Снова тест :)

1. Что такое view? А materialized view?
2. В чем минусы использования индексов?
3. Опишите основные этапы обработки запроса в postgresql.
4. Для чего может использоваться нормализация? А денормализация?
5. Что такое теорема CAP?

DIGITAL HABITS
EDUCATION

Spring Core

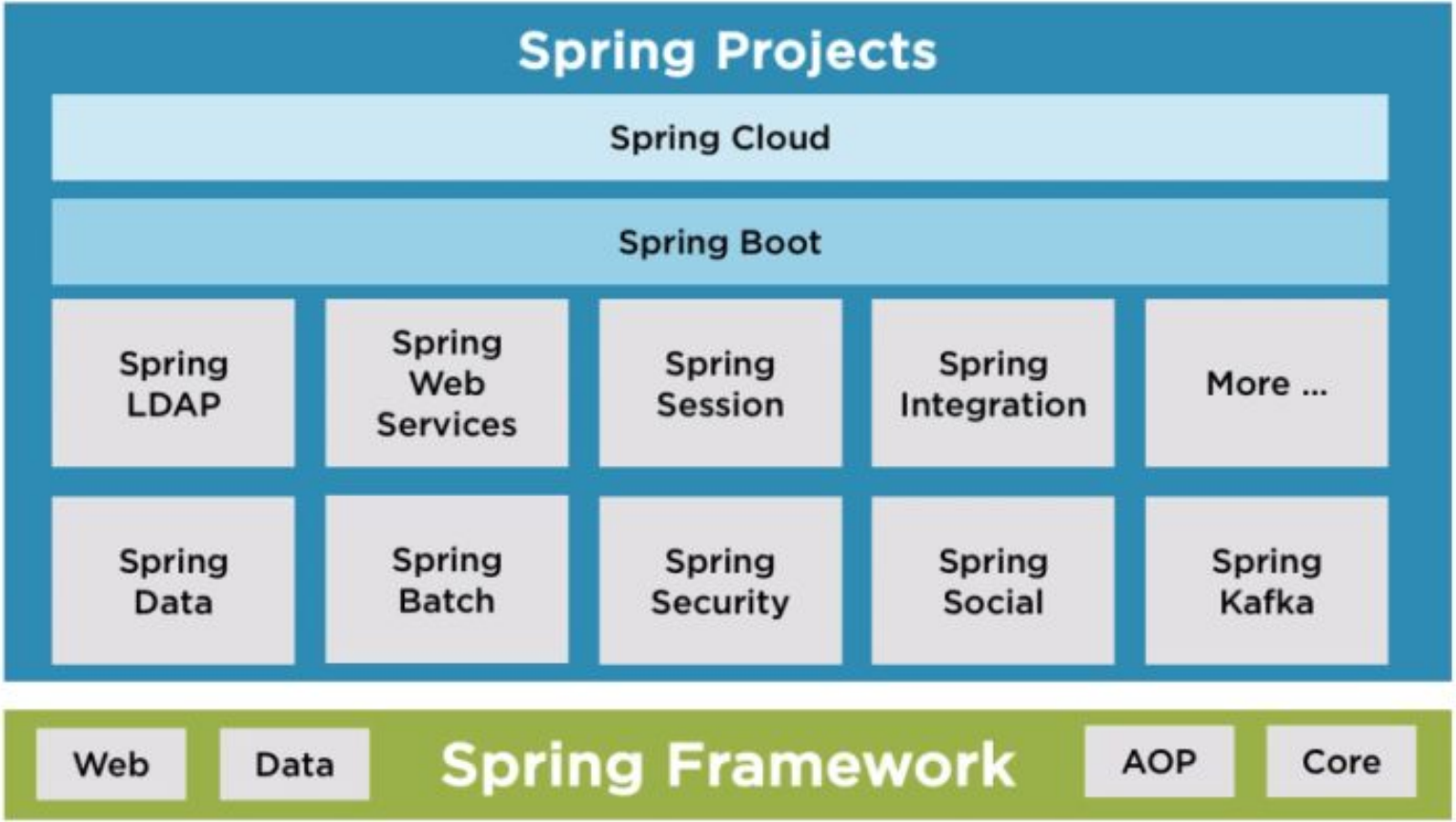
>_ java

Agenda

- Spring IoC
- Bean’s scope
- BeanPostProcessing
- Annotations
- и другое

```
>_  java
```

