**Создание сессии - варианты**

**Как делать правильнее всего**

Правильнее всего делать вот так

**return new Configuration().configure().buildSessionFactory();**

Основная причина, на мой взгляд, это то, что всенижеперечисленные подходы не будут вообще (!) правильно работать с Hibernate 5. Об этом [подробнее можно почитать здесь](https://stackoverflow.com/a/32711654/3405171).

**Хотя buildSessionFactory() объявлена как @Deprecated в Hibernate 4, в Hibernate 5 она запилена обратно**! Соответственно, вышеприведенный код будет правильно работать во всех версиях Hibernate.

**Или можно еще:**

**Первый подход**

private static SessionFactory configureSessionFactory() throws HibernateException {

Configuration configuration = new Configuration();

configuration.configure();

serviceRegistry = new ServiceRegistryBuilder()

.applySettings(configuration.getProperties())

.buildServiceRegistry();

sessionFactory = configuration.buildSessionFactory(serviceRegistry);

return sessionFactory;

}

Зачитывает конфигурацию из hibernate.cfg.xml и hibernate.properties. Должен работать во всех подверсиях Hibernate 4 правильно.

hibernate.cfg.xml зачитывается здесь configuration.configure()

Вот это applySettings(configuration.getProperties()) необходимо, чтобы не потерять properties из hibernate.cfg.xml (в этом файле можно указывать свойства так же, как в hibernate.properties). Например, если у Вас там свойства не указаны (а обычно они там не указываются), то достаточно вот этого

return new Configuration().configure().

buildSessionFactory(new ServiceRegistryBuilder().buildServiceRegistry());

Вообще, эта некрасивая возня с ServiceRegistryBuilder из-за того, что дополнительный тюнинг конфигурации вынесли в отдельный класс, аbuildSessionFactory() опрометчиво сделали @Deprecated.

В Hibernate 5 вообще хотели выбросить класс Configuration и перейти к конфигурированию с использованием StandardServiceRegistryBuilder и MetadataSources. Из javadocs [Configuration Hibernate 4.3](https://docs.jboss.org/hibernate/orm/4.3/javadocs/)

NOTE : This will be replaced by use of StandardServiceRegistryBuilder and MetadataSources instead after the 4.0 release at which point this class will become deprecated and scheduled for removal in 5.0. See HHH-6183, HHH-2578 and HHH-6586 for details

Далее, в Hibernate 5, судя по всему, от этой идеи отказались и данного замечания в документации к Configuraton нет.

hibernate.properties зачитываются извращенным образом в статическом инициализаторе класса [Environment](http://grepcode.com/file/repo1.maven.org/maven2/org.hibernate/hibernate-core/4.3.10.Final/org/hibernate/cfg/Environment.java?av=f" \l "206). То есть, при первом обращении к нему. Это, скорее всего, происходит здесь new Configuration().

**Второй подход**

Configuration configuration = new Configuration().addProperties(properties);

SessionFactory sf = configuration.buildSessionFactory(

new StandardServiceRegistryBuilder()

.applySettings(properties)

.build()

);

Будет работать только начиная с версии Hibernate 4.3, так как только начиная с этой версии присутствует класс StandardServiceRegistryBuilder. Вы можете считать, что StandardServiceRegistryBuilder — это тоже самое, что и ServiceRegistryBuilder. Начиная с версии 4.3 ServiceRegistryBuilder был помечен, как @Deprecated, а все его содержимое было скопировано в StandardServiceRegistryBuilder.

Так как здесь

Configuration configuration = new Configuration().addProperties(properties);

метод configure() не вызывается, то соответственно, информация о маппинге из файла hibernate.cfg.xml зачитываться не будет. Вообще, этот подход можно рассматривать как неполный вариант третьего подхода.

**Третий подход**

Configuration cfg = new Configuration().configure("hibernate.cfg.xml");

StandardServiceRegistryBuilder sb = new StandardServiceRegistryBuilder();

sb.applySettings(cfg.getProperties());

StandardServiceRegistry standardServiceRegistry = sb.build();

return new Configuration().configure().buildSessionFactory(standardServiceRegistry);

Этот подход будет работать правильно начиная с версии Hibernate 4.3 (из-за уже упомянутого StandardServiceRegistryBuilder). Но он содержит избыточность, о которой упомянул @Sergey. Если убрать избыточность, то этот вариант сведется к

Configuration cfg = new Configuration().configure();

StandardServiceRegistryBuilder sb = new StandardServiceRegistryBuilder();

sb.applySettings(cfg.getProperties());

StandardServiceRegistry standardServiceRegistry = sb.build();

return сfg.buildSessionFactory(standardServiceRegistry);

Что является тем же самым, что первый подход, так как StandardServiceRegistryBuilder — это тоже самое, что и ServiceRegistryBuilder.