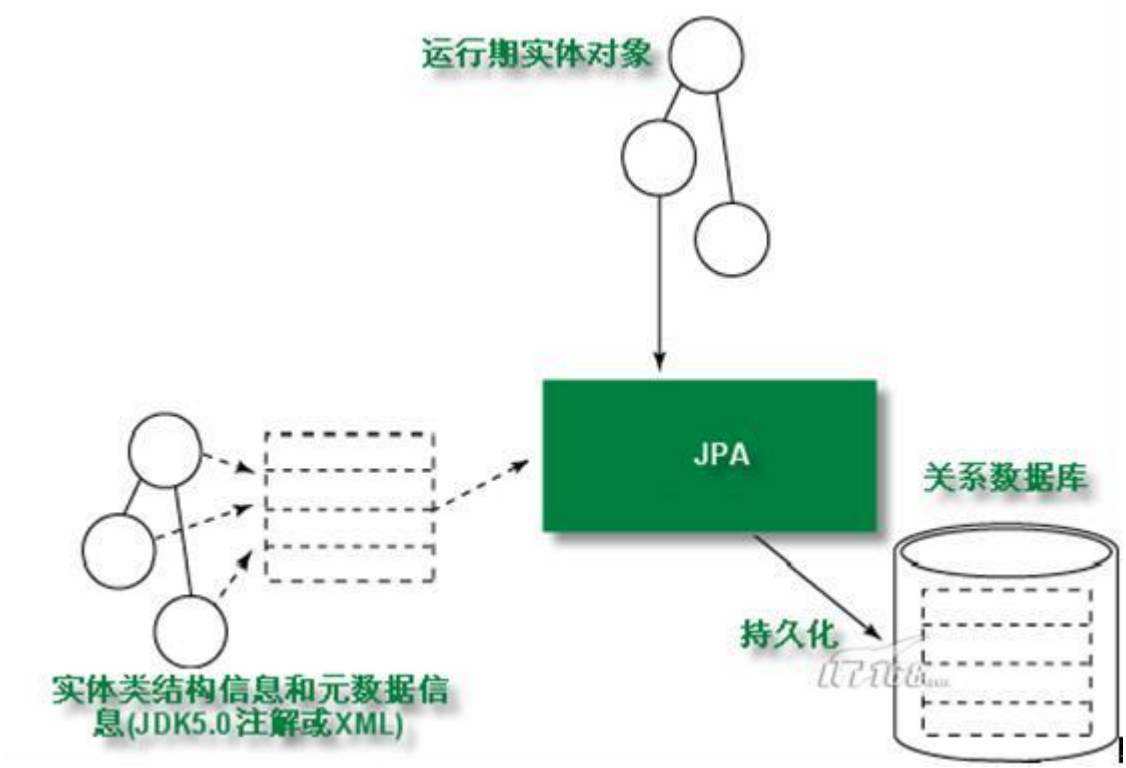
JPA 教程

1.JPA概述

JPA(Java Persistence API)作为 Java EE 5.0 平台标准的 ORM 规范，将得到所有 Java EE 服务器的支持。Sun 这次吸取了之前 EJB 规范惨痛失败的经历，在充分吸收现有 ORM 框架的基础上，得到了一个易于使用、伸缩性强的 ORM 规范。从目 前的开发社区的反应上看，JPA 受到了极大的支持和赞扬，JPA 作为 ORM 领域标准化整合者的目标应该不难实现。

JPA 通过 JDK 5.0 注解或 XML 描述对象－关系表的映射关系，并将运行期的实体对象持久化到数据库中，图 1 很好地描述了 JPA 的结构：



Sun 引入新的 JPA ORM 规范出于两个原因：其一，简化现有 Java EE 和 Java SE 应用的对象持久化的开发工作；其二，Sun 希望整合对 ORM 技术，实现天下归一。

JPA 由 EJB 3.0 软件专家组开发，作为 JSR-220 实现的一部分。但它不囿于 EJB 3.0，你可以在 Web 应用、甚至桌面应用中使用。JPA 的宗旨是为 POJO 提供持久化标准规范，由此可见，经过这几年的实践探索，能够脱离容器独立运 行，方便开发和测试的理念已经深入人心了。目前 Hibernate 3.2、TopLink 10.1.3 以及 OpenJpa 都提供了 JPA 的实现。

JPA 的总体思想和现有 Hibernate、TopLink，JDO 等 ORM 框架大体一致。总的来说，JPA 包括以下 3 方面的技术：

ORM 映射元数据，JPA 支持 XML 和 JDK 5.0 注解两种元数据的形式，元数据描述对象和表之间的映射关系，框架据此将实体对象持久化到数据库表中；

JPA 的 API，用来操作实体对象，执行 CRUD 操作，框架在后台替我们完成所有的事情，开发者从繁琐的 JDBC 和 SQL 代码中解脱出来。

查询语言，这是持久化操作中很重要的一个方面，通过面向对象而非面向数据库的查询语言查询数据，避免程序的 SQL 语句紧密耦合。

2.实体对象

访问数据库前，我们总是要设计在应用层承载数据的领域对象（Domain Object），ORM 框架将它们持久化到数据库表中。为了方便后面的讲解，我们用论坛应用为例，建立以下的领域对象：

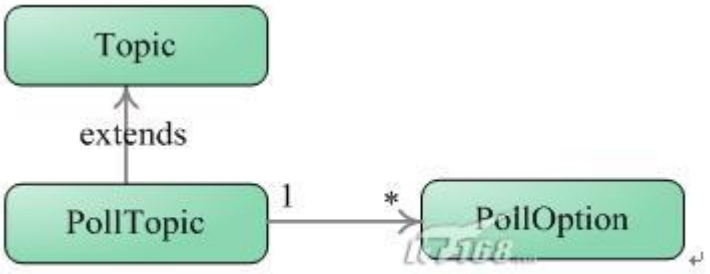


图 2 领域对象

Topic 是论坛的主题，而 PollTopic 是调查性质的论坛主题，它扩展于 Topic，一个调查主题拥有多个选项 PollOption。这三个领域对象很好地展现了领域对象之间继承和关联这两大核心的关系。这 3 个领域对象将被映射到数据库的两张表中：

