Spring Beans and Dependency Injection

You are free to use any of the standard Spring Framework techniques to define your beans and their injected dependencies. We generally recommend using constructor injection to wire up dependencies and `@ComponentScan` to find beans.

If you structure your code as suggested above (locating your application class in a top package), you can add `@ComponentScan` without any arguments or use the `@SpringBootApplication` annotation which implicitly includes it. All of your application components (`@Component`, `@Service`, `@Repository`, `@Controller`, and others) are automatically registered as Spring Beans.

The following example shows a `@Service` Bean that uses constructor injection to obtain a required `RiskAssessor` bean:

Java Kotlin

```java
@Service
public class MyAccountService implements AccountService {
```

Spring Boot / Reference / Developing with Spring Boot / Spring Beans and Dependency Injection

Edit this Page
GitHub Project
Stack Overflow

```java
        this.riskAssessor = riskAssessor;
    }

    // ...

}
```

If a bean has more than one constructor, you will need to mark the one you want Spring to use with `@Autowired`:

Java Kotlin

```java
@Service
public class MyAccountService implements AccountService {

    private final RiskAssessor riskAssessor;

    private final PrintStream out;

    @Autowired
    public MyAccountService(RiskAssessor riskAssessor) {
        this.riskAssessor = riskAssessor;
        this.out = System.out;
    }

    public MyAccountService(RiskAssessor riskAssessor, PrintStream out) {
        this.riskAssessor = riskAssessor;
        this.out = out;
    }

    // ...

}
```

TIP

Notice how using constructor injection lets the `riskAssessor` field be marked as `final`, indicating that it cannot be subsequently changed.

Prev
Auto-configuration

Next
Using the @SpringBootApplication Annotation