**EntityManager в Spring**

Итак, каким же образом инициализируется EntityManager в Spring?

Начнём по-порядку. Spring читает конфигурацию, в которой должен быть указан реализующий класс **AbstractEntityManagerFactoryBean**. Существует их всего два: **LocalContainerEntityManagerFactoryBean** и **LocalEntityManagerFactoryBean**. Как несложно догадаться, это абстрактная фабрика.

Первый класс, **LocalContainerEntityManagerFactoryBean**, реализует ORM контейнер по спецификации JPA и предполагает установку соответсвующего провайдера, который будет заниматься реализацией ORM.

Второй, **LocalEntityManagerFactoryBean**, реализует нативный интерфейс **EntityManager**, что в свою очередь предполагает работу с нативным SQL и данными соответсвенно. Мы же будем рассматривать первый вариант.

Итак, создание менеджера сущностей происходит примерно таким образом:

@Bean

**public** LocalContainerEntityManagerFactoryBean **entityManagerFactory**() {

LocalContainerEntityManagerFactoryBean entityManagerFactoryBean = **new** LocalContainerEntityManagerFactoryBean();

entityManagerFactoryBean.setDataSource(dataSource());

entityManagerFactoryBean.setPersistenceProviderClass(HibernatePersistenceProvider.class);

entityManagerFactoryBean.setPackagesToScan("my.package.domain");

entityManagerFactoryBean.setJpaProperties(hibernateProperties());

**return** entityManagerFactoryBean;

}

Как можем видеть, мы реализуем одну из фабрик — **LocalContainerEntityManagerFactoryBean**

Так как мы указали анотацию @Bean, Spring зарустит механизм инициализации этого бина. При каждой инициализации, вызывается метод *afterPropertySet*, который определён в интерфейсе **InitializingBean**, который так же имплементируется в **AbstractEntityManagerFactoryBean**, в последствии и в **LocalContainerEntityManagerFactoryBean**, так как это же абстрактна фабрика, не забыли?

Вобщем инициализация фабрики **EntityManager** начинается с этого метода *afterPropertySet*. Реализующий метод располагается в **AbstractEntityManagerFactoryBean**. В этом методе происходят различные проверки, инициализации локальных полей, а также реализацией выбранного провайдера, ORM или Native Query. Так как мы выбрали ORM путь, то наш путь ещё не закончен, как в случае с **LocalEntityManagerFactoryBean**. В нашем случае, мы передаём управление инициализацией пакету Hibernate, который использую указанный нами класс **HibernatePersistenceProvider** вызовет реализацию **SessionFactory** в **EntityManagerFactoryBuilderImpl**

Но на этом класс **LocalContainerEntityManagerFactoryBean** не останавливается, после всех этих манипуляций, он любезно предоставит нам прокси интерфейс для доступа к **EntityManager**. Таким образом в системе появилось два реализованных бина:

@Autowired

**private** EntityManagerFactory entityManagerFactory;

@Autowired

**private** EntityManager entityManager

При этом *entityManagerFactory* будет указывать на **LocalContainerEntityManagerFactoryBean**, а второй, **entityManager** на Proxy EntityManager. При этом, **entityManager** будет реализовывать интефейс **SessionFactory**. Собственно на этом реализация закончена. В этой статье я упустил подробности реализации **SessionFactory**, так как это немного выходит за рамки этого обсуждения. Но надеюсь в следующей статье я разберусь и в этой теме.