JPA ile Veri Erişimi

harezmi bilişim çözümleri

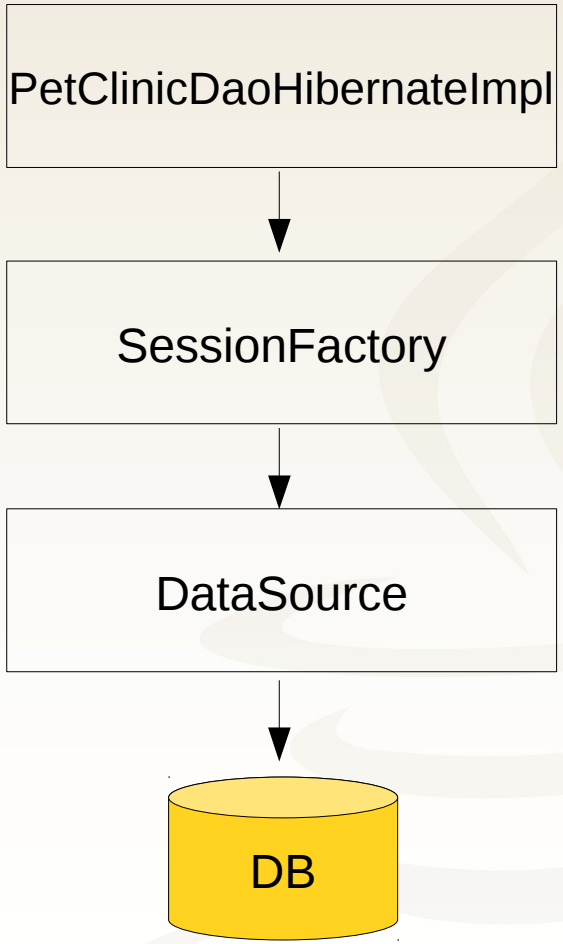
ORM ile Veri Erişimi

- Spring veri erişimi, Hibernate **Session** ve JPA **EntityManager** nesnelerine erişimi kolaylaştırır
- JDBC işlemleri ile ORM işlemlerini **aynı TX** içerisinde gerçekleştirebilirsiniz
- **Ortak** bir veri erişim **exception hiyerarşisi** sunar
- **Entegrasyon** birim testlerinin yazılmasını ve çalıştırılmasını kolaylaştırır

Hibernate ile Veri Erişimi ve SessionFactory Konfigürasyonu

DataSource ve SessionFactory **Spring managed bean**'lar olarak tanımlanır

SessionFactory nesnesini ApplicationContext içerisinde yönetmek için **LocalSessionFactoryBean** ile bean tanımı yapılır



Hibernate ile Veri Erişimi ve SessionFactory Konfigürasyonu

```
<bean id="sessionFactory"
class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">

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

   <property name="packagesToScan"
              value="com.javaegitimleri.petclinic.model"/>

    <property name="hibernateProperties">
      <value>
          hibernate.dialect=org.hibernate.dialect.H2Dialect
      </value>
     </property>

</bean>
```

Hibernate API ile DAO Geliştirilmesi

- Doğrudan **Hibernate API**'si ile çalışılabilir
- **SessionFactory** DAO nesnesine **enjekte** edilir
- Spring 3 öncesi dönemin **HibernateTemplate** ve **HibernateDaoSupport** sınıflarına gerek kalmamıştır
- TX yönetimi devrede ise Hibernate'in **Current Session Context** özelliği ile current Session'a kolayca erişilebilir

