В xml конфиге описываюит конфигурации бинов

Типы того, что мы можем использовать в настройке бина

id="pare" айди бина

class="sping.task1.classes.pareImpl.PareImpl" Указатель на класс бина

scope = "prototype"Как бин будет реагировать на запрос создания бина

autowire

no: без автоматического связ ывания

byname: будет смотреть по сеттерам полей и автоматически подставлять бины

byType: так же только поиск не по названию, а по типу

constructor: Режим конструктора внедряет зависимость, вызывая конструктор класса.

Autodetect: В этом режиме Spring сначала пытается выполнить автоматическое подключение с помощью конструктора. Если это не удается, он пытается автоматически подключиться с помощью byType

Связать бины между собой можно следующими способами:

Property

Setter

Constructor

autowire-candidate - По умолчанию выполняется автоматическое сканирование и сопоставляются все определения компонентов в области видимости. Если вы хотите исключить некоторые определения bean-компонентов, чтобы их нельзя было внедрить в режиме автопроводки, вы можете сделать это, используя для параметра «autowire - андидата» значение false

Другой вариант - ограничить число кандидатов на автопроводку на основе сопоставления с образцом по отношению к именам бинов. Элемент верхнего уровня <beans /> принимает один или несколько шаблонов в своем атрибуте default-autowire-sources.

For example, to limit autowire candidate status to any bean whose name ends with ‘Impl’, provide a value of ‘\*Impl’. To provide multiple patterns, define them in a comma-separated list.

Описывание конфигов с помощью аннотаций

Преимущества использования:

Короче по сравнению с XML аннотацией

Удобнее чем XML аннотация (кому как)

Код становится более читабельным

Как работает:

Spring сканирует все классы и находит те, которые отмечены специальными аннотациями @Component и создает их бины.

При указании аннотации @Component можно так же указать id, если этого не сделать он будет генерироваться автоматически исходя из названия класса с маленькой буквы.

Для использования все равно потребуется xml файл но, в нем мы указываем

<**context:component-scan base-package="sping.task1.classes"**/>

Описывая где конкретно следует искать спринг бины.

Для того чтобы внедять зависимости есть аннотация @Autowared

Как он работает:

Spring сканирует все классы с аннотацией Component и создает бины для этих классов

Spring сканирует все созданные бины и проверяет подходит ли хотя бы один из них в качестве зависимости в месте, где указана аннотация @Autowared. Если находит, то он внедряется в качестве зависимости. Если не находит ни одного бина ошибка, если несколько – неоднозначность.

аннотация @Autowared может использоваться на полях, сеттерах и конструкторах

если на конструкцию (спринг попытается внедрить все поля в которые сможет). Аннотация внедряется в приватное поле с помощью рефлексии.

Неоднозначность бинов – ситуация когда несколько бинов реализуют один интерфейс. Решается эта ситуацию можно так же решить с помощью аннотации @Qualifier в скобках которой указывается id того бина которого конкретно мы хотим использовать. Можно использовать так же как @Autowared

!В случае с конструктором используется следующая структура

@Autowared

Constructor(@Qualifier(“bean1”) bean ObjectImpl1, @ Qualifier(“bean2”) bean ObjectImpl2)