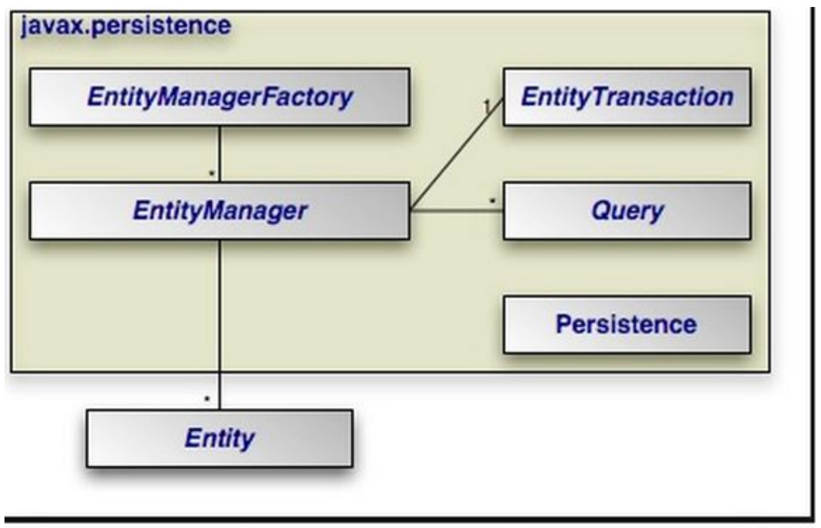
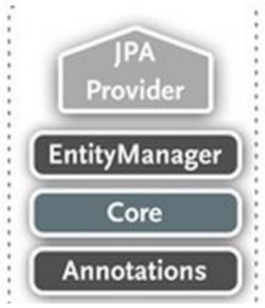
8 JPA interview questions and answers

Posted on August 27, 2014 by Arulkumaran Kumaraswamipillai

Q1. What is a JPA? What are its key components?

A1. The process of mapping Java objects to database tables and vice versa is called "Object-relational mapping" (ORM). The Java Persistence API provides Java developers with an object/relational mapping (ORM) facility for managing relational data in Java applications. JPA is a specification and several implementations are available like EJB, JDO, Hibernate, and Toplink. Using JPA and relevant implementation like Hibernate, developers can map, store, update and retrieve data from relational databases to Java objects and vice versa.





JPA

Q2. What is the difference between hibernate.cfg.xml and persistence.xml?

A2. If you are using Hibernate's proprietary API, you'll need the hibernate.cfg.xml. If you are using JPA i.e. Hibernate **EntityManager**, you'll need the **persistence.xml**. You will not need both as you will be using either Hibernate proprietary API or JPA. However, if you had used Hibernate Proprietary API using hibernate.cfg.xml with hbm.xml mapping files, and now wanted to start using JPA, you can reuse the existing configuration files by referencing the hibernate.cfg.xml in the persistence.xml in the hibernate.ejb.cfgfile property and reuse the existing hbm.xml files. In a long run, migrate hbm.xml files to JPA annotations.

Q3. What are the 3 artifacts required to implement a JPA compliant project?

A3.

1. An entity class
2. A persistence.xml file

16 Technical Key Areas

80+ step by step Java Tutorials

3. An interface which you will use to perform CRUD operations like insert, update, or find an entity

Q4. What is an EntityManagerFactory and a Persistence unit?
A4. The **EntityManager** is created by the **EntitiyManagerFactory** which is configured by the persistence unit. The persistence unit is described via the file "**persistence.xml**" in the directory META-INF in the source folder. It defines a set of entities which are logically connected and the connection properties as shown below.

persistence.xml

```
<persistence version="1.0"
 xmlns="http://java.sun.com/xml/ns/persistence"
 xsi:schemaLocation="http://java.sun.com/xml/ns/

  <persistence-unit name="myapp-server" transact

    <provider>org.hibernate.ejb.HibernatePersist

    <!-- import other mapping files if any -->
    <mapping-file>META-INF/myApp2.xml</mapping-f

    <class>com.mycompany.myapp.Person</class>

    <exclude-unlisted-classes>true</exclude-unli

    <properties>
        <property name="javax.persistence.jdbc.d
        <property name="javax.persistence.jdbc.u
        <property name="javax.persistence.jdbc.u
        <property name="javax.persistence.jdbc.p
    </properties>

  </persistence-unit>
</persistence>

```

Usually, JPA defines a persistence unit through the META-INF/persistence.xml file. Starting with Spring 3.1, this XML file is no longer necessary – the **LocalContainerEntityManagerFactoryBean** now supports a 'packagesToScan' property where the packages to scan for @Entity classes can be specified. The snippet below shows how you can bootstrap with or without persistence.xml.

```
//...
```

100+ Java pre-interview coding tests

open all | close all
- Can you write code? (
- ♦ Complete the given
- Converting from A to I
- Designing your classe
- Java Data Structures
- Passing the unit tests
- What is wrong with th
- Writing Code Home A
- Written Test Core Jav
- Written Test JEE (1)

