JAVA PERSISTENCE API

Ненко Табаков
Пламен Танов
Технологическо училище “Електронни системи”
Технически университет – София

9 февруари 2009

ЛИТЕРАТУРА
НЕОБХОДИМИ ПРОГРАМИ

- The Java EE 5 Tutorial - http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf
- Step-by-step tutorial - http://schuchert.wikispaces.com/JPA+Tutorial+1+-+Getting+Started
- Java API документация - http://java.sun.com/javase/6/docs/api/
- Eclipse - www.eclipse.org
- Apache Tomcat - http://tomcat.apache.org/

ВЪВЕДЕНИЕ

- Предоставя възможност за управление на обекти:
  - Съхранение
  - Извличане
  - Промяна
  - Триене
- Тези обекти се наричат единици (entity)
- В общия случай всеки:
  - клас се представя с таблица
  - член-променлива – с колона в таблицата
- Конкретната реализация на JPA и СУБД, която се използва, се определя от конфигурацията

ENTITY CLASSES

- Служат за съхранение на данни
- Представят се като ред в таблица(и)
- Към тях съществуват следните изисквания:
  - Трябва да е анотиран с `@javax.persistence.Entity`
  - Да съществува `public` или `protected` конструктор по подразбиране (без аргументи)
- Член-променливите, които съхраняват данните трябва да са:
  - Примитивни типове
  - Дата
  - Низ
  - Изброими типове (enum)
  - Други entity classes дефинирани от потребителя
  - И други
- Не се съхраняват член-променливи, които са `@Transient`

ПЪРВИЧЕН КЛЮЧ

- Всеки обект трябва да има уникален ключ
- Може да бъде прост (simple) или съставен (composite)
- Използва се анотация `@javax.persistence.Id`
- Автоматична стойност се генерира с `@javax.persistence.GeneratedValue`
  - Само за целочислени член-променливи

```
@Id
@GeneratedValue
private long id;
```

ТИПОВЕ ВРЪЗКИ

- Налични са четири типа връзки:
  - Един към един (`@javax.persistence.OneToOne`)
  - Един към много (`@javax.persistence.OneToMany`)
  - Много към един (`@javax.persistence.ManyToOne`)
  - Много към много (`@javax.persistence.ManyToMany`)
    - Създава се помощна таблица

```
public class ForumUser extends AbstractEntity {
    @OneToMany(cascade=REMOVE, mappedBy="author")
    private List<ForumPost> posts;
}

...
public class ForumPost extends AbstractEntity {
    @ManyToOne(optional=false)
    private ForumUser author;
}
```

ТИПОВЕ ВРЪЗКИ

- Притежаващата страна е тази, която е от страната на „много“
- Чрез нея се създава или премахва връзка между обекти
- Дали дадена връзка е задължителна се определя с `optional`
- Дали дадено действие да се изпълни и за другата страна във връзката се определя с `cascade`
- Подчинената страна (`inverse`) оказва член-променливата в притежаващата страна (`owning`) посредством `mappedBy`

```
public class