean>
</beans>
```

- Bean attributes :
  - Which id
  - Which class (in which package)
  - Which init and destroy methods
- Bean elements :
  - Property which initialize values into attributes (through setters !!)

Example 1 : Main class

```
package com.ingesup.web.spring.ex1;

import org.springframework.beans.factory.BeanFactory;
import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.ingesup.web.spring.ex1.beans.Acteur;


public class MainEx1
{
	public static void main(String[] args)
	{
		AbstractApplicationContext context=new ClassPathXmlApplicationContext(
			new String[]{"spring-config-ex1.xml"});
		context.registerShutdownHook();
		BeanFactory factory=context;
		Acteur a1=(Acteur) factory.getBean("acteur1");
		System.out.println(a1);
		Acteur a2=(Acteur) factory.getBean("acteur2");
		System.out.println(a2);
	}
}
```

Spring config : more syntaxes

Spring config syntaxes

- In a bean element, you can define an attribute with :

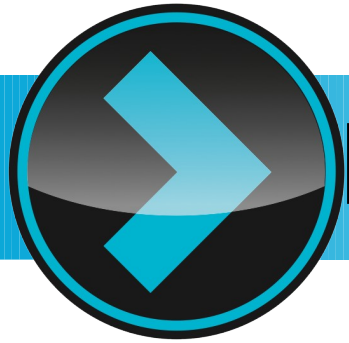
```
<property name="..." value="..." />
```

- Or with :

```
<constructor-arg index="n°" value="..." />
```

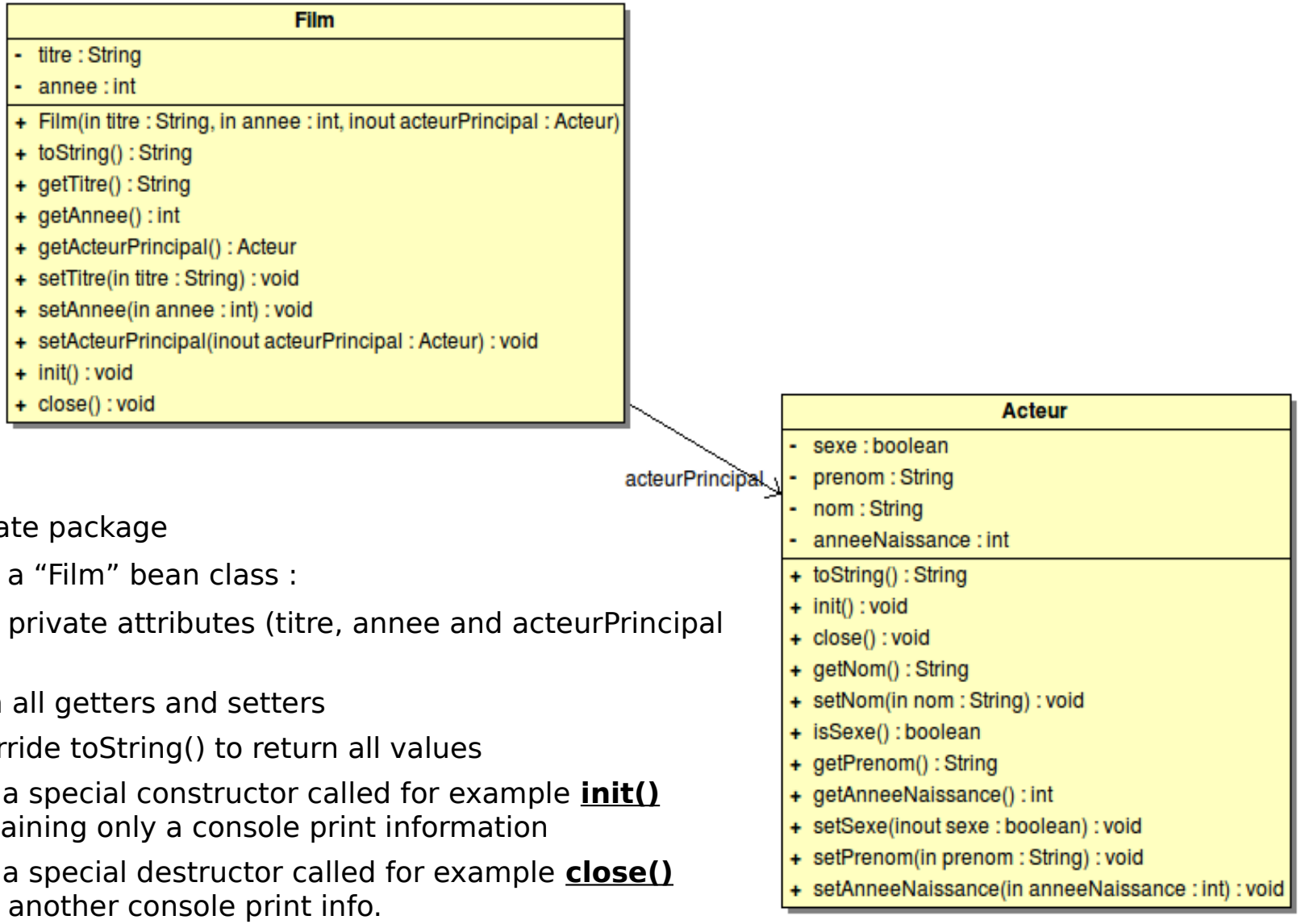
- You can insert an already defined bean in another bean with :

```
<constructor-arg index="n°">
    <ref local="bean name" />
</constructor-arg>
```



Example 2 : two beans

Example 2 : a bean using another bean



- Duplicate package
- Create a “Film” bean class :
  - with private attributes (titre, annee and acteurPrincipal)
  - With all getters and setters
  - Override toString() to return all values
  - Add a special constructor called for example **init()** containing only a console print information
  - Add a special destructor called for example **close()** with another console print info.

Example 2 : a bean using another bean

- Duplicate XML configuration file and keep the 2 actor beans (care : class package has changed).
- Create 2 film beans with :
  - Property names and values for title and year
  - To store already defined actor bean in a film, see precedent syntax.
- In the main class, retrieve film beans : the actor beans, thanks to hierarchy classes, will be automatically retrieved.

```
AbstractApplicationContext context=new ClassPathXmlApplicationContext(new String[]{"spring-config-ex2.xml"});
context.registerShutdownHook();
BeanFactory factory=context;
Film f1=(Film) factory.getBean("film1");
System.out.println(f1);
Film f2=(Film) factory.getBean("film2");
System.out.println(f2);
```

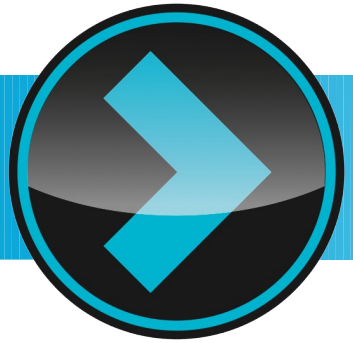
```
Appel de la méthode init pour Acteur
Appel de la méthode init pour Acteur
Appel de la méthode init pour Film
Appel de la méthode init pour Film
Le film Contact est sorti en 1997 et a pour acteur principal Mme Jodie FOSTER  née en 1962
Le film L'armée des 12 singes est sorti en 1995 et a pour acteur principal M. Bruce WILLIS  né en 1955
Appel de la méthode close pour Film
Appel de la méthode close pour Film
Appel de la méthode close pour Acteur
Appel de la méthode close pour Acteur
```

Spring config : more syntaxes again

Spring config syntaxes

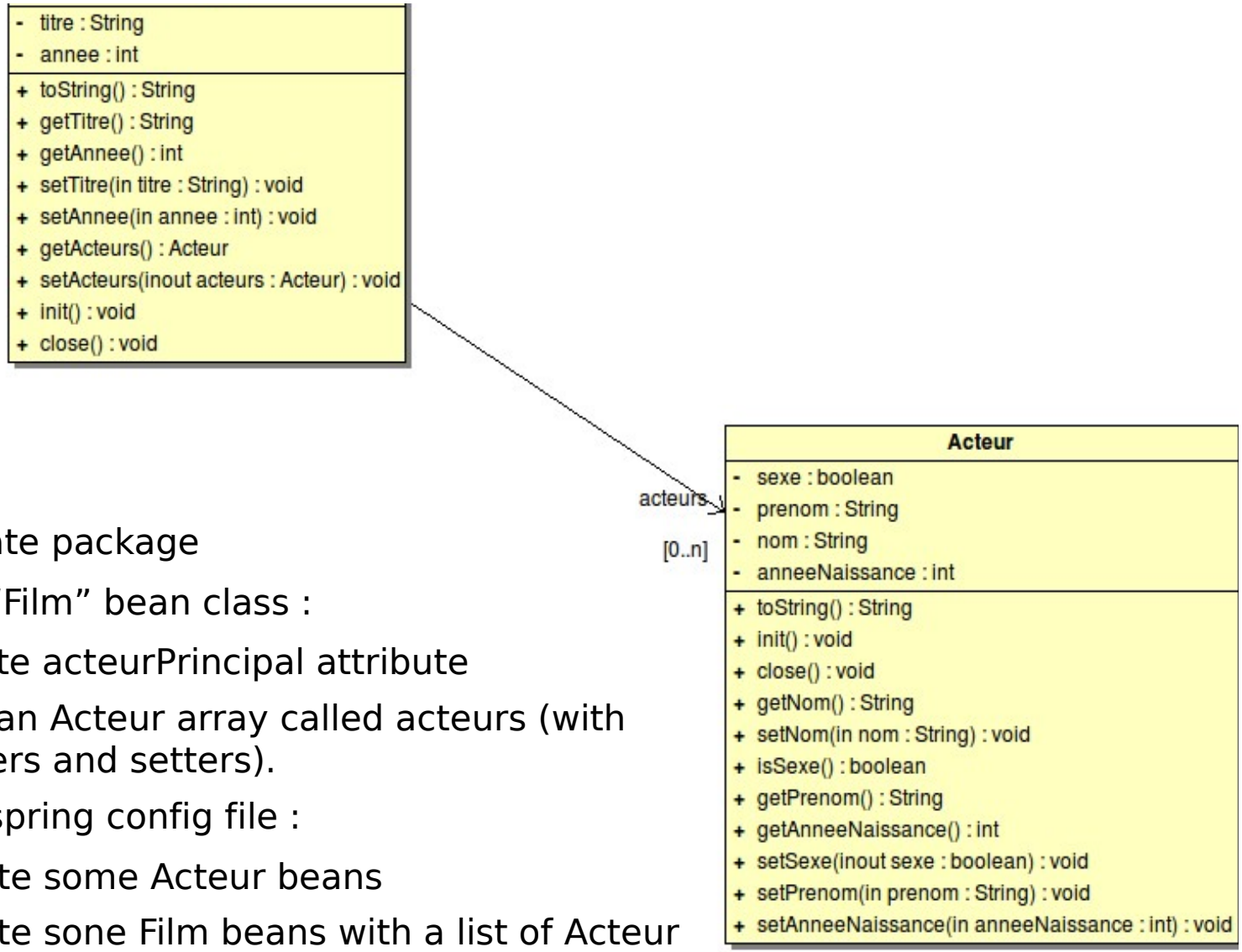
- In a bean element, you can define an list with already defined beans :

```
<property name="...">
    <list>
        <ref local="bean 1" />
        <ref local="bean 2" />
    </list>
</property>
```



Example 3 : beans array

Example 3 : a bean using beans array



- Duplicate package
- In the “Film” bean class :
  - Delete acteurPrincipal attribute
  - Add an Acteur array called acteurs (with getters and setters).
- In the spring config file :
  - Create some Acteur beans
  - Create sone Film beans with a list of Acteur

Example 3 : a bean using beans array

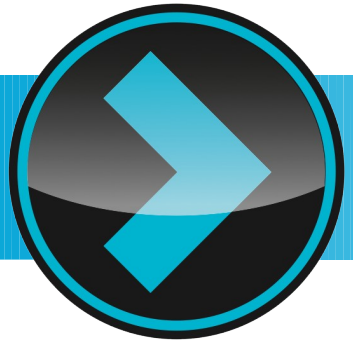
```
init Acteur [nom=[M. Bruce WILLIS], année de naissance=[1955]]
init Acteur [nom=[Mme Madeleine STOWE], année de naissance=[1958]]
...
film1=Film : titre=[L'armée des 12 singes],
             année=[1995],
             Acteurs=[
                   nom=[M. Bruce WILLIS], année de naissance=[1955] ;
                   nom=[Mme Madeleine STOWE], année de naissance=[1958]
             ]