How good are your .....?

open all | close all
- Career Making Know-
- Job Hunting & Resum

```
@Configuration
@PropertySource(value =
{"classpath:/common/jpa.properties"
})
@EnableTransactionManagement
public class JpaConfig
{
    @Value("${my_app_common_jpa_showSql:false}")
    private Boolean showSql;

    @Value("${my_app_common_jpa_hibernateDialect
    private String hibernateDialect;

    @Value("${my_app_common_jpa_generateStatisti
    private Boolean generateStatistics;

    @Value("${my_app_common_jpa_generateDdl:fals
    private Boolean generateDdl;

    @Value("${my_app_common_jpa_databasePlatform
    private String databasePlatform;

    @Resource(name = "myAppDataSource")
    private DataSource dataSource;

    @Bean
    public Map<String, Object> jpaProperties()
    {
        Map<String, Object> props = new HashMap<
        props.put("hibernate.dialect", hibernate
        props.put("hibernate.generate_statistics

        return props;
    }

    @Bean
    public JpaVendorAdapter jpaVendorAdapter()
    {
        HibernateJpaVendorAdapter hibernateJpaVe
        hibernateJpaVendorAdapter.setShowSql(sho
        hibernateJpaVendorAdapter.setGenerateDdl
        hibernateJpaVendorAdapter.setDatabasePla

        return hibernateJpaVendorAdapter;
    }

    @Bean
    public PlatformTransactionManager transactio
    {
        return new JpaTransactionManager(entityM
    }

    @Bean
    public EntityManagerFactory entityManagerFac
    {
        LocalContainerEntityManagerFactoryBean l
        lef.setDataSource(dataSource);
        lef.setJpaPropertyMap(this.jpaProperties
        lef.setJpaVendorAdapter(this.jpaVendorAd
        lef.setPersistenceXmlLocation("META-INF/
        lef.afterPropertiesSet();
        return lef.getObject();
    }

    @Bean
    public PersistenceExceptionTranslationPostPr
    {
        return new PersistenceExceptionTranslati
    }

    @Bean
    public HibernateExceptionTranslator hibernat
    {
        return new HibernateExceptionTranslator(
    }
}

```

The **jpa.properties** can be defined as shown below.

```
# properties for JPA
my_app_common_jpa_showSql=false
my_app_common_jpa_generateDdl=false

my_app_common_jpa_databasePlatform=SYBASE
my_app_common_jpa_hibernateDialect=org.hibernate.
my_app_common_jpa_generateStatistics=true
my_app_common_aesJndiName=java:comp/env/jdbc/my_d

```

Q5. What is an EntityManager?
A5. The entity manager javax.persistence.**EntityManager** provides the operations from and to the database, e.g. find objects, persists them, remove objects from the database, etc. Entities which are managed by an EntityManager will automatically propagate these changes to the database (if this happens within a commit statement). These objects are known as persistent object. If the Entity Manager is closed (via close()) then the managed entities are in a detached state. These are known as the detached objects. If you want synchronize them again with the database, the a Entity Manager provides the merge() method. Once merged, the object(s) becomes perstent objects again.

The **EntityManager** is the API of the persistence context, and an EntityManager can be injected directly in to a DAO without requiring a JPA Template. The Spring Container is capable of acting as a JPA container and of injecting the EntityManager by honoring the @PersistenceContext (both as field-level and a method-level annotation).

```
//...
import com.mycompany.myapp.model.Person
import java.util.ArrayList;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

import org.springframework.stereotype.Repository

@Repository("personDao")
public class PersonDaoImpl implements PersonDao
{
    @PersistenceContext
    private EntityManager em;

    @Override
    public List<Person> fetchPersonByFirstname(S
    {
        Query query = em.createQuery( "from Pers
        query.setParameter("firstName", fName);

        List<Person> persons = new ArrayList<Per
        List<Person> results = query.getResultLi

        return persons;
    }

    public Person find(Integer id)
    {
        return em.find(Person.class, id);
    }
}

```

Q6. What is an Entity?

A6. A class which should be persisted in a database it must be annotated with javax.persistence.**Entity**. Such a class is called Entity. An instances of the class will be a row in the person table. So, the columns in the person table will be mapped to the Person java object annotated as @Entity. Here is the sample Person class.

```
@Entity
@Table(name = "person", schema = "dbo", catalog
@Where(clause = "inactiveFlag = 'N'")
@TypeDefs(
{@TypeDef(name = "IdColumn", typeClass = IdColum
public class UnitTrustPrice implements java.io.S
{
   private int id;
   private String firstName;
   private byte[] timestamp;

   public Person(){}

   @Id
   @GeneratedValue(strategy = GenerationType.IDE
   @Column(name = "person_id", nullable = false,
   @Type(type = "int")
   public int getId()
```

```
    {
        return this.id;
    }

    public void setId(int id)
    {
        this.id = id;
    }

    @Version
    @Column(name = "timestamp", insertable = fals
    @Generated(GenerationTime.ALWAYS)
    public byte[] getTimestamp()
    {
        return this.timestamp;
    }

    public void setTimestamp(byte[] timestamp)
    {
       this.timestamp = timestamp;
    }

    @Column(name = "first_name", nullable = false
    public String getFirstName()
    {
        return this.firstName;
    }

    public void setFirstName(String fName)
    {
        this.firstName = fName;
    }
}

```

Q7. What are the dependency jars required for a JPA application?