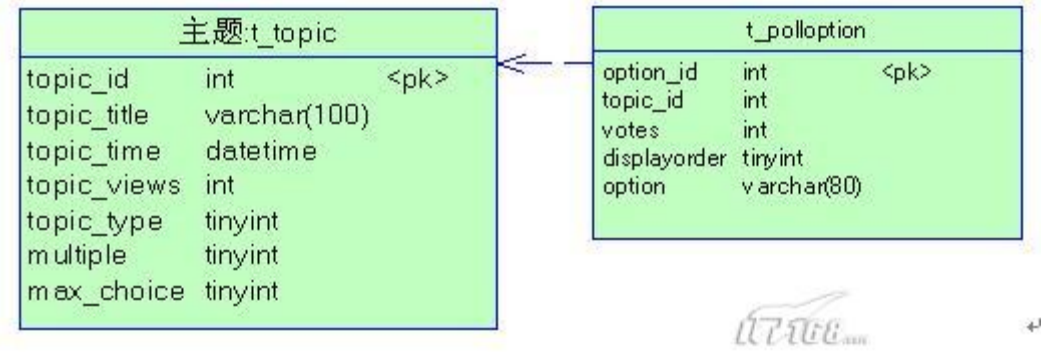


图 3 领域对象对应的数据表

其中，Topic 及其子类 PollTopic 将映射到同一张 t_topic 表中，并用 topic_type 字段区分两者。而 PollOption 映射到 t_polloption 中。

具有 ORM 元数据的领域对象称为实体（Entity），按 JPA 的规范，实体具备以下的条件：

必须使用 javax.persistence.Entity 注解或者在 XML 映射文件中有对应的元素；

必须具有一个不带参的构造函数，类不能声明为 final，方法和需要持久化的属性也不能声明为 final；

如果游离状的实体对象需要以值的方式进行传递，如通 Session bean 的远程业务接口传递，则必须实现 Serializable 接口；

需要持久化的属性，其访问修饰符不能是 public，它们必须通过实体类方法进行访问。

3.使用注解元数据

基本注解

首先，我们对 Topic 领域对象进行注解，使其成为一个合格的实体类：

代码清单 1：Topic 实体类的注解

```
package com.baobaotao.domain;

…

import javax.persistence.Column;

import javax.persistence.DiscriminatorColumn;

import javax.persistence.DiscriminatorType;

import javax.persistence.DiscriminatorValue;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Inheritance;

import javax.persistence.InheritanceType;
```

```
import javax.persistence.Temporal;

import javax.persistence.TemporalType;

@Entity(name = "T_TOPIC") ①

public class Topic implements Serializable ...{

@Id   ②-1

@GeneratedValue(strategy = GenerationType.TABLE)
②-2

@Column(name = "TOPIC_ID") ②-3

private int topicId;

@Column(name = "TOPIC_TITLE", length = 100) ③

private String topicTitle;

@Column(name = "TOPIC_TIME")

@Temporal(TemporalType.DATE) ④

private Date topicTime;

@Column(name = "TOPIC_VIEWS")

private int topicViews;

//省略 get/setter 方法

}
```

@Entity：将领域对象标注为一个实体，表示需要保存到数据库中，默认情况下类名即为表名，通过 name 属性显式指定表名，如①处的 name = "T_TOPIC"，表示 Topic 保存到 T_TOPIC 表中；

@Id ：对应的属性是表的主键，如②-1 所示；

@GeneratedValue：主键的产生策略，通过 strategy 属性指定。默认情况下，JPA 自动选择一个最适合底层数据库的主键生成策略，如 SqlServer 对应 identity，MySql 对应 auto increment。在 javax.persistence.GenerationType 中定义了以下几种可供选择的策略：

1) IDENTITY：表自增键字段，Oracle 不支持这种方式；

2) AUTO： JPA 自动选择合适的策略，是默认选项；

3) SEQUENCE：通过序列产生主键，通过@SequenceGenerator 注解指定序列名，MySql 不支持这种方式；

4) TABLE：通过表产生主键，框架借由表模拟序列产生主键，使用该策略可以使应用更易于数据库移植。不同的 JPA 实现商生成的表名是不同的，如 OpenJPA 生成 openjpa_sequence_table 表 Hibernate 生成一个 hibernate_sequences 表，而 TopLink 则生成 sequence 表。这些表都具有一个序列名和对应值两个字段，如 SEQ_NAME 和 SEQ_COUNT。

@Column(name = "TOPIC_ID")：属性对应的表字段。我们并不需要指定表字段的类型，因为 JPA 会根据反射从实体属性中获取类型；如果是字符串类型，我们可以指定字段长度，以便可以自动生成 DDL 语句，如③处所示；

@Temporal(TemporalType.DATE)：如果属性是时间类型，因为数据表对时间类型有更严格的划分，所以必须指定具体时间类型，如④所示。在 javax.persistence.TemporalType 枚举中定义了 3 种时间类型：

1) DATE ： 等于 java.sql.Date

2) TIME ： 等于 java.sql.Time

3) TIMESTAMP ： 等于 java.sql.Timestamp

继承关系

Topic 和 PollTopic 是父子类，JPA 采用多种方法来支持实体继承。在父类中必须声明继承实体的映射策略，如代码清单 2 所示：

代码清单 2：继承实体的映射策略

```
…

@Entity(name = "T_TOPIC")

@Inheritance(strategy =
InheritanceType.SINGLE_TABLE) ①

@DiscriminatorColumn(name = "TOPIC_TYPE",
discriminatorType =

DiscriminatorType.INTEGER, length = 1) ②

@DiscriminatorValue(value="1")③
```

```
public class Topic implements Serializable ...{

…

}
```

对于继承的实体，在 javax.persistence.InheritanceType 定义了 3 种映射策略：

SINGLE_TABLE：父子类都保存到同一个表中，通过字段值进行区分。这是我们 Topic 实体所采用的策略， Topic 和 PollTopic 都保存到同一张表中，通过 TOPIC_TYPE 字段进行区分，Topic 在 T_TOPIC 表中对应 TOPIC_TYPE= 1 的记录，而 PollTopic 对应 TOPIC_TYPE=2 的记录（稍后在 PollTopic 实体中指定）；区别的字段通过 @DiscriminatorColumn 说明，如②所示，区分字段对应该实体的值通过@DiscriminatorValue 指定，如③所示；

JOINED：父子类相同的部分保存在同一个表中，不同的部分分开存放，通过表连接获取完整数据；

TABLE_PER_CLASS：每一个类对应自己的表，一般不推荐采用这种方式。

关联关系

我们再继续对 PollTopic 进行注解，进一步了解实体继承的 JPA 映射定义：

代码清单 3：PollTopic 映射描述

```
package com.baobaotao.domain;

…

@Entity

@DiscriminatorValue(value="2") ①

public class PollTopic extends Topic ...{②继承于 Topic 实体

private boolean multiple; ③

@Column(name = "MAX_CHOICES")

private int maxChoices;

@OneToMany(mappedBy="pollTopic",cascade=CascadeType.ALL) ④
```

```
private Set options = new HashSet();

//省略 get/setter 方法

}
```

在①处，通过@DiscriminatorValue 将区分字段 TOPIC_TYPE 的值为 2。由于 PollTopic 实体继承于 Topic 实体，其它的元数据信息直接从 Topic 获得。

JPA 规范规定任何属性都默认映射到表中，所以虽然我们没有给③处的 multiple 属性提供注解信息，但 JPA 将按照 默认的规则对该字段进行映射：字段名和属性名相同，类型相同。如果我们不希望将某个属性持久化到数据表中，则可以通过 @Transient 注解显式指定：

@Transient

private boolean tempProp1;

在④处，我们通过@OneToMany 指定了一个一对多的关联关系，一个 PollTopic 包括多个 PollOption 对象（我们将在稍后的 PollOption 中通过 ManyToOne 描述 PollOption 和 PollTopic 的关系，以建立 PollTopic 和 PollOption 的双向关联关系）。

@OneToMany 中通过 mappedBy 属性指定“Many”方类引用“One”方类的属性名，这里 mappedBy="pollTopic"表示 PollOption 实体拥有一个指定 PollTopic 的 pollTopic 属性。

下面，我们来看一下 Many 方 PollOption 实体类的映射描述：

代码清单 4：PollOption 映射描述

```
package com.baobaotao.domain;

…

@Entity(name="T_POLL_OPTION")

public class PollOption implements
Serializable ...{

@Id

@GeneratedValue(strategy = GenerationType.TABLE)

@Column(name = "OPTION_ID")
```

```
private int optionId;

@Column(name = "OPTION_ITEM")

private String optionItem;

@ManyToOne  ①

@JoinColumn(name="TOPIC_ID", nullable=false) ②

private PollTopic pollTopic;

}
```

在①处通过@ManyToOne 描述了 PollOption 和 PollTopic 的多对一关联关系，并通过@JoinColumn 指定关联 PollTopic 实体所对应表的“外键”，如②所示。

当然也可以通过@OneToOne 和@ManyToMany 指定一对一和多以多的关系，方法差不多，不再赘述。

Lob 字段

在 JPA 中 Lob 类型类型的持久化很简单，仅需要通过特殊的 Lob 注解就可以达到目的。下面，我们对 Post 中的 Lob 属性类型进行标注：

代码清单 5 Post：标注 Lob 类型属性

```
package com.baobaotao.domain;

…

import javax.persistence.Basic;

import javax.persistence.Lob;

@Entity(name = "T_POST")

public class Post implements Serializable ...{

…

@Lob ①-1

@Basic(fetch = FetchType.EAGER) ①-2

@Column(name = "POST_TEXT", columnDefinition =
"LONGTEXT NOT NULL") ①-3

private String postText;
```

```
@Lob ②-1

@Basic(fetch = FetchType. LAZY) ②-2

@Column(name = "POST_ATTACH", columnDefinition =
"BLOB") ②-3

private byte[] postAttach;

…

}
```

postText 属性对应 T_POST 表的 POST_TEXT 字段，该字段的类型是 LONTTEXT，并且非空。JPA 通过@Lob 将属性标注为 Lob 类型，如①-1 和②-1 所示。通过@Basic 指定 Lob 类型数据的获取策略，FetchType.EAGER 表示非延迟 加载，而 FetchType. LAZY 表示延迟加载，如①-2 和②-2 所示。通过@Column 的 columnDefinition 属性指定数据表对应的 Lob 字段类型，如①-3 和② -3 所示。

关于 JPA 注解的更多信息，你可以通过这篇文章进行更加深入的学习：http: //www.oracle.com/technology/products/ias/toplink/jpa/resources/toplink-jpa-annotations.html

4.使用 XML 元数据

除了使用注解提供元数据信息外，JPA 也允许我们通过 XML 提供元数据信息。条条道路通罗马，路路都是安康道，开发者安 全可以根据自己的习惯喜好择一而从。按照 JPA 的规范，如果你提供了 XML 元数据描述信息，它将覆盖实体类中的注解元数据信息。XML 元数据信息以 orm.xml 命名，放置在类路径的 META-INF 目录下。

JPA 尽量让 XML 和注解的元数据在描述的结构上相近，降低学习曲线和转换难度，所以我们在学习注解元数据后，学习 XML 元数据变得非常简单。下面，我们给出以上实体的 XML 描述版本，你可以对照注解的描述进行比较学习：

代码清单 6 XML 元数据配置：orm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings xmlns="http://java.sun.com/xml/ns/persistence/orm"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence/orm
   http://java.sun.com/xml/ns/persistence/orm_1_0.xsd"
    version="1.0">
①实体对象所在的包
    <package>com.baobaotao.domain</package>
```

```
     <entity class="Topic">
②Topic 实体配置
         <table name="T_TOPIC" />
         <attributes>
             <id name="topicId">
                 <column name="TOPIC_ID"/>
                 <generated-value strategy="TABLE" />
             </id>
             <basic name="topicTitle">
                 <column name="TOPIC_TITLE" length="30" />
             </basic>
             <basic name="topicTime">
                 <column name="TOPIC_TIME" />
                 <temporal>DATE</temporal>
             </basic>
             <basic name="topicViews">
                 <column name="TOPIC_VIEWS" />
             </basic>
         </attributes>
     </entity>
     <entity class="PollTopic">
②PollTopic 实体配置
         <discriminator-value>2</discriminator-value>
         <attributes>
             <basic name="maxChoices">
                 <column name="MAX_CHOICES" />
             </basic>
             <one-to-many name="options" mapped-by="pollTopic">
                 <cascade>
                     <cascade-all/>
                 </cascade>
             </one-to-many>
         </attributes>
     </entity>
     <entity class="PollOption">
②PollOption 实体配置
           <table name="T_POLL_OPTION" />
         <attributes>
             <id name="optionId">
                 <column name="OPTION_ID" />
                 <generated-value strategy="TABLE" />
             </id>
             <basic name="optionItem">
```