init Acteur [nom=[Mme Jodie FOSTER], année de naissance=[1962]]
init Acteur [nom=[M. William FICHTNER], année de naissance=[1956]]
...
film2=Film : titre=[Contact],
             année=[1997],
             Acteurs=[
                   nom=[Mme Jodie FOSTER], année de naissance=[1962] ;
                   nom=[M. William FICHTNER], année de naissance=[1956]
             ]
```

Spring config : more syntaxes again

Spring config syntaxes

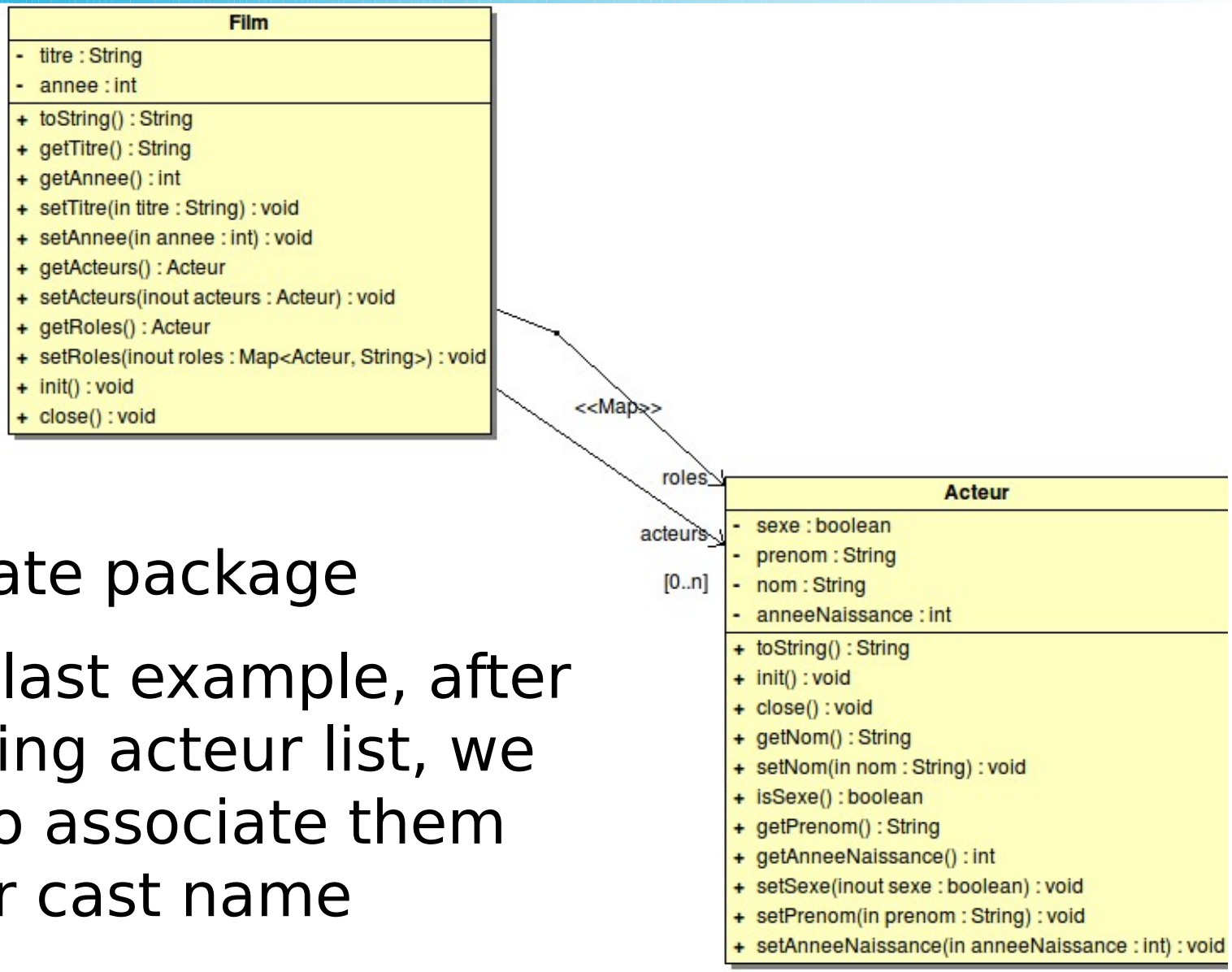
- In a bean element, you can define a Map with already defined beans. In the following example, the key is a bean, and the value is a String

```
<property name="...">
    <map>
        <entry>
            <key>
                <ref local="bean 1" />
            </key>
            <value>String value 1</value>
        </entry>
        <entry>
            <key>
                <ref local="bean 2" />
            </key>
            <value>String value 2</value>
        </entry>
    </map>
</property>
```



Example 4 : (bean) Map

Example 4 : a bean using a Map



- Duplicate package
- In this last example, after retrieving acteur list, we want to associate them to their cast name

Example 4 : a bean using a Map

```
init Acteur [nom=[M. Bruce WILLIS],
             année de naissance=[1955]
            ]
init Acteur [nom=[Mme Madeleine STOWE],
             année de naissance=[1958]
            ]
init Film [Film :
             titre=[L'armée des 12 singes],
             année=[1995],
             Acteurs=[nom=[M. Bruce WILLIS],
                          année de naissance=[1955] ;
                      nom=[Mme Madeleine STOWE],
                          année de naissance=[1958]] ;
             Rôles [{ nom=[M. Bruce WILLIS], année de naissance=[1955]=James COLE,
                      nom=[Mme Madeleine STOWE], année de naissance=[1958]=Dr. Kathryn Railly
                    }]]
```

The config file chronology :

- First create some Acteur beans
- Create a Film Bean
  - With simple attributes
  - With the array of bean Acteur
  - With the Map association between an Acteur and a name

Points abordés

Bean characteristics

<jsp:usebean

<jsp:getproperty

<jsp:setproperty