A7.Add relevant jar files via your Maven pom.xml file.

```
<!-- JPA and hibernate -->
<dependency>
 <groupId>org.hibernate</groupId>
 <artifactId>hibernate-core</artifactId>
</dependency>
<dependency>
 <groupId>org.hibernate</groupId>
 <artifactId>hibernate-entitymanager</artifactId
</dependency>
<dependency>
 <groupId>org.hibernate.javax.persistence</group
 <artifactId>hibernate-jpa-2.0-api</artifactId>
</dependency>

<!-- validator -->
<dependency>
 <groupId>javax.validation</groupId>
 <artifactId>validation-api</artifactId>
</dependency>

<dependency>
 <groupId>org.hibernate</groupId>
 <artifactId>hibernate-validator</artifactId>
</dependency>

```

Q8 What is difference between CrudRepository and JpaRepository interfaces in Spring Data?

A8.**JpaRepository** extends **PagingAndSortingRepository**, which in turn extends **CrudRepository**. Their main functions are:

— CrudRepository mainly provides CRUD (Create Read Update and Delete) functions.
— PagingAndSortingRepository provide methods to do pagination and sorting records.
— JpaRepository provides some JPA related method such as flushing the persistence context and delete records in a batch.

Q9. What steps are required for Spring CrudRepository to work with JPA?

A9.

Step 1: Add the jar dependency to your maven pom.xml file.

```
<dependency>
 <groupId>org.springframework.data</groupId>
 <artifactId>spring-data-jpa</artifactId>
 <version>1.2.0-RELEASE</version>
<dependency>

```

Step 2: Define the JPA entity — that is your model class that maps to the table in the database.

```
//...
@Entity
@Table(name = "ReportStructure")
public class Node extends AbstractPersistable<Lo
{
    private static final long serialVersionUID =

    @ManyToOne
    @JoinColumn(name = "ParentId", insertable =
    private Node parent;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "ParentId", nullable = fa
    private List<Node> children = new ArrayList<

    @OneToOne(cascade = CascadeType.ALL)
    @PrimaryKeyJoinColumn
    private NodeAttributes attributes;

    //....
}

```

Step 3: Define the CRUD repository by extending Spring's CrudRepository class. The CrudRepository gives you out of the box access to the following standard methods

— **delete(T entity)**, which deletes the entity given as a parameter.
— **findAll()** , which returns a list of entities.
— **findOne(ID id)**, which returns the entity using the id given a parameter as a search criteria.
— **save(T entity)** which saves the entity given as a parameter.

You can provide additional custom methods as shown below,

```
import org.springframework.data.jpa.repository.J
import org.springframework.data.jpa.repository.Q
import org.springframework.data.repository.CrudR

public interface NodeRepository extends CrudRepo
{
    @Query("SELECT n from Node n JOIN n.key k WI
            + "WHERE n.parent is null and n.isSo
    List<Node> find(String clientId, Date evalDa

    @Query("SELECT key from NodeKey key WHERE ke
    List<NodeKey> fetch(String clientId);
}

```

Step 4:: The Spring config file to wire up JPA. This example uses HSQL.

```
<!-- Directory to scan for repository classes --
<jpa:repositories
   base-package="com.mydomain.model" />

<bean class="org.springframework.orm.jpa.JpaTran
  id="transactionManager">
  <property name="entityManagerFactory"
      ref="entityManagerFactory" />
  <property name="jpaDialect">
```

```
    <bean class="org.springframework.orm.jpa.ven
  </property>
</bean>

<bean id="entityManagerFactory"
  class="org.springframework.orm.jpa.LocalContai
  <property name="dataSource" ref="dataSource" /
  <property name="jpaVendorAdapter">
    <bean class="org.springframework.orm.jpa.ven
      <property name="generateDdl" value="true"
      <property name="database" value="HSQL" />
    </bean>
  </property>
</bean>

```

Step 5:: Use the NodeRepository for CRUD operations in the service layer.

```
public class ReportServiceImpl extends ReportServ

   @Autowired
   NodeRepository nodeRepository;

   //...
}
```

Popular Posts

♦ 11 Spring boot interview questions & answers

825 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

766 views

18 Java scenarios based interview Questions and Answers

400 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | **Latest Posts**

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ JTA interview Q&A

001A: ♦ 7+ Java integration styles & patterns interview questions & answers ›

Posted in Hibernate Job Interview Essentials, JPA, member-paid

Tags: Java/JEE FAQs, JEE FAQs

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from "Blah blah" to "Wow"? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer