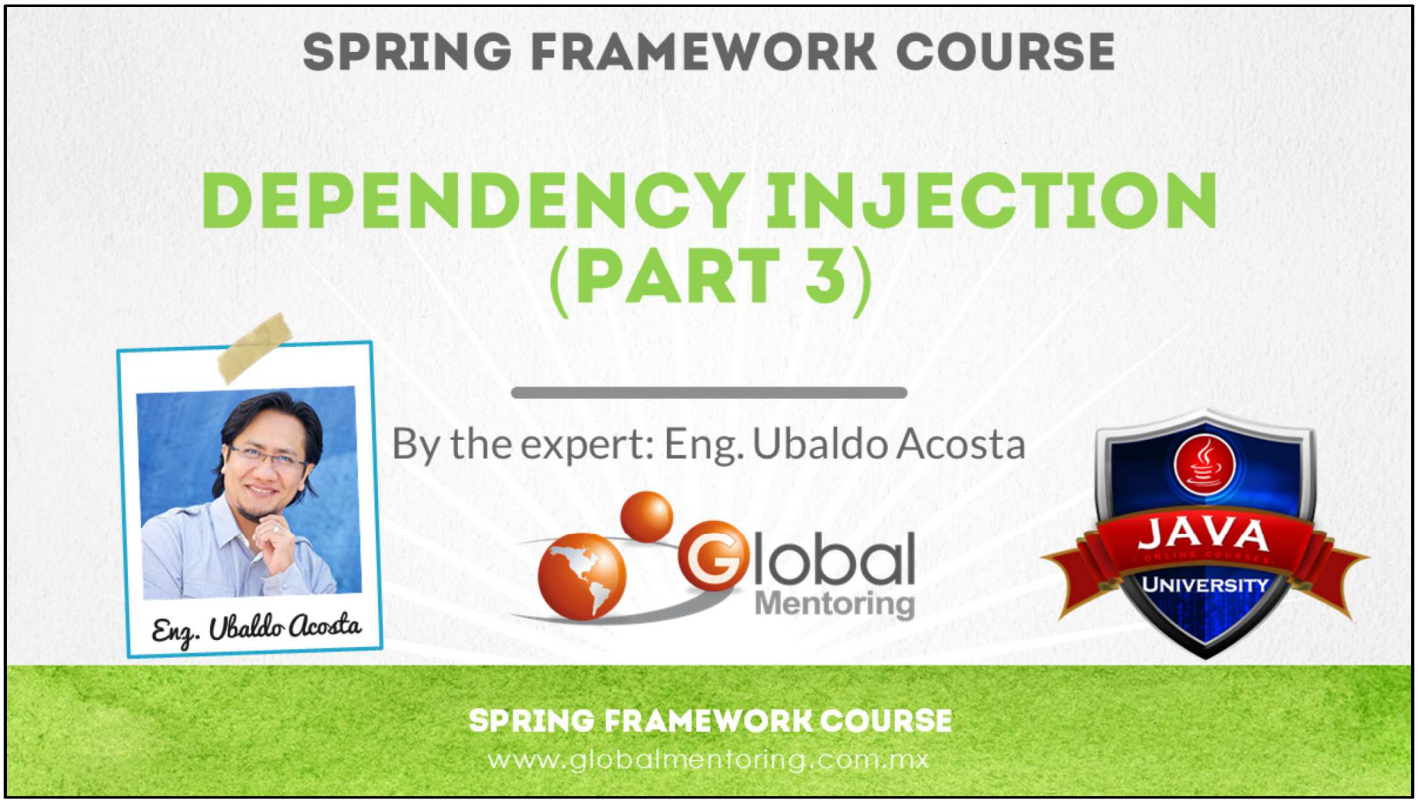
