USO DE @BEAN

@BEAN

- Anotación a nivel de método
- Define un bean <bean … />
- En clases
  - **@Configuration: preferible**
  - @Component (y derivados): lite mode
- Atributos: *name*, *init-method*, *destroy-mehtod*, *autowiring*
  - Con este enfoque, seguramente esto se configure vía anotaciones

DECLARACIÓN DE UN BEAN

- Un método anotado @Bean
- El tipo de retorno es el tipo del bean
- El nombre por defecto es el nombre del método.

```
@Bean
public TransferServiceImpl transferService() {
    return new TransferServiceImpl();
}
```

```
<bean id="transferService"
                  class="com.acme.TransferServiceImpl"/>
```

DECLARACIÓN DE UN BEAN

- El tipo de retorno puede ser un interfaz o supertipo de la clase instanciada.

```
@Bean
public TransferService transferService() {
    return new TransferServiceImpl();
}
```

DEPENDENCIAS DE UN BEAN

- Un método @Bean puede recibir cero o más parámetros.
- Los objetos son dependencias del bean definido.
- El contenedor inyectará las mismas (al estilo de la inyección por constructor)

DEPENDENCIAS DE UN BEAN

```
@Configuration
public class AppConfig {

  @Bean
  public TransferService transferService(
                    AccountRepository accountRepository) {
      return new TransferServiceImpl(accountRepository);
    }
}
```

El contenedor se encargará de inyectar la dependencia de tipo *AccountRepository*.

ÁMBITO DE UN @BEAN

- Podemos definir su ámbito a través de anotaciones.
  - *Scope("singleton")*: por defecto
  - *Scope("prototype")*
  - *@RequestScope*
  - *@SessionScope*
  - *@ApplicationScope*

En un contexto web

USO DE @PRIMARY

- Ante varios beans de un tipo, es el primer candidato (*primus inter pares*)
- A nivel de clase (@Component y derivados)
- A nivel de método (@Bean)

USO DE @PRIMARY

```
@Primary
@Component
public class HibernateFooRepository extends FooRepository
{
     public HibernateFooRepository(
                         SessionFactory sessionFactory) {
          // ...
     }
 }
```

USO DE @PRIMARY

```
@Configuration
public class MovieConfiguration {

    @Bean
    @Primary
    public MovieCatalog firstMovieCatalog() { ... }

    @Bean
    public MovieCatalog secondMovieCatalog() { ... }

    // ...
}
```

INYECCIÓN DE VALORES @VALUE

- ¿Cómo podemos inyectar valores de tipo primitivo (por ejemplo, String)?
- @Value
  - Uso de ficheros de properties
  - Variables de entorno
  - Valores por defecto
- En realidad, podemos inyectar valores en otros tipos: wrapper, List, ...

INYECCIÓN DE VALORES @VALUE

```
mensaje=Hola a todos desde un fichero de propiedades!
```

```
@Component
public class Saludator {

     @Value("${mensaje}")
     private String mensaje;

     public String saludo() {
          return mensaje;
     }

}
```