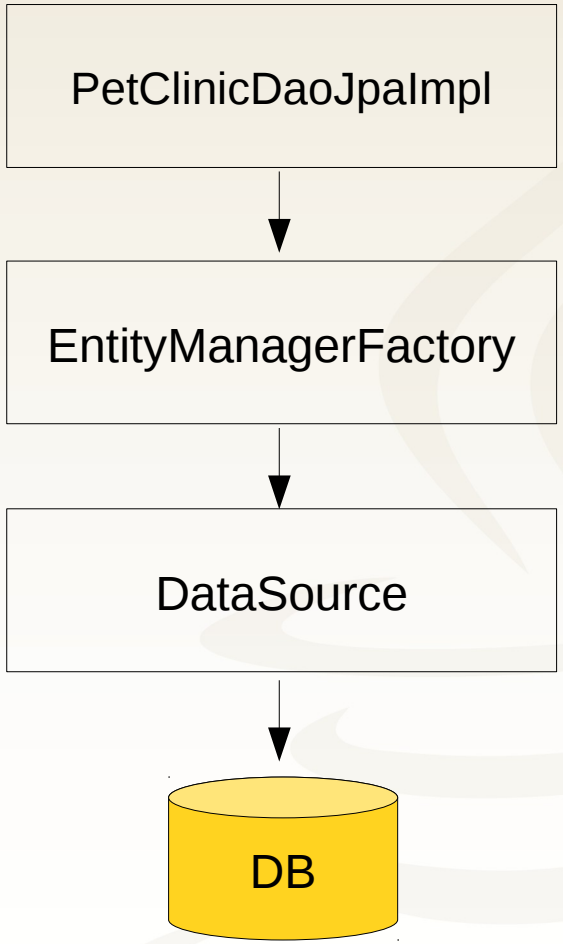
Hibernate API ile DAO Geliştirilmesi

```
@Repository
public class ProductDaoImpl implements ProductDao {

    private SessionFactory sessionFactory;

    @Autowired
    public void setSessionFactory(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    public Collection loadProductsByCategory(String category) {
        return sessionFactory.getCurrentSession()
                .createQuery("from test.Product product where
product.category=?")
                .setParameter(0, category)
                .list();
    }
}
```

JPA ile Veri Erişimi ve EMF Konfigürasyonu

DataSource ve EntityManagerFactory **Spring managed bean**'lar olarak tanımlanır

EntityManagerFactory nesnesini ApplicationContext içerisinde yönetmek için **LocalContainerEntityManagerFactoryBean** ile bean tanımı yapılır



JPA ile Veri Erişimi ve EMF Konfigürasyonu

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

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

    <property name="packagesToScan"
                        value="com.javaegitimleri.petclinic.model" />
    <property name="jpaProperties">
        <value>
            hibernate.dialect=org.hibernate.dialect.H2Dialect
        </value>
    </property>

    <property name="jpaVendorAdapter">
        <bean
        class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"/>
    </property>
</bean>
```

JPA ile Veri Erişimi ve EMF Konfigürasyonu

- **JpaVendorAdapter**, **vendor spesifik özellikler**in Spring'in JPA EntityManagerFactory konfigürasyonunda **tek nokta**dan yapılmasını sağlar
- **JpaDialect** tanımı aktif **JDBC Connection nesnesine erişim** gibi JPA'nın imkan vermediği özellikleri devreye alır
- Tanımı **HibernateJpaVendorAdapter** tarafından da yapıldığından ayrıca bean konfigürasyonuna eklemeye gerek yoktur

JPA ile Veri Erişimi ve EMF Konfigürasyonu

- javax.persistence.spi.**PersistenceProvider** arayüzünden bir sınıf ise **EntityManagerFactory nesnesini yaratmak** için kullanılır
- Bean konfigürasyonuna **persistenceProviderClass** olarak verilir
- Tanımı **HibernateJpaVendorAdapter** tarafından da yapıldığından ayrıca bean konfigürasyonuna eklemeye gerek yoktur

JPA API ile DAO Geliştirilmesi

- Doğrudan **JPA API**'si ile çalışılabilir
- **@PersistenceContext** annotasyonu sayesinde transactional **EntityManager** nesnesi DAO nesnesine **enjekte** edilebilir
- @PersistenceContext'in kullanılabilmesi için Spring'in **TX yönetiminin aktif olması** gerekir
- İstenirse **@PersistenceUnit** ile EntityManagerFactory'de DAO'ya enjekte edilebilir

JPA API ile DAO Geliştirilmesi

```
@Repository
public class ProductDaoImpl implements ProductDao {

    private EntityManager entityManager;

    @PersistenceContext
public void setEntityManager(EntityManager entityManager) {
        this.entityManager = entityManager;
    }

    public Collection loadProductsByCategory(String category) {
        return entityManager.createQuery(
                "select p from Product p where p.category=?1")
                .setParameter(1, category)
                .getResultList();
    }
}
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com