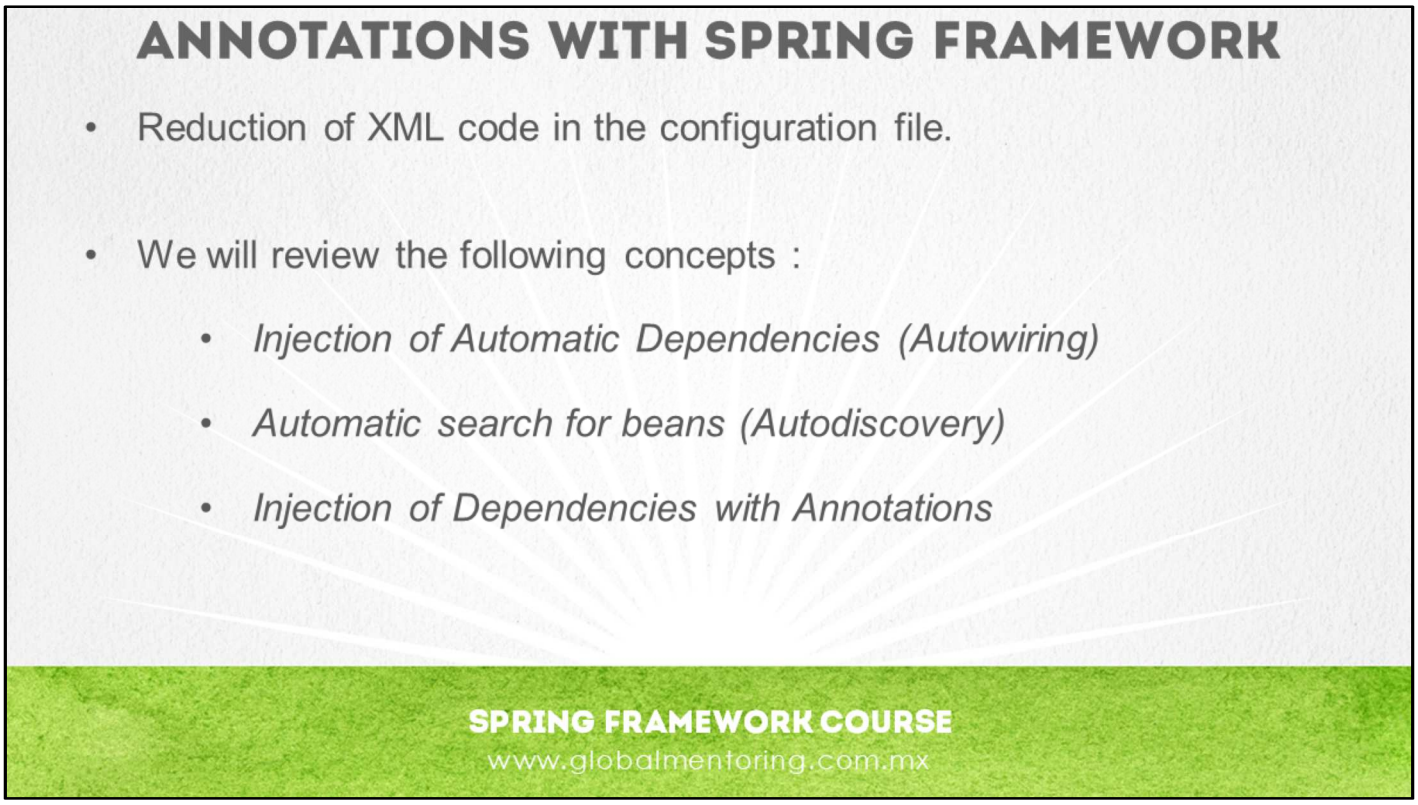


