<http://blog.csdn.net/chaoowang/article/details/8562262>

JPA是Java EE5规范之一，是一个orm规范，由厂商来实现该规范。目前有hibernate，OpenJPA，TopLink和EclipseJPA等实现

[**spring**](http://lib.csdn.net/base/javaee)提供三种方法集成JPA：

1、LocalEntityManagerFactoryBean：适用于那些仅使用JPA进行数据访问的项目。该FactoryBean根据 JPA PersistenceProvider自动检测配置文件进行工作，一般从“META-INF/persistence.xml”读取配置信息。这种方式最简单，但是不能设置Spring中定义的DataSource，且不支持Spring管理的全局事务。不建议使用此方式。这种方法实际上只适用于独立的应用程序和[**测试**](http://lib.csdn.net/base/softwaretest)环境（这正是JPA规范设计它的原因）。

    在Spring中的配置：

    <bean id=”entityManagerFactory” class=”org.springframework.orm.jpa.LocalEntityManagerFactoryBean”>

       <property name=”persistenceUnitName” value=”persistenceUnit”/>

    </bean>

2、从JNDI中获取：用于从**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)**EE服务器中获取指定的EntityManagerFactory，这种方式在Spring事务管理时一般要使用JTA事务管理。

    Spring中的配置：

<beans xm[ln](http://www.linuxso.com/command/ln.html)s="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:jee="http://www.springframework.org/schema/jee"

xsi:schemaLo[cat](http://www.linuxso.com/command/cat.html)ion="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/jee

http://www.springframework.org/schema/jee/spring-jee-3.0.xsd">

<jee:jndi-[look](http://www.linuxso.com/command/look.html)up id="entityManagerFactory" jndi-name="persistence/persistenceUnit"/>

</beans>

在标准的**[java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)**EE 5启动过程中，**[Java EE](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "_blank)**服务器自动检测持久化单元（例如应用程序文件包中的META-INF/persistence.xml） ，以及**[java ee](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "_blank)**部署描述符中定义给那些持久化单元命名上下文位置的环境的persistence-unit-ref项（例如web.xml）。

在这种情况下，整个持久化单元部署，包括持久化类的织入（字码码转换）都取决于**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "_blank)** EE服务器。 JDBC DataSource 通过在META-INF/persistence.xml 文件中的JNDI位置进行定义；EntityManager事务与服务器的JTA子系统整合。Spring仅仅用获得的 EntityManagerFactory, 通过依赖注入将它传递给应用程序对象，并为它管理事务（一般通过JtaTransactionManager）。

注意，如果在同一个应用程序中使用了多个持久化单元，JNDI获取的这种持久化单元的bean名称 应该与应用程序用来引用它们的持久化单元名称相符（例如@PersistenceUnit和 @PersistenceContext注解）。

在部署到Java EE 5服务器时使用该方法。关于如何将自定义JPA提供者部署到服务器，以及允许使用服务器提供的缺省提供者之外的JPA提供者，请查看服务器文档的相关说明。

3、LocalContainerEntityManagerFactoryBean：适用于所有环境的FactoryBean，能全面控制EntityManagerFactory配置，非常适合那种需要细粒度定制的环境。

该bean有以下属性：

persistenceUnitManager：用于获取JPA持久化单元，默认实现DefaultPersistenceUnitManager用于解决多配置文件情况。

dataSource：用于指定Spring定义的数据源。

persistenceXmlLocation:用于指定JPA配置文件，对于多JPA配置文件情况请选择设置persistenceUnitManager属性来解决。

persistenceUnitName：用于指定持久化单元名称。

persistenceProvider：用于指定持久化实现厂商类，如**[hibernate](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "_blank)**为：org.hibernate.ejb.HibernateProvider 类。