```
                <column name="OPTION_ITEM"/>
            </basic>
            <many-to-one name="pollTopic" >
                <join-column name="TOPIC_ID" nullable="false"/>
            </many-to-one>
        </attributes>
    </entity>
    <entity class="Post">
②Post 实体配置
        <table name="T_POST" />
 <attributes>
            <id name="postId">
                <column name="POST_ID" />
                <generated-value strategy="TABLE" />
            </id>
            <basic name="postText" fetch="EAGER">
                <column name="POST_TEXT" column-definition="LONGTEXT
NOT NULL"/>
                <lob/>
            </basic>
            <basic name="postAttach" fetch="LAZY">
                <column name="POST_ATTACH" column-definition="BLOB"/>
                <lob/>
            </basic>
        </attributes>
    </entity>
 </entity-mappings>
```

从代码清单 6 中，我们可以看出 PollTopic 并不需要通过特殊配置指定和 Topic 的继承关系，这些信息将从实体类反射信息获取。所以从严格意义上来说，元数据信息或 XML 和实体类结构信息共同构成的。

5.JPA的编程结构及重要的API

JavaEE 5.0 中所定义的JPA接口个数并不多，它们位于javax.persistence和javax.persistence.spi两个包中。 javax.persistence包中大部分API都是注解类，除此之外还包括EntityManager、Query等持久化操作接口。而 javax.persistence.spi包中的 4 个API，是JPA的服务层接口。下面，我们就来认识一下这些重要的接口。

EntityManager的类型

实体对象由实体管理器进行管理，JPA使用javax.persistence.EntityManager代表实体管理器。实体管理器和持久化上下文关联，持久化上下文是一系列实体的管理环境，我们通过EntityManager和持久化上下文进行交互。

有两种类型的实体管理器：

容器型：容器型的实体管理器由容器负责实体管理器之间的协作，在一个JTA事务中，一个实体管理器的持久化上下文的状态会自动广播到所有使用EntityManager的应用程序组件中。Java EE应用服务器提供的就是管理型的实体管理器；

应用程序型：实体管理器的生命周期由应用程序控制，应用程序通过javax.persistence.EntityManagerFactory的createEntityManager创建EntityManager实例。

EntityManager 的创建过程

javax.persistence.spi.PersistenceProvider 接口由 JPA 的实现者提供，该接口由启动者调用，以便创建一个 EntityManagerFactory 实例。它定义了创建一个 EntityManagerFactory 实例的方法：

EntityManagerFactorycreateContainerEntityManagerFactory(PersistenceUnitInfo info, Map map)

javax.persistence.spi.PersistenceUnitInfo 入参提供了创建实体管理器所需要的所有信息，这些信息根据 JPA 的规范，必须放置在 META-INF/persistence.xml 文件中。

PersistenceUnitInfo 接口拥有了一个 void addTransformer(ClassTransformer transformer)方法，通过该方式可以添加一个 javax.persistence.spi.ClassTransformer，并通过 PersistenceProvider 开放给容器，以便容器在实体类文件加载到 JVM 之前进行代码的增强，使元数据生效。JPA 厂商负责提供 ClassTransformer 接口的实现。

图 4 描述了创建 EntityManager 的过程：

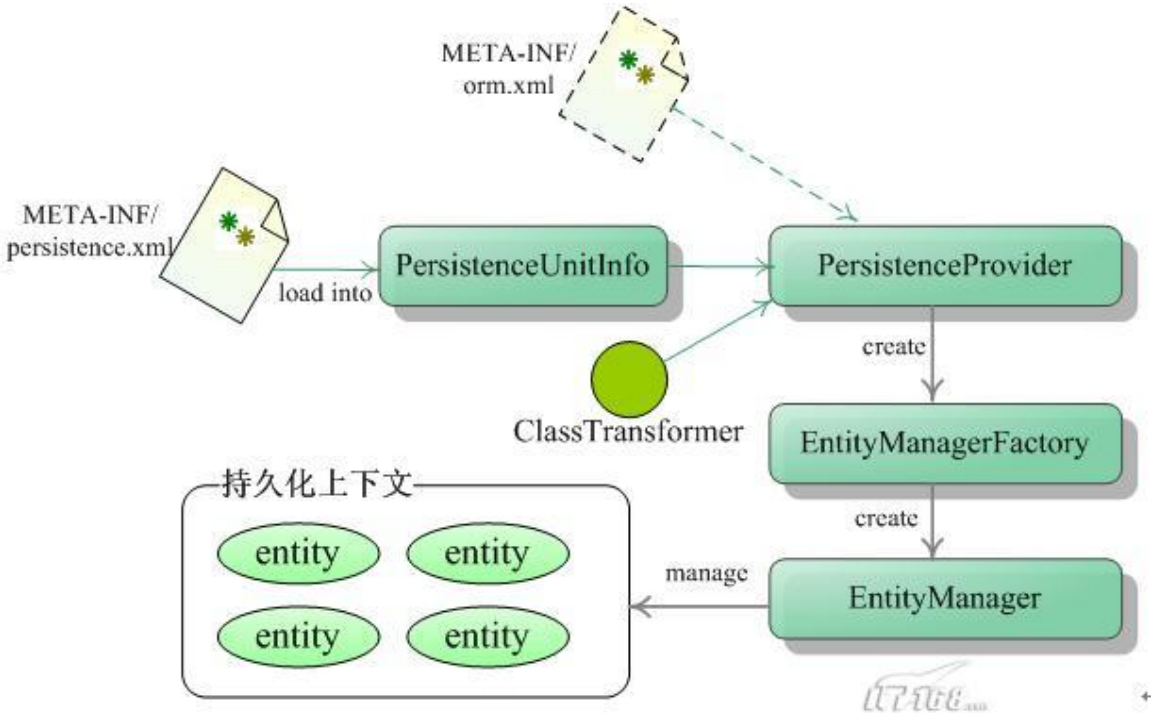


图 4 创建 EntityManager 的过程

实体的状态

实体对象拥有以下 4 个状态，这些状态通过调用 EntityManager 接口方法发生迁移：

新建态：新创建的实体对象，尚未拥有持久化主键，没有和一个持久化上下文关联起来。

受控态：已经拥有持久化主键并和持久化上下文建立了联系；

游离态：拥有持久化主键，但尚未和持久化上下文建立联系；

删除态：拥有持久化主键，已经和持久化上下文建立联系，但已经被安排从数据库中删除。

EntityManager 的 API

下面是 EntityManager 的一些主要的接口方法：

void persist(Object entity)

通过调用 EntityManager 的 persist()方法，新实体实例将转换为受控状态。这意谓着当 persist ()方法所在的事务提交时，实体的数据将保存到数据库中。如果实体已经被持久化，那么调用 persist()操作不会发生任何事情。如果对一个已经删除的 实体调用 persist()操作，删除态的实体又转变为受控态。如果对游离状的实体执行 persist()操作，将抛出 IllegalArgumentException。

在一个实体上调用 persist()操作，将广播到和实体关联的实体上，执行相应的级联持久化操作；

void remove(Object entity)

通过调用 remove()方法删除一个受控的实体。如果实体声明为级联删除(cascade=REMOVE 或者 cascade=ALL)，被关联的实体也会被删除。在一个新建状态的实体上调用 remove()操作，将被忽略。如果在游离实体上调用 remove()操作，将抛出 IllegalArgumentException，相关的事务将回滚。如果在已经删除的实体上执行 remove()操作，也会被忽略；

void flush()

将受控态的实体数据同步到数据库中；

T merge(T entity)

将一个游离态的实体持久化到数据库中，并转换为受控态的实体；

T find(Class entityClass, Object primaryKey)

以主键查询实体对象，entityClass 是实体的类，primaryKey 是主键值，如以下的代码查询 Topic 实体：

Topic t = em.find(Topic.class,1);

Query createQuery(String qlString)

根据 JPA 的查询语句创建一个查询对象 Query，如下面的代码：

