**@Component, @Repository, @Controller и @Service**

**Сходство**

Еще один момент, на который стоит обратить внимание: **автоматическое обнаружение сканирования и внедрение зависимостей для BeanDefinition** все эти аннотации (а именно, @Component, @Service, @Repository, @Controller) одинаковы. **Мы можем использовать одно вместо другого и все еще обойтись.**

**Различия между @Component, @Repository, @Controller и @Service**

**@Составная часть**

Это универсальная аннотация стереотипа, указывающая, что класс является компонентом Spring.

**Что особенного в @Component**  
<context:component-scan> сканирует только @Component и вообще не ищет @Controller, @Service и @Repository. Они сканируются, потому что сами @Component.

Просто взгляните на определения аннотаций @Controller, @Service и @Repository:

@Component

public @interface Service {

….

}

@Component

public @interface Repository {

….

}

@Component

public @interface Controller {

…

}

Таким образом, нет ничего плохого в том, что @Controller, @Service и @Repository являются специальными типами аннотации @Component. <context:component-scan> подхватывает их и регистрирует их следующие классы как @Component, как если бы они были аннотированы @Component.

Аннотации специальных типов также сканируются, потому что они сами @Component аннотацией @Component, что означает, что они также являются @Component s. Если мы определим нашу собственную пользовательскую аннотацию и аннотируем ее с помощью @Component, она также будет сканироваться с помощью <context:component-scan>

**@Repository**

Это указывает на то, что класс определяет хранилище данных.

**Что особенного в @Repository?**

В дополнение к указанию, что это конфигурация на основе аннотаций, задание @Repository должно перехватывать специфичные для платформы исключения и повторно @Repository их как одно из унифицированных непроверенных исключений Springs. Для этого были предоставлены PersistenceExceptionTranslationPostProcessor, которые мы должны добавить в контекст приложения Springs, например так:

<bean class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor"/>

Этот @Repository бина добавляет советник к любому @Repository аннотированному @Repository так что любые специфичные для платформы исключения перехватываются и затем перебрасываются как одно из неконтролируемых исключений доступа к данным Springs.

**@Controller**

Аннотация @Controller указывает, что определенный класс выполняет роль контроллера. Аннотация @Controller действует как стереотип для аннотированного класса, указывая на его роль.

**Что особенного в @Controller?**

Мы не можем поменять эту аннотацию на любую другую, такую как @Service или @Repository, даже если они выглядят одинаково. Диспетчер просматривает классы, аннотированные @Controller и обнаруживает в них аннотации @RequestMapping. Мы можем использовать @RequestMapping только в тех классах, которые аннотированы @Controller.

**@Обслуживание**

@Service содержат бизнес-логику и вызывают методы на уровне хранилища.

**Что особенного в @Service?**

Помимо того факта, что он имел обыкновение указывать, что он содержит бизнес-логику, в этой аннотации нет ничего более заметного; но кто знает, spring может добавить некоторые дополнительные исключительные в будущем.

**Что-то еще?**

Как и выше, в будущем Spring может добавить специальные функции для @Service, @Controller и @Repository с их соглашениями о размещении. Следовательно, всегда полезно соблюдать соглашение и использовать его в соответствии со слоями.