jpaVendorAdapter：用于设置JPA实现厂商的特定属性，如设置hibernate的是否自动生成DDL的属性generateDdl，这些属性是厂商特定的，因此最好在这里设置。目前spring提供HibernateJpaVendorAdapter，OpenJpaVendorAdapter，EclipseJpaVendorAdapter，TopLinkJpaVenderAdapter四个实现。其中最主要的属性是“database”，用来指定使用的**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "_blank)**类型。从而根据数据库类型决定如何将数据库特定异常转换为Spring一致性异常。目前支持以下数据库：DB2，DERBY,H2，HSQL,INFORMIX,[**MySQL**](http://lib.csdn.net/base/mysql),[**Oracle**](http://lib.csdn.net/base/oracle),POSTGRESQL,SQL\_SERVER,SYBASE

jpaDialect:用于指定一些高级特性，如事务管理等。目前Spring提供HibernateJpaDialect，OpenJpaDialect，EclipseJpaDialect，TopLinkJpaDialect和DefaultJpaDialect实现。注意DefaultJpaDialect不提供任何功能，因此在使用特定实现厂商的JPA实现时需要指定jpaDialect实现，如使用hibernate就使用HibernateJpaDialect。当指定**jpaVendorAdapter属性时可以不指定jpaDialect，会自动设置相应的JpaDialect实现；**

jpaProperties和jpaPropertyMap：指定JPA属性；如Hibernate中指定是否显示SQL的“hibernate.show\_sql”属性，对于jpaProperties设置的属性自动会放进jpaPropertyMap中；

loadTimeWeaver：用于指定LoadTimeWeaver实现，从而允许JPA 加载时修改相应的类文件。具体使用得参考相应的JPA规范实现厂商文档，如Hibernate就不需要指定loadTimeWeaver。

JPA配置实例：

  persistence.xml：

<?xml version="1.0" encoding="UTF-8"?>

<persistence version="1.0"

xmlns="http://java.[su](http://www.linuxso.com/command/su.html)n.com/xml/ns/persistence"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/persistence

http://java.sun.com/xml/ns/persistence/persistence\_1\_0.xsd">

<persistence-unit name="persistenceUnit" transaction-type="RESOURCE\_LOCAL"/>

</persistence>

persistence.xml中，指定持久化单元名称和事务类型，其他在Spring中配置。

applicationContext.xml

<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">

<property name="dataSource" ref="dataSource"/>

<property name="persistenceXmlLocation" value="chapter8/persistence.xml"/>

<property name="persistenceUnitName" value="persistenceUnit"/>

<property name="persistenceProvider" ref="persistenceProvider"/>

<property name="jpaVendorAdapter" ref="jpaVendorAdapter"/>

<property name="jpaDialect" ref="jpaDialect"/>

<property name="jpaProperties">

<props>

<prop key="hibernate.show\_sql">true</prop>

</props>

</property>

</bean>

<bean id="persistenceProvider" class="org.hibernate.ejb.HibernatePersistence"/>

<bean id="jpaVendorAdapter" class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">

<property name="generateDdl" value="fa[ls](http://www.linuxso.com/command/ls.html)e" />

<property name="database" value="HSQL"/>

</bean>

<bean id="jpaDialect" class="org.springframework.orm.jpa.vendor.HibernateJpaDialect"/>

* **LocalContainerEntityManagerFactoryBean:**指定使用本地容器管理EntityManagerFactory，从而进行细粒度控制；
* **dataSource**属性指定使用Spring定义的数据源；
* **persistenceXmlLocation**指定JPA配置文件为chapter8/persistence.xml，且该配置文件非常简单，具体配置完全在Spring中进行；
* **persistenceUnitName**指定持久化单元名字，即JPA配置文件中指定的;
* **persistenceProvider**:指定JPA持久化提供商，此处使用Hibernate实现HibernatePersistence类；
* **jpaVendorAdapter**：指定实现厂商专用特性，即generateDdl= false表示不自动生成DDL，database= HSQL表示使用hsqldb数据库；
* **jpaDialect**：如果指定jpaVendorAdapter此属性可选，此处为HibernateJpaDialect；
* **jpaProperties**：此处指定“hibernate.show\_sql =true”表示在日志系统debug级别下将打印所有生成的SQL。

JpaTemplate类：

Spring提供JpaTemplate模板类用于简化事务管理及常见操作，类似于JdbcTemplate模板类。