loose-coupling
tight-coupling
spring-beans
ApplicationContext
@Bean
@Configuration
dependency-injection
java-configuration
programming-against-interface
POJO
AnnotationApplicationContext
single-responsibility-principle

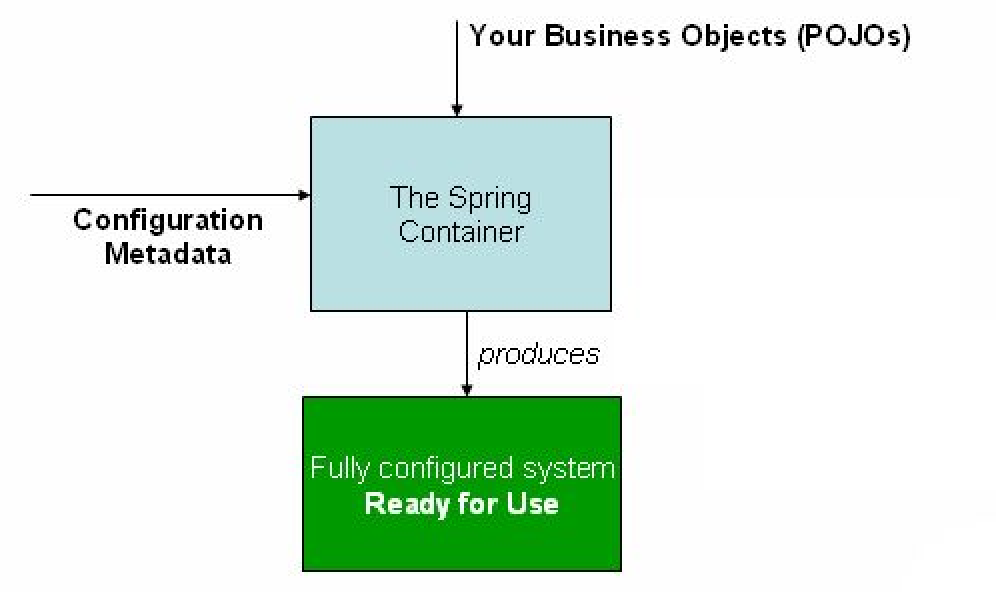
Dependency Injection / Inversion of Control

- The **Spring Framework** is an application framework and inversion of control container for the Java platform. *https://en.wikipedia.org/wiki/Spring_Framework*

- *Dependency Injection* implies that clients delegate the dependency resolution to an external service. The client is not allowed to call the injector service, which is why this software pattern is characterized by *Inversion of Control* behavior, also known as the *Don't call us, we'll call you!* principle. - *Pivotal Certified Professional Spring Developer Exam: A Study Guide*

- In software engineering, **dependency injection** is a technique whereby one object supplies the dependencies of another object. A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it. The service is made part of the client's state.[1] Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern. https://en.wikipedia.org/wiki/Dependency_injection

Spring IoC Container



POJO - Single Responsibility Principle

- **plain old Java object** (**POJO**) is an ordinary Java object, not bound by any special restriction and not requiring any class path https://en.wikipedia.org/wiki/Plain_old_Java_object
- POJOs, simple Java objects each with a single responsibility - *Pivotal Certified Professional Spring Developer Exam: A Study Guide*
- The **single responsibility principle** is a computer programming principle that states that every module or class should have responsibility over a single part of the functionality provided by the software, and that responsibility should be entirely encapsulated by the class. All its services should be narrowly aligned with that responsibility. Robert C. Martin expresses the principle as, "A class should have only one reason to change."[*https://en.wikipedia.org/wiki/Single_responsibility_principle*

Java Based Configuration

```java
@Configuration
public class AppConfig {

   @Bean
   public ITeam crvenaZvezda() {
       return new CrvenaZvezda();
   }

   @Bean
   public ITeam partizan() {
       return new Partizan();
   }

   @Bean
   public IGame game() {
       return new Game(crvenaZvezda(), partizan());
   }
}
```

loose-coupling
tight-coupling
spring-beans
ApplicationContext
@Bean
@Configuration
dependency-injection
java-configuration
programming-against-interface
POJO
AnnotationApplicationContext
single-responsibility-principle