Hello, Ubaldo Acosta greets you. Welcome again.

We are going to continue the topic of Injection of Dependencies with Spring Framework.

Are you ready? OK let's go!



So far we have studied how to perform injections using the configuration file with Spring. We will now review how to minimize the XML configuration file using the concept of annotations when declaring Spring beans.

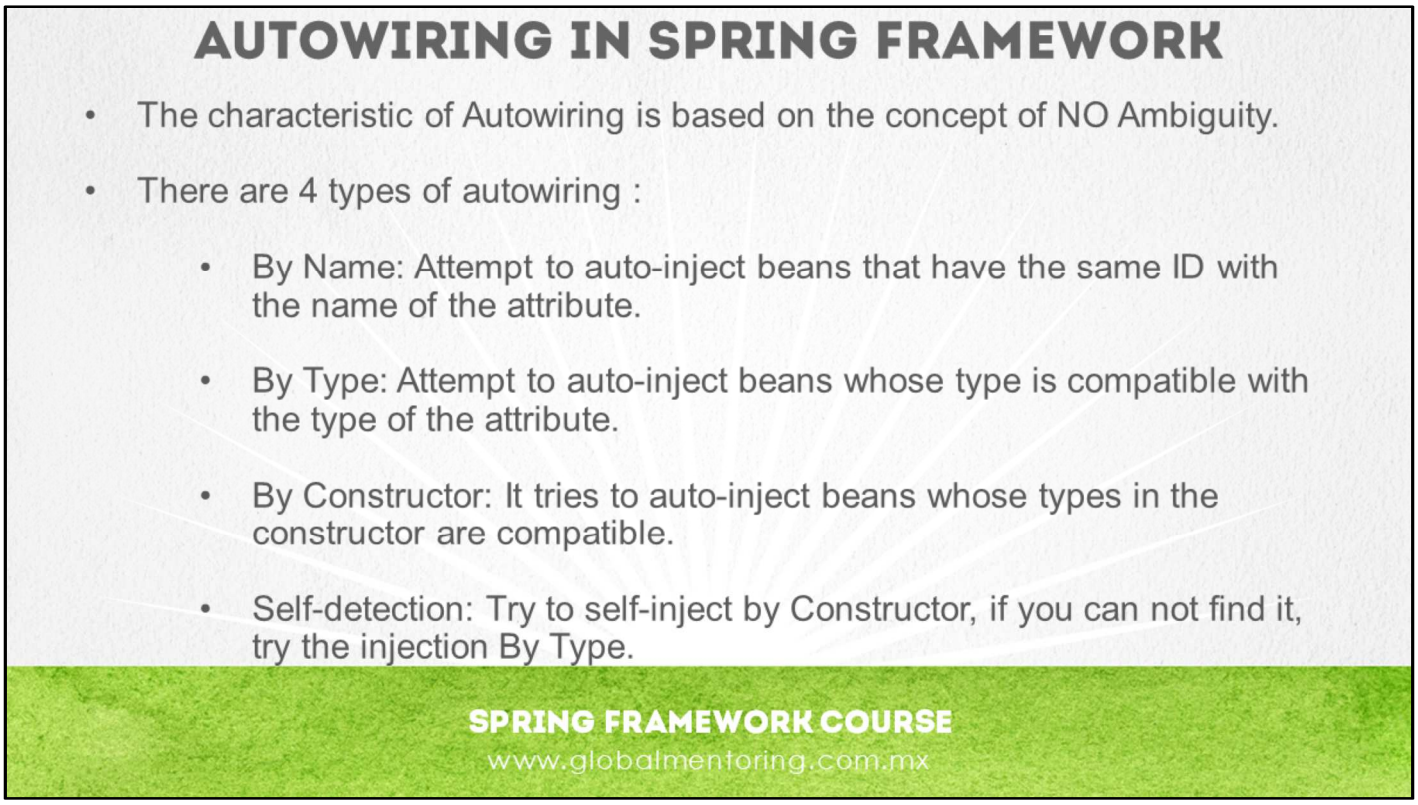
We will review the following concepts:

- Automatic Dependency Injection (autowiring): Reduces the need for the <property> and <constructor-arg> tags in the declaration of the beans.

- Automatic Bean Search (Autodiscovery): In this case, Spring tries to detect which Java classes will be used as beans, which allows the <bean> reduction

- Injection of Dependencies with Annotations: By combining autowiring with autodiscovery we are practically eliminating the need to declare our beans in the Spring configuration file.

Next we will review these characteristics.



The concept of Autowiring is based on NO Ambiguity, for example, if we only have one DataSource object in our application, then there is no ambiguity about which bean it is that must be injected.

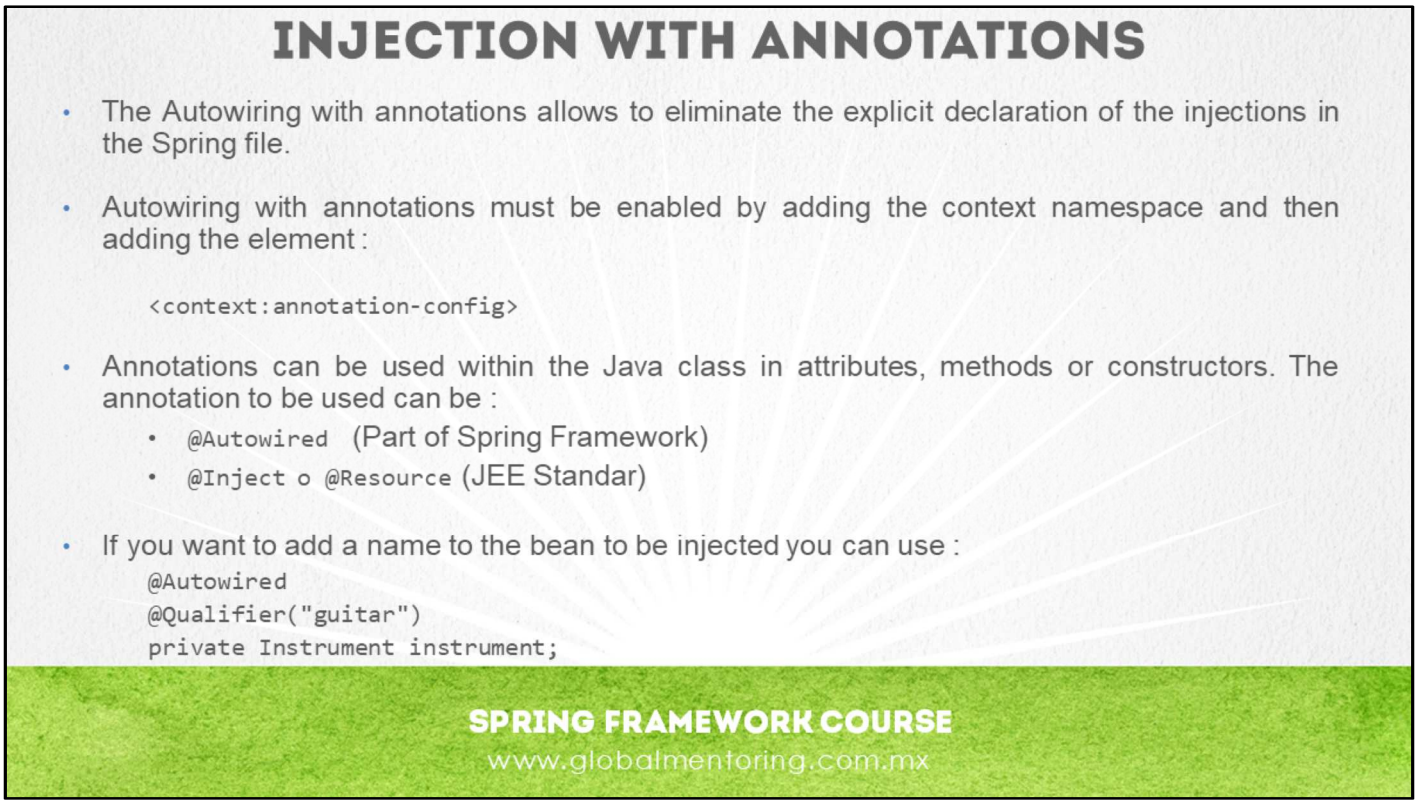
Taking advantage of this context and the non-ambiguity when injecting the dependencies Spring will automatically make the dependency injection based on 4 types of autowiring. Here is an example of each:

- By Name: In case the name of the bean (DI) matches the name of the attribute of the Java class, then Spring can infer that this dependency must be injected automatically (autowire = "byName").

- By Type: In this case, what is revised is that the types of Spring beans compatible with the types of properties to be injected. If there are several compatible beans, you can specify, in the declaration of the Bean, if it is primary (primary = "true") or if it should be discarded as a candidate to be autoinjected (autowire-candidate = "false"). The autowire = "byType" attribute is used

- By Constructor: It tries to auto-inject beans whose types in the constructor are compatible. In the declaration of the bean, autowire = "constructor" is used

- Self-detection: Try to self-inject by Constructor, if you can not find it, try the injection By Type. The autowire = "autodetect" attribute is used



In the previous sheet, although part of the bean configuration is removed, there is still a lot of XML code left. For this the issue of autowiring with Annotations arises.

It is possible to use the concept of Autowiring using annotations in our Java class.

Once the <context: annotation-config> element has been enabled in the Spring configuration file, it is possible to avoid the explicit configuration of dependency injection in the beans, so that the <constructor-arg> and <property> elements are practically eliminate. Only the definition of the bean remains in itself.

The @Inject annotation, unlike the annotation @Autowire, belongs to the JSR-330 specification, which is used inside a JEE container (Java Enterprise) such as Glassfish, JBoss, Weblogic, etc.

One of the advantages of using the annotation @Autowire instead of @Inject, is that we do not need a Java application server, but a Java Web Server, such as Apache Tomcat or Jetty.

However, if we are using technology such as EJB, JPA and JSF it might be more convenient to use the @Inject annotation to have compatibility with the Java business standard and the Spring framework.

AUTODISCOVERING WITH SPRING

- The concept of Autodiscovering allows practically eliminating the need to declare the Bean in the Spring file.
- The Autodiscovering must be enabled by adding the context namespace and then adding the element:

```
<context:component-scan>
```

- This element allows you to perform the same task as <context: annotation-config>, but with more features.
- The annotations to use in the Java class can be :
  - `@Component` – It is the most general
  - `@Controller` – Indicates that it is a class of the presentation layer (web layer)
  - `@Service` – Indicates that it is a class of Business Logic
  - `@Repository` – Indicates that it is a class of the data layer

SPRING FRAMEWORK COURSE
www.globalmentoring.com.mx

In the previous sheet, although part of the bean configuration is removed, it is still necessary to declare it in the Spring file.

With the concept of Autodiscovering it is possible to eliminate practically all the definition of the bean from the Spring file, and through the use of annotations, indicate the way in which the Java class will be added as a bean to the Spring container.

There are different annotations, depending on the role of the class in a 3-layer architecture. The most generic annotation is @Component, however if we want to be more explicit and self-document our Java code, we can use the following annotations for each JEE layer.

Presentation layer (with Spring MVC): @Controller

Service Layer: @Service

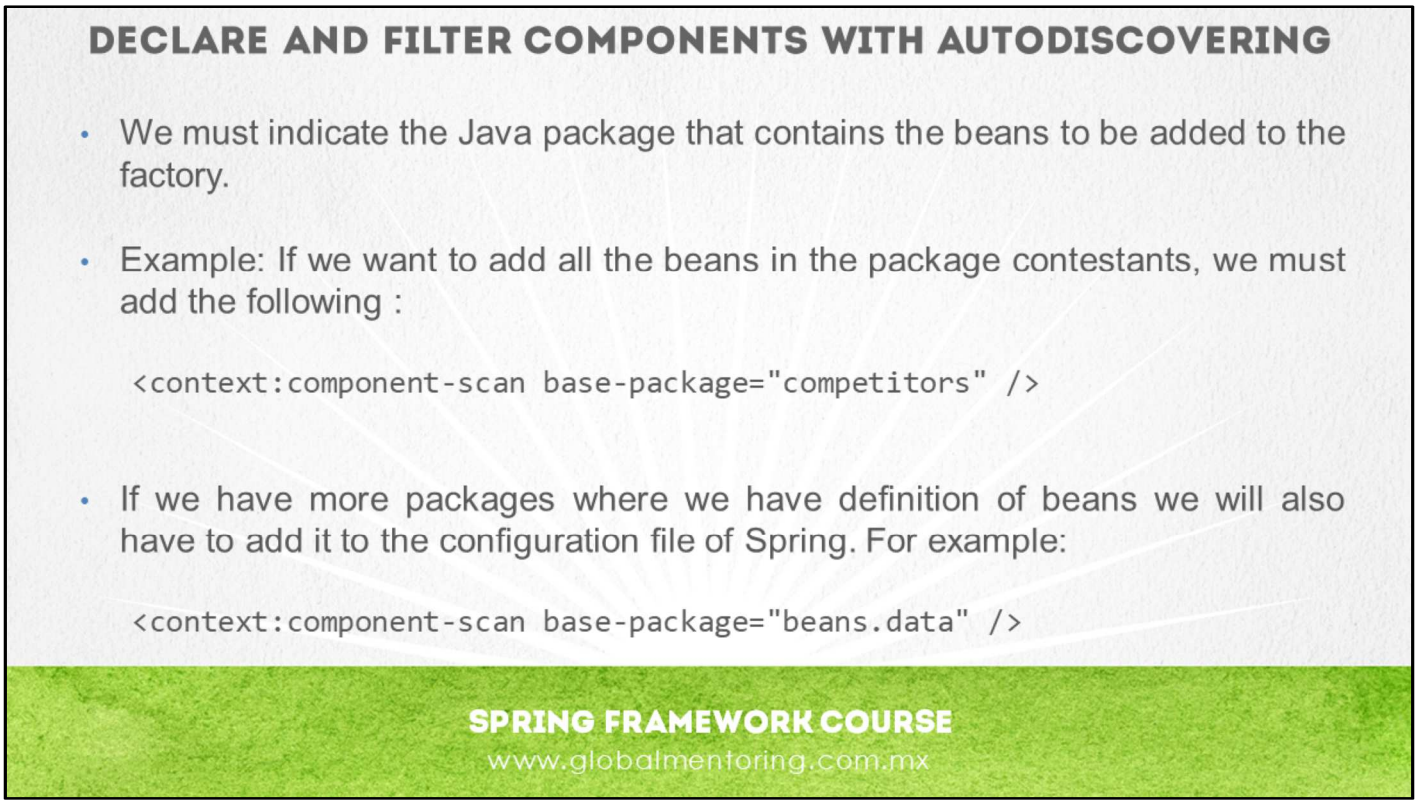
Data Layer: @Repository

If we want to modify the name of the bean that is added to the repository, we must use the following code:

```
@Component ("guitar")
public class Guitar implements Instrument {...}
```

The above is equivalent to the classic statement:

```
<bean id = "guitar" class = "instruments.Guitar" />
```



The <context: component-scan> element is quite convenient and flexible to find the candidate beans to add to the Spring container.

We can add all the packages that are necessary, and just by indicating the Java root package, it is enough for Spring to search recursively in the subpackages.

In order for Spring to search for the beans, we must indicate the Java package to be added to the factory.

- Example: If we want to add all the beans in the package contestants, we must add the following:

```
<context:component-scan base-package="competitors" />
```

- If we have more packages where we have definition of beans we must also add it to the Spring configuration file. For example:

```
<context:component-scan base-package="beans.data" />
```

Then we will perform an exercise to put these concepts into practice.


ONLINE COURSE
SPRING FRAMEWORK
By: Eng. Ubaldo Acosta
Global Mentoring
SPRING FRAMEWORK COURSE
www.globalmentoring.com.mx