



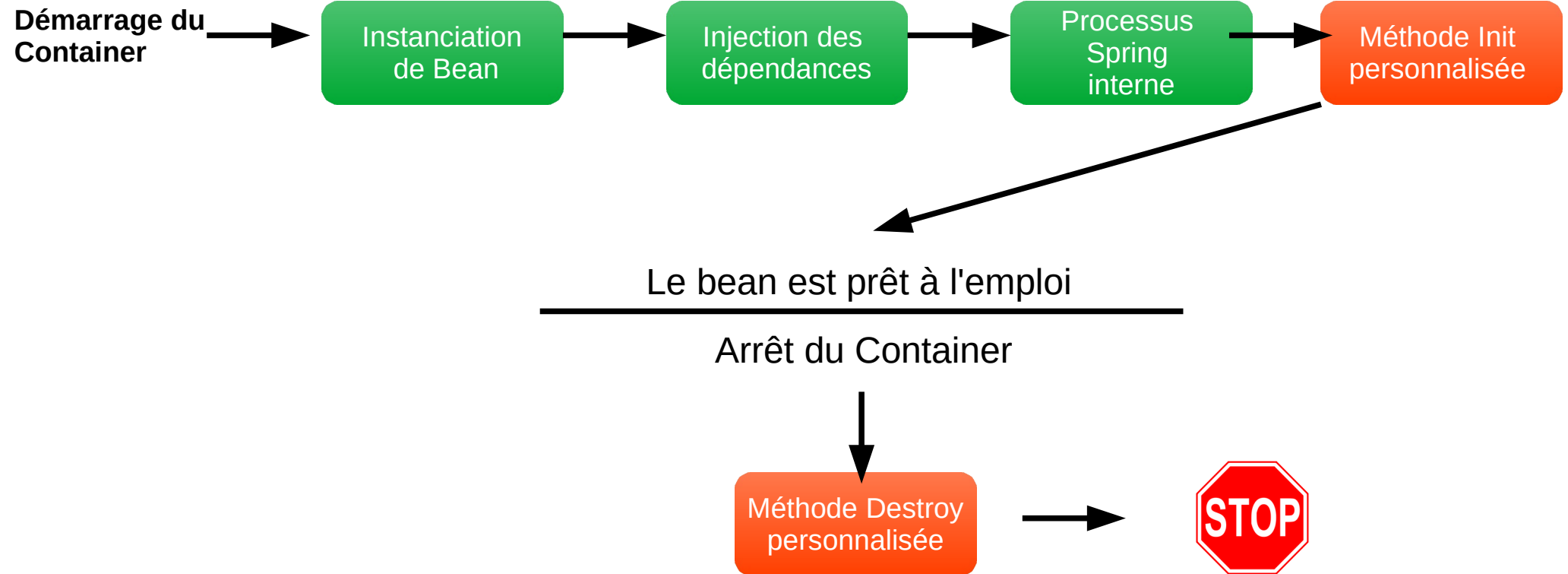
# 02 Spring Core – 02 Xml

04 Bean 02 Lifecycle

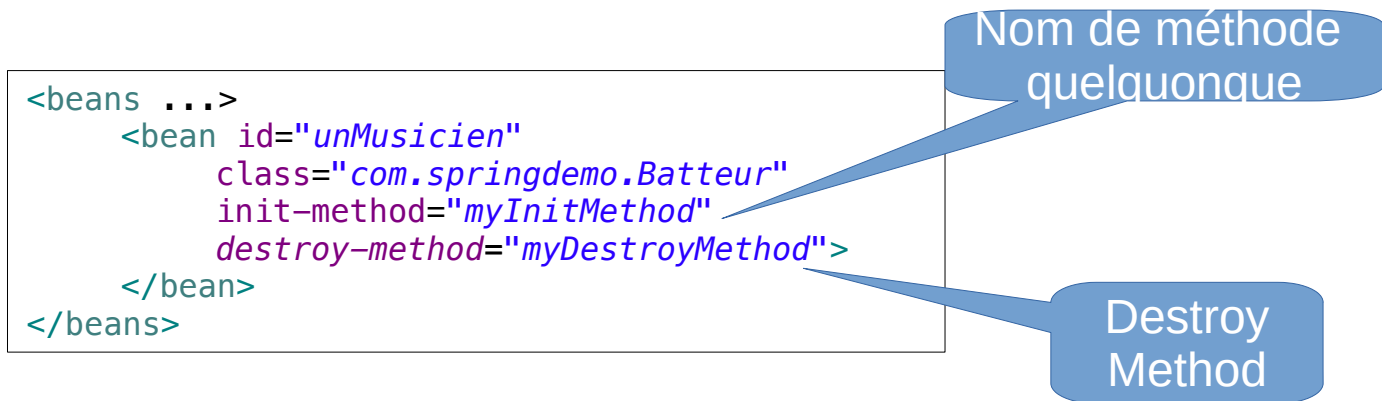
# LifeCycle

- Les beans sont créés sur la base de notre configuration.
- On voit quelles sont les méthodes qui nous permettent d'interagir sur ce cycle de vie des beans.
- Le cycle de vie ce sont les états successifs que connaît un bean , à partir de son l'instanciation, jusqu'à sa destruction