Spring Framework

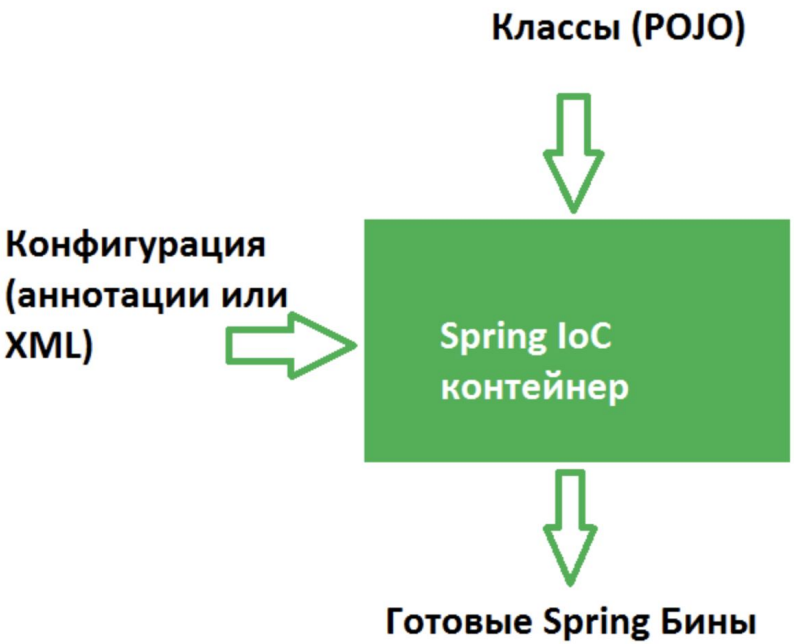


Inversion of Control

Инверсия управления (Inversion of Control) - один из популярных принципов объектно-ориентированного программирования, при помощи которого можно снизить связанность между компонентами, а также повысить модульность и расширяемость ПО.

Beans

Bean (бин) – это java объект, который управляется с помощью Spring IoC контейнера.



Dependency Injection

Внедрение зависимостей (DI - Dependency Injection) - это инициализация полей бинов другими бинами (зависимостями).

Виды контейнеров в Spring

В Spring имеется 2 различных вида контейнеров:

BeanFactory

org.springframework.beans.factory.BeanFactory

ApplicationContext

org.springframework.context.ApplicationContext

Виды контейнеров в Spring

ClassPathXmlApplicationContext

Загружает определение контекста из XML-файла, расположенного в библиотеке классов (classpath), и обрабатывает файлы с определениями контекстов как ресурсы