```
Query q= em.createQuery(""SELECT t FROM Topic t

  WHERE t.topicTitle LIKE :topicTitle")");

Query createNativeQuery(String sqlString)
```

使用本地数据库的 SQL 语句创建一个 Query 对象，Query 通过 getResultList()方法执行查询后，返回一个 List 结果集，每一行数据对应一个 Vector。

Query

JPA 使用 javax.persistence.Query 接口代表一个查询实例，Query 实例由 EntityManager 通过指定查询语句构建。该接口拥有众多执行数据查询的接口方法：

◆Object getSingleResult()：执行 SELECT 查询语句，并返回一个结果；

◆List getResultList() ：执行 SELECT 查询语句，并返回多个结果；

◆Query setParameter(int position, Object value)：通过参数位置号绑定查询语句中的参数，如果查询语句使用了命令参数，则可以使用 Query setParameter(String name, Object value)方法绑定命名参数；

◆Query setMaxResults(int maxResult)：设置返回的最大结果数；

◆int executeUpdate()：如果查询语句是新增、删除或更改的语句，通过该方法执行更新操作；

6.JPA 的查询语言

JPA 的查询语言是面向对象而非面向数据库的，它以面向对象的自然语法构造查询语句，可以看成是 Hibernate HQL 的等价物。

简单的查询

你可以使用以下语句返回所有 Topic 对象的记录：

SELECT t FROM Topic t

t 表示 Topic 的别名，在 Topic t 是 Topic AS t 的缩写。

如果需要按条件查询 Topic，我们可以使用：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = ?1

通过 WHERE 指定查询条件，?1 表示用位置标识参数，尔后，我们可以通过 Query 的 setParameter(1, "主题 1")绑定参数。而 DISTINCT 表示过滤掉重复的数据。

如果需要以命名绑定绑定数据，可以改成以下的方式：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = :title

这时，需要通过 Query 的 setParameter("title", "主题 1")绑定参数。

关联查询：从 One 的一方关联到 Many 的一方

返回 PollOptions 对应的 PollTopic 对象，可以使用以下语句：

SELECT DISTINCT p FROM PollTopic p, IN(p.options) o WHERE o.optionItem LIKE ?1

这个语法和 SQL 以及 HQL 都有很大的区别，它直接实体属性连接关联的实体，这里我们通过 PollTopic 的 options 属性关联到 PollOption 实体上，对应的 SQL 语句为：

```
SELECT DISTINCT t0.TOPIC_ID, t0.TOPIC_TYPE,
t0.TOPIC_TITLE,

t0.TOPIC_TIME, t0.TOPIC_VIEWS, t0.MULTIPLE,
t0.MAX_CHOICES FROM T_TOPIC t0,

T_POLL_OPTION t1 WHERE (((t1.OPTION_ITEM LIKE ?)
AND (t0.TOPIC_TYPE = ?))

AND (t1.TOPIC_ID = t0.TOPIC_ID))
```

该查询语句的另外两种等价的写法分别是：

```
SELECT DISTINCT p FROM PollTopic p JOIN p.options
o WHERE o.optionItem LIKE ?1
```

和

```
SELECT DISTINCT p FROM PollTopic p WHERE
p.options.optionItem LIKE ?1
```

关联查询：从 Many 的一方关联到 One 的一方

从 Many 一方关联到 One 一方的查询语句和前面所讲的也很相似。如我们希望查询某一个调查主题下的所示调查项，则可以编写以下的查询语句：

```
SELECT p FROM PollOption p JOIN p.pollTopic t
WHERE t.topicId = :topicId
```

对应的 SQL 语句为：

```
SELECT t0.OPTION_ID, t0.OPTION_ITEM, t0.TOPIC_ID
```

```
FROM T_POLL_OPTION t0,

  TOPIC t1 WHERE ((t1.TOPIC_ID = ?)

AND ((t1.TOPIC_ID = t0.TOPIC_ID) AND
(t1.TOPIC_TYPE = ?)))
```

使用其它的关系操作符

使用空值比较符，比如查询附件不空的所有帖子对象：

```
SELECT p FROM Post p WHERE p.postAttach IS NOT
NULL
```

范围比较符包括 BETWEEN..AND 和>、>= 、<、<=、<>这些操作符。比如下面的语句查询浏览次数在 100 到 200 之间的所有论坛主题：

SELECT t FROM Topic t WHERE t.topicViews BETWEEN 100 AND 200

Query

JPA 使用 javax.persistence.Query 接口代表一个查询实例，Query 实例由 EntityManager 通过指定查询语句构建。该接口拥有众多执行数据查询的接口方法：

◆Object getSingleResult()：执行 SELECT 查询语句，并返回一个结果；

◆List getResultList() ：执行 SELECT 查询语句，并返回多个结果；

◆Query setParameter(int position, Object value)：通过参数位置号绑定查询语句中的参数，如果查询语句使用了命令参数，则可以使用 Query setParameter(String name, Object value)方法绑定命名参数；

◆Query setMaxResults(int maxResult)：设置返回的最大结果数；

◆int executeUpdate()：如果查询语句是新增、删除或更改的语句，通过该方法执行更新操作；

6.JPA 的查询语言

JPA 的查询语言是面向对象而非面向数据库的，它以面向对象的自然语法构造查询语句，可以看成是 Hibernate HQL 的等价物。

简单的查询

你可以使用以下语句返回所有 Topic 对象的记录：

SELECT t FROM Topic t

t 表示 Topic 的别名，在 Topic t 是 Topic AS t 的缩写。

如果需要按条件查询 Topic，我们可以使用：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = ?1

通过 WHERE 指定查询条件，?1 表示用位置标识参数，尔后，我们可以通过 Query 的 setParameter(1, "主题 1")绑定参数。而 DISTINCT 表示过滤掉重复的数据。

如果需要以命名绑定绑定数据，可以改成以下的方式：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = :title

这时，需要通过 Query 的 setParameter("title", "主题 1")绑定参数。

关联查询：从 One 的一方关联到 Many 的一方

返回 PollOptions 对应的 PollTopic 对象，可以使用以下语句：

SELECT DISTINCT p FROM PollTopic p, IN(p.options) o WHERE o.optionItem LIKE ?1

这个语法和 SQL 以及 HQL 都有很大的区别，它直接实体属性连接关联的实体，这里我们通过 PollTopic 的 options 属性关联到 PollOption 实体上，对应的 SQL 语句为：

```
SELECT DISTINCT t0.TOPIC_ID, t0.TOPIC_TYPE,
t0.TOPIC_TITLE,

t0.TOPIC_TIME, t0.TOPIC_VIEWS, t0.MULTIPLE,
t0.MAX_CHOICES FROM T_TOPIC t0,

T_POLL_OPTION t1 WHERE (((t1.OPTION_ITEM LIKE ?)
AND (t0.TOPIC_TYPE = ?))

AND (t1.TOPIC_ID = t0.TOPIC_ID))
```

该查询语句的另外两种等价的写法分别是：

```
SELECT DISTINCT p FROM PollTopic p JOIN p.options
o WHERE o.optionItem LIKE ?1
```

和

```
SELECT DISTINCT p FROM PollTopic p WHERE
p.options.optionItem LIKE ?1
```

关联查询：从 Many 的一方关联到 One 的一方

从 Many 一方关联到 One 一方的查询语句和前面所讲的也很相似。如我们希望查询某一个调查主题下的所示调查项，则可以编写以下的查询语句：

```
SELECT p FROM PollOption p JOIN p.pollTopic t
WHERE t.topicId = :topicId
```

对应的 SQL 语句为：

```
SELECT t0.OPTION_ID, t0.OPTION_ITEM, t0.TOPIC_ID
FROM T_POLL_OPTION t0,

  TOPIC t1 WHERE ((t1.TOPIC_ID = ?)

AND ((t1.TOPIC_ID = t0.TOPIC_ID) AND
(t1.TOPIC_TYPE = ?)))
```

使用其它的关系操作符

使用空值比较符，比如查询附件不空的所有帖子对象：

```
SELECT p FROM Post p WHERE p.postAttach IS NOT
NULL
```

范围比较符包括 BETWEEN..AND 和>、>= 、<、<=、<>这些操作符。比如下面的语句查询浏览次数在 100 到 200 之间的所有论坛主题：

SELECT t FROM Topic t WHERE t.topicViews BETWEEN 100 AND 200

Query

JPA 使用 javax.persistence.Query 接口代表一个查询实例，Query 实例由 EntityManager 通过指定查询语句构建。该接口拥有众多执行数据查询的接口方法：

◆Object getSingleResult()：执行 SELECT 查询语句，并返回一个结果；

◆List getResultList() ：执行 SELECT 查询语句，并返回多个结果；

◆Query setParameter(int position, Object value)：通过参数位置号绑定查询语句中的参数，如果查询语句使用了命令参数，则可以使用 Query setParameter(String name, Object value)方法绑定命名参数；

◆Query setMaxResults(int maxResult)：设置返回的最大结果数；

◆int executeUpdate()：如果查询语句是新增、删除或更改的语句，通过该方法执行更新操作；

6.JPA 的查询语言

JPA 的查询语言是面向对象而非面向数据库的，它以面向对象的自然语法构造查询语句，可以看成是 Hibernate HQL 的等价物。

简单的查询

你可以使用以下语句返回所有 Topic 对象的记录：

SELECT t FROM Topic t

t 表示 Topic 的别名，在 Topic t 是 Topic AS t 的缩写。

如果需要按条件查询 Topic，我们可以使用：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = ?1

通过 WHERE 指定查询条件，?1 表示用位置标识参数，尔后，我们可以通过 Query 的 setParameter(1, "主题 1")绑定参数。而 DISTINCT 表示过滤掉重复的数据。

如果需要以命名绑定绑定数据，可以改成以下的方式：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = :title

这时，需要通过 Query 的 setParameter("title", "主题 1")绑定参数。

关联查询：从 One 的一方关联到 Many 的一方

返回 PollOptions 对应的 PollTopic 对象，可以使用以下语句：

SELECT DISTINCT p FROM PollTopic p, IN(p.options) o WHERE o.optionItem LIKE ?1

这个语法和 SQL 以及 HQL 都有很大的区别，它直接实体属性连接关联的实体，这里我们通过 PollTopic 的 options 属性关联到 PollOption 实体上，对应的 SQL 语句为：

```
SELECT DISTINCT t0.TOPIC_ID, t0.TOPIC_TYPE,
t0.TOPIC_TITLE,

t0.TOPIC_TIME, t0.TOPIC_VIEWS, t0.MULTIPLE,
t0.MAX_CHOICES FROM T_TOPIC t0,

T_POLL_OPTION t1 WHERE (((t1.OPTION_ITEM LIKE ?)
AND (t0.TOPIC_TYPE = ?))

AND (t1.TOPIC_ID = t0.TOPIC_ID))
```

该查询语句的另外两种等价的写法分别是：

```
SELECT DISTINCT p FROM PollTopic p JOIN p.options
o WHERE o.optionItem LIKE ?1
```

和

```
SELECT DISTINCT p FROM PollTopic p WHERE
p.options.optionItem LIKE ?1
```

关联查询：从 Many 的一方关联到 One 的一方

从 Many 一方关联到 One 一方的查询语句和前面所讲的也很相似。如我们希望查询某一个调查主题下的所示调查项，则可以编写以下的查询语句：

```
SELECT p FROM PollOption p JOIN p.pollTopic t
WHERE t.topicId = :topicId
```

对应的 SQL 语句为：

```
SELECT t0.OPTION_ID, t0.OPTION_ITEM, t0.TOPIC_ID
FROM T_POLL_OPTION t0,

  TOPIC t1 WHERE ((t1.TOPIC_ID = ?)

AND ((t1.TOPIC_ID = t0.TOPIC_ID) AND
(t1.TOPIC_TYPE = ?)))
```

使用其它的关系操作符

使用空值比较符，比如查询附件不空的所有帖子对象：

```
SELECT p FROM Post p WHERE p.postAttach IS NOT
NULL
```

范围比较符包括 BETWEEN..AND 和>、>= 、<、<=、<>这些操作符。比如下面的语句查询浏览次数在 100 到 200 之间的所有论坛主题：

SELECT t FROM Topic t WHERE t.topicViews BETWEEN 100 AND 200

Query

JPA 使用 javax.persistence.Query 接口代表一个查询实例，Query 实例由 EntityManager 通过指定查询语句构建。该接口拥有众多执行数据查询的接口方法：

◆Object getSingleResult()：执行 SELECT 查询语句，并返回一个结果；

◆List getResultList() ：执行 SELECT 查询语句，并返回多个结果；

◆Query setParameter(int position, Object value)：通过参数位置号绑定查询语句中的参数，如果查询语句使用了命令参数，则可以使用 Query setParameter(String name, Object value)方法绑定命名参数；

◆Query setMaxResults(int maxResult)：设置返回的最大结果数；

◆int executeUpdate()：如果查询语句是新增、删除或更改的语句，通过该方法执行更新操作；

6.JPA 的查询语言

JPA 的查询语言是面向对象而非面向数据库的，它以面向对象的自然语法构造查询语句，可以看成是 Hibernate HQL 的等价物。

简单的查询

你可以使用以下语句返回所有 Topic 对象的记录：

SELECT t FROM Topic t

t 表示 Topic 的别名，在 Topic t 是 Topic AS t 的缩写。

如果需要按条件查询 Topic，我们可以使用：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = ?1

通过 WHERE 指定查询条件，?1 表示用位置标识参数，尔后，我们可以通过 Query 的 setParameter(1, "主题 1")绑定参数。而 DISTINCT 表示过滤掉重复的数据。

如果需要以命名绑定绑定数据，可以改成以下的方式：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = :title

这时，需要通过 Query 的 setParameter("title", "主题 1")绑定参数。

关联查询：从 One 的一方关联到 Many 的一方

返回 PollOptions 对应的 PollTopic 对象，可以使用以下语句：

SELECT DISTINCT p FROM PollTopic p, IN(p.options) o WHERE o.optionItem LIKE ?1

这个语法和 SQL 以及 HQL 都有很大的区别，它直接实体属性连接关联的实体，这里我们通过 PollTopic 的 options 属性关联到 PollOption 实体上，对应的 SQL 语句为：

```
SELECT DISTINCT t0.TOPIC_ID, t0.TOPIC_TYPE,
t0.TOPIC_TITLE,

t0.TOPIC_TIME, t0.TOPIC_VIEWS, t0.MULTIPLE,
t0.MAX_CHOICES FROM T_TOPIC t0,

T_POLL_OPTION t1 WHERE (((t1.OPTION_ITEM LIKE ?)
AND (t0.TOPIC_TYPE = ?))

AND (t1.TOPIC_ID = t0.TOPIC_ID))
```

该查询语句的另外两种等价的写法分别是：

```
SELECT DISTINCT p FROM PollTopic p JOIN p.options
o WHERE o.optionItem LIKE ?1
```

和

```
SELECT DISTINCT p FROM PollTopic p WHERE
p.options.optionItem LIKE ?1
```

关联查询：从 Many 的一方关联到 One 的一方

从 Many 一方关联到 One 一方的查询语句和前面所讲的也很相似。如我们希望查询某一个调查主题下的所示调查项，则可以编写以下的查询语句：

```
SELECT p FROM PollOption p JOIN p.pollTopic t
WHERE t.topicId = :topicId
```

对应的 SQL 语句为：

```
SELECT t0.OPTION_ID, t0.OPTION_ITEM, t0.TOPIC_ID
FROM T_POLL_OPTION t0,

  TOPIC t1 WHERE ((t1.TOPIC_ID = ?)

AND ((t1.TOPIC_ID = t0.TOPIC_ID) AND
(t1.TOPIC_TYPE = ?)))
```

使用其它的关系操作符

使用空值比较符，比如查询附件不空的所有帖子对象：

```
SELECT p FROM Post p WHERE p.postAttach IS NOT
NULL
```

范围比较符包括 BETWEEN..AND 和>、>= 、<、<=、<>这些操作符。比如下面的语句查询浏览次数在 100 到 200 之间的所有论坛主题：

SELECT t FROM Topic t WHERE t.topicViews BETWEEN 100 AND 200

Query

JPA 使用 javax.persistence.Query 接口代表一个查询实例，Query 实例由 EntityManager 通过指定查询语句构建。该接口拥有众多执行数据查询的接口方法：

◆Object getSingleResult()：执行 SELECT 查询语句，并返回一个结果；

◆List getResultList() ：执行 SELECT 查询语句，并返回多个结果；

◆Query setParameter(int position, Object value)：通过参数位置号绑定查询语句中的参数，如果查询语句使用了命令参数，则可以使用 Query setParameter(String name, Object value)方法绑定命名参数；

◆Query setMaxResults(int maxResult)：设置返回的最大结果数；

◆int executeUpdate()：如果查询语句是新增、删除或更改的语句，通过该方法执行更新操作；

6.JPA 的查询语言

JPA 的查询语言是面向对象而非面向数据库的，它以面向对象的自然语法构造查询语句，可以看成是 Hibernate HQL 的等价物。

简单的查询

你可以使用以下语句返回所有 Topic 对象的记录：

SELECT t FROM Topic t

t 表示 Topic 的别名，在 Topic t 是 Topic AS t 的缩写。

如果需要按条件查询 Topic，我们可以使用：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = ?1

通过 WHERE 指定查询条件，?1 表示用位置标识参数，尔后，我们可以通过 Query 的 setParameter(1, "主题 1")绑定参数。而 DISTINCT 表示过滤掉重复的数据。

如果需要以命名绑定绑定数据，可以改成以下的方式：

SELECT DISTINCT t FROM Topic t WHERE t.topicTitle = :title

这时，需要通过 Query 的 setParameter("title", "主题 1")绑定参数。

关联查询：从 One 的一方关联到 Many 的一方

返回 PollOptions 对应的 PollTopic 对象，可以使用以下语句：

SELECT DISTINCT p FROM PollTopic p, IN(p.options) o WHERE o.optionItem LIKE ?1

这个语法和 SQL 以及 HQL 都有很大的区别，它直接实体属性连接关联的实体，这里我们通过 PollTopic 的 options 属性关联到 PollOption 实体上，对应的 SQL 语句为：

```
SELECT DISTINCT t0.TOPIC_ID, t0.TOPIC_TYPE,
t0.TOPIC_TITLE,

t0.TOPIC_TIME, t0.TOPIC_VIEWS, t0.MULTIPLE,
t0.MAX_CHOICES FROM T_TOPIC t0,

T_POLL_OPTION t1 WHERE (((t1.OPTION_ITEM LIKE ?)
```

```
AND (t0.TOPIC_TYPE = ?))

AND (t1.TOPIC_ID = t0.TOPIC_ID))
```

该查询语句的另外两种等价的写法分别是：

```
SELECT DISTINCT p FROM PollTopic p JOIN p.options
o WHERE o.optionItem LIKE ?1
```

和

```
SELECT DISTINCT p FROM PollTopic p WHERE
p.options.optionItem LIKE ?1
```

关联查询：从 Many 的一方关联到 One 的一方

从 Many 一方关联到 One 一方的查询语句和前面所讲的也很相似。如我们希望查询某一个调查主题下的所示调查项，则可以编写以下的查询语句：

```
SELECT p FROM PollOption p JOIN p.pollTopic t
WHERE t.topicId = :topicId
```

对应的 SQL 语句为：

```
SELECT t0.OPTION_ID, t0.OPTION_ITEM, t0.TOPIC_ID
FROM T_POLL_OPTION t0,

  TOPIC t1 WHERE ((t1.TOPIC_ID = ?)

AND ((t1.TOPIC_ID = t0.TOPIC_ID) AND
(t1.TOPIC_TYPE = ?)))
```

使用其它的关系操作符

使用空值比较符，比如查询附件不空的所有帖子对象：

```
SELECT p FROM Post p WHERE p.postAttach IS NOT
NULL
```

范围比较符包括 BETWEEN..AND 和>、>= 、<、<=、<>这些操作符。比如下面的语句查询浏览次数在 100 到 200 之间的所有论坛主题：

SELECT t FROM Topic t WHERE t.topicViews BETWEEN 100 AND 200

集合关系操作符

和其它实体是 One-to-Many 或 Many-to-Many 关系的实体，通过集合引用关联的实体，我们可以通过集合关系操作符进行数据查询。下面的语句返回所有没有选项的调查论坛的主题：

SELECT t FROM PollTopic t WHERE t.options IS EMPTY

我们还可以通过判断元素是否在集合中进行查询：

SELECT t FROM PollTopic t WHERE :option MEMBER OF t.options

这里参数必须绑定一个 PollOption 的对象，JPA 会自动将其转换为主键比较的 SQL 语句。

子查询

JPA 可以进行子查询，并支持几个常见的子查询函数：EXISTS、ALL、ANY。如下面的语句查询出拥有 6 个以上选项的调查主题：

```
SELECT t FROM PollTopic t WHERE (SELECT COUNT(o)
FROM t.options o) > 6
```

可用函数

JPA 查询支持一些常见的函数，其中可用的字符串操作函数有：

◆CONCAT(String, String)：合并字段串；

◆LENGTH(String)：求字段串的长度；

◆LOCATE(String, String [, start])：查询字段串的函数，第一个参数为需要查询的字段串，看它在出现在第二个参数字符串的哪个位置，start 表示从哪个位置开始查找，返回查 找到的位置，没有找到返回 0。如 LOCATE ('b1','a1b1c1',1)返回为 3；

◆SUBSTRING(String, start, length)：子字段串函数；

◆TRIM([[LEADING|TRAILING|BOTH] char) FROM] (String)：将字段串前后的特殊字符去除，可以通过选择决定具体的去除位置和字符；

◆LOWER(String)：将字符串转为小写；

◆UPPER(String)：将字符串转为大写。

数字操作函数有：

◆ABS(number)：求绝对值函数；

◆MOD(int, int)：求模的函数；

◆SQRT(double)：求平方函数；

◆SIZE(Collection)：求集合大小函数。

更改语句

可以用 EntityManager 进行实体的更新操作，也可以通过查询语言执行数据表的字段更新，记录删除的操作：

下面的语句将某一个调查选项更新为新的值：

UPDATE PollOption p SET p.optionItem = :value WHERE p.optionId = :optionId

我们使用 Query 接口的 executeUpdate()方法执行更新。下面的语句删除一条记录：

DELETE FROM PollOption p WHERE p.optionId = :optionId

排序和分组

我们还可以对查询进行排序和分组。如下面的语句查询出主题的发表时间大于某个时间值，返回的结果按浏览量降序排列：

```
SELECT t FROM Topic t WHERE t.topicTime > :time
ORDER BY t.topicViews DESC
```

下面的语句计算出每个调查主题对应的选项数目：

SELECT COUNT(p),p.pollTopic.topicId FROM PollOption p GROUP BY p.pollTopic.topicId

这里，我们使用了计算数量的聚集函数 COUNT()，其它几个可用的聚集函数分别为：

◆AVG：计算平均值，返回类型为 double；

◆MAX：计算最大值；

◆MIN：计算最小值；

◆SUM：计算累加和；

我们还可以通过 HAVING 对聚集结果进行条件过滤：

```
SELECT COUNT(p),p.pollTopic.topicId

FROM PollOption p GROUP BY p.pollTopic.topicId
HAVING

p.pollTopic.topicId IN(1,2,3)
```

7.小结

在不久的将来，Sun 可能会将 JPA 作为一个单独的 JSR 对待，同时 JPA 还可能作为 Java SE 的一部分。不过这些都不太重要，重要的是，我们现在已经可以在脱离容器的情况下、在 Java SE 应用中使用 JPA 了。

JPA 已经作为一项对象持久化的标准，不但可以获得 Java EE 应用服务器的支持，还可以直接在 Java SE 中使用。开发者将无需在现有多种 ORM 框架中艰难地选择，这对开发人员来说是个福音。