

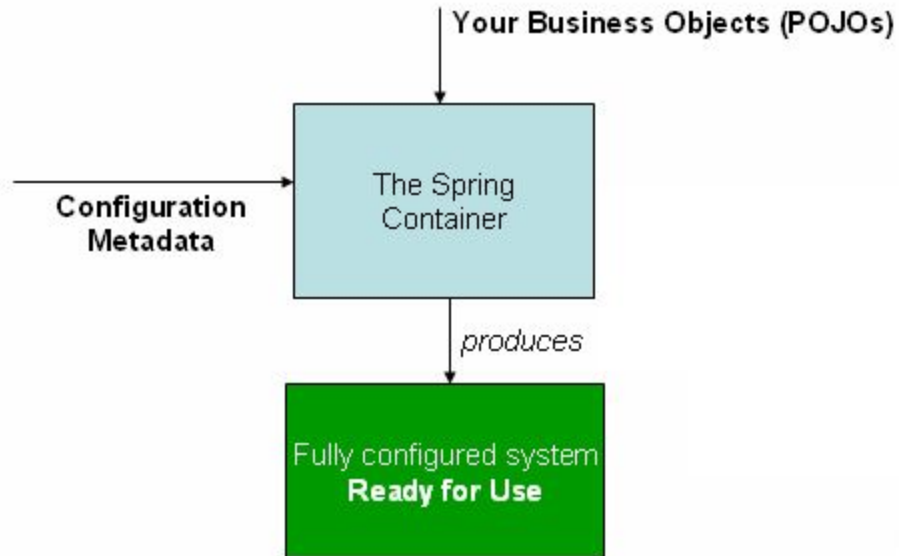
A word cloud featuring various Spring framework concepts. The words are arranged in a diagonal pattern from the top-left to the bottom-right. The largest word is 'dependency-injection' in light blue. Other prominent words include 'spring-beans' in pink, 'ApplicationContext' in purple, '@Bean' in green, and '@Configuration' in orange. Smaller words include 'loose-coupling', 'tight-coupling', 'programming-against-interface', 'POJO', 'AnnotationApplicationContext', 'single-responsibility-principle', and 'java-configuration'.

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

Dependency Injection / Inversion of Control

- The **Spring Framework** is an [application framework](https://en.wikipedia.org/wiki/Application_framework) and [inversion of control container](https://en.wikipedia.org/wiki/Inversion_of_control_container) for the [Java platform](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Service_(computer_programming))). An injection is the passing of a dependency to a dependent object (a [client](https://en.wikipedia.org/wiki/Client_(computer_programming))) that would use it. The service is made part of the client's [state](https://en.wikipedia.org/wiki/State_(computer_programming)).^[1] Passing the service to the client, rather than allowing a client to build or [find the service](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/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."¹ https://en.wikipedia.org/wiki/Single_responsibility_principle

Java Based Configuration

@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());  
}  
}
```

A word cloud featuring various Spring framework concepts. The words are arranged in a diagonal pattern from the top-left to the bottom-right. The largest word is 'dependency-injection' in light blue. Other prominent words include 'spring-beans' in pink, 'ApplicationContext' in purple, '@Bean' in green, and '@Configuration' in orange. Smaller words include 'loose-coupling', 'tight-coupling', 'programming-against-interface', 'POJO', 'AnnotationApplicationContext', 'single-responsibility-principle', and 'java-configuration'.

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