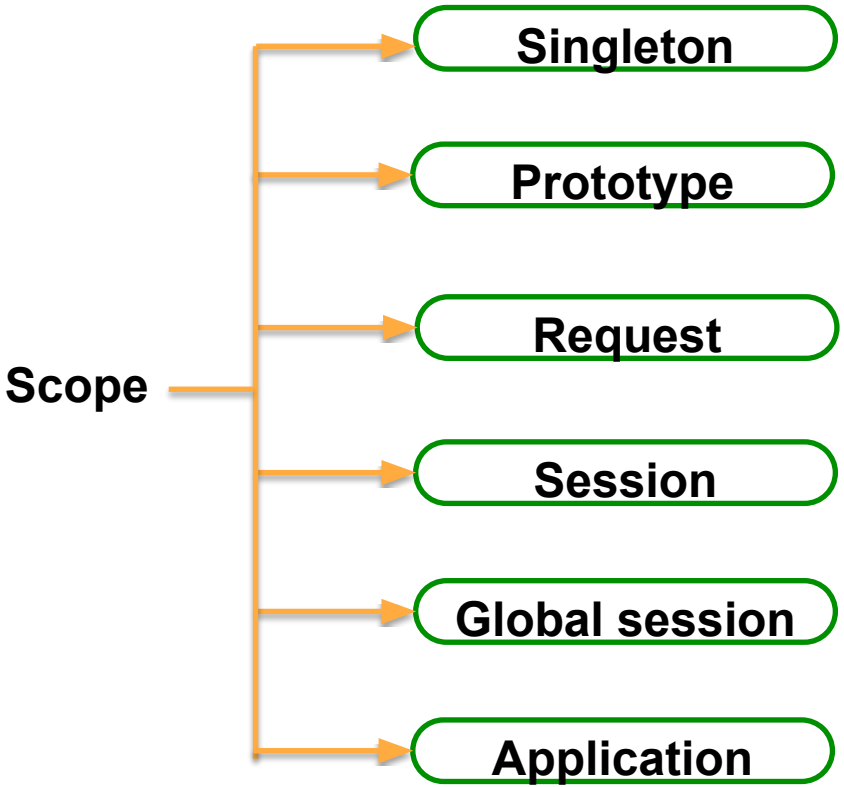
FileSystemXmlApplicationContext

Загружает определение контекста из XML-файла в файловой системе

XmlWebApplicationContext

Загружает определение контекста из XML-файла, содержащегося внутри веб-приложения

Жизненный цикл бинов (scope)



Создается только один экземпляр данного класса (используется по-умолчанию).

Каждый раз по требованию создается новый экземпляр класса.

Жизненный цикл экземпляра ограничен единственным HTTP запросом. Для каждого нового HTTP запроса создается новый экземпляр класса.

Жизненный цикл экземпляра ограничен в пределах одной и той же HTTP сессии.

Жизненный цикл экземпляра ограничен в пределах глобальной HTTP сессии.

Жизненный цикл экземпляра ограничен в пределах ServletContext. Действует, только если вы используете web-aware ApplicationContext

Как все работает?

Что делает разработчик:

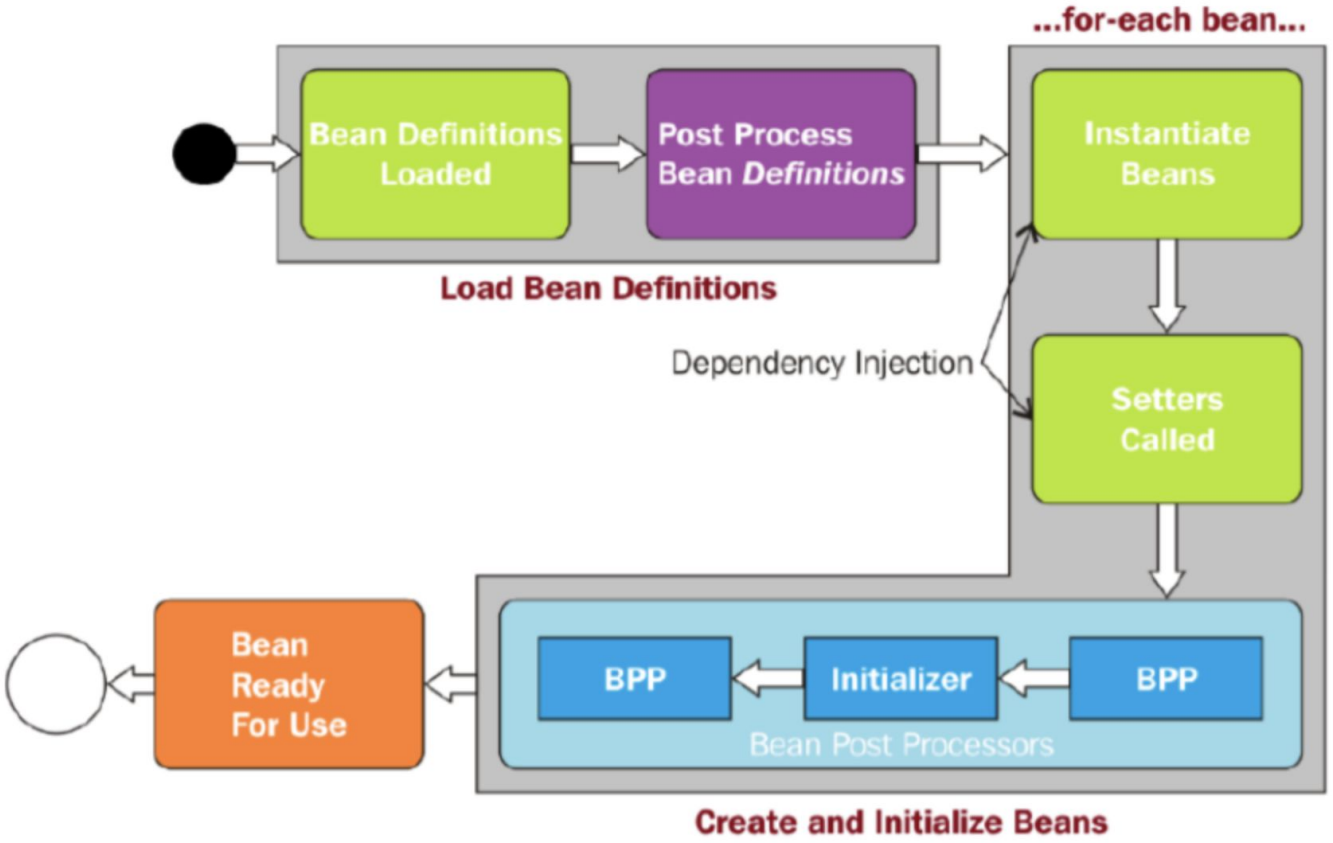
- ➔ Прописывает бины в XML

Как все работает?

Что делает Spring при запуске приложения:

- ➔ BeanDefinitionReader считывает декларации бинов из xml
- ➔ Создаются BeanDefinitions (map)
- ➔ BeanFactory создает бины по описанию BeanDefinitions
  - ◆ если бин singlethon, то кладет в контейнер
  - ◆ prototype не хранит
- ➔ Готовый контекст

Как все работает?



BeanPostProcessor

- Позволяет настраивать бины до того, как они попадут в контейнер
- У этого интерфейса есть 2 метода :
  - postProcessBeforeInitialization
  - postProcessAfterInitialization
- Между ними вызывается init() метод

Как все работает?

- ➔ Init method
  - init-method
  - InitializingBean + afterPropertiesSet()
  - @PostConstruct
- ➔ Destroy method
  - destroy-method
  - DisposableBean + destroy()

Proxy class

- ➔ Имплементировать класс
  - Dynamic Proxy
- ➔ Наследовать класс
  - CG lib

Аннотации

@Component - говорит о том, что класс является источником определения бинов.

@Service - то же самое, что и @Component.

Аннотации

@Configuration - эта аннотация, прописанная перед классом, означает, что класс может быть использован контейнером Spring IoC как конфигурационный класс для бинов.

@Bean - аннотация, прописанная перед методом, информирует Spring о том, что возвращаемый данным методом объект должен быть зарегистрирован, как бин.

Аннотации

@Autowired - применяется для внедрения бина в другой бин (предпочтительней ставить над конструктором или сеттерами).

@Qualifier - указывает конкретный бин для внедрения, если их несколько (бинов с одинаковым типом).

Аннотации

@PostConstruct - метод, помеченный данной аннотацией вызывается Spring после того, как внедрены все бины-property. Используется для дополнительной настройки бина, если требуется, например, считать из прочерти значение и проинициализировать по нему другой атрибут класса.