# Lifecycle - diagram



# init-method pour configurer l'initialisation du bean



La signature des méthodes en détail :

- acces modifier (visibilité) : public, protected ou private
- return type : "**void**" est souvent utilisé mais on peut utiliser ce que l'on veut. (car on n'est pas en mesure de capturer l'élément retourné).
- Nom de méthode : au choix
- Argument : aucun argument autorisé (obligatoire).

# demo

- Ajoutons les méthodes à notre musicien Batteur

```
public class Batteur implements Musicien {
    ...etc
    //init-method
    public void myInitMethod() {
        System.out.println("Batteur : à l'intérieur de ma methode myinitMethod");
    }

    //destroy method
    public void myDestroyMethod() {
        System.out.println("Batteur : à l'intérieur de ma methode myDestroyMethod");
    }
    ...}
}
```

- Configurer les méthodes : beanLifecycle-applicationContext.xml

```
<bean id="unMusicien" class="com.springdemo.Batteur"
    init-method="myInitMethod"
    destroy-method="myDestroyMethod">
    <!-- Définir l' Injection par constructeur -->
    <constructor-arg ref="unPrepareService"/>
</bean>
```

# On peut créer un main() dans BeanLifeCycleDemoApp.java

```
package com.springdemo;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class BeanLifeCycleDemoApp {

    public static void main(String[] args) {
        //charger le fichier de configuration xml
        ClassPathXmlApplicationContext context =
            new ClassPathXmlApplicationContext("beanLifeCycle-applicationContext.xml");

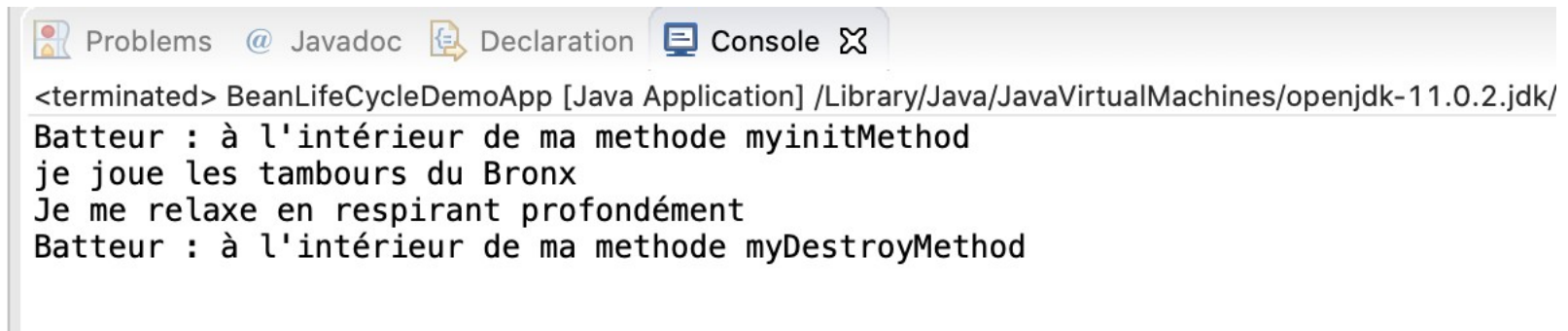
        // accéder au bean géré par le spring container
        Musicien musicien= context.getBean("unMusicien", Musicien.class);

        // utiliser les methode de musicien
        System.out.println(musicien.joueTaPartition());
        System.out.println(musicien.getPrepa());

        // fermer le context
        context.close();
    }
}
```

# Run As > java Application

- Voici la sortie en console qui illustre le cycle de vie du bean "unMusicien" :



The screenshot shows an IDE console window with the following tabs: Problems, Javadoc, Declaration, and Console. The Console tab is active and displays the following output:

```
<terminated> BeanLifeCycleDemoApp [Java Application] /Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/  
Batteur : à l'intérieur de ma methode myinitMethod  
je joue les tambours du Bronx  
Je me relaxe en respirant profondément  
Batteur : à l'intérieur de ma methode myDestroyMethod
```

- Précision sur les objets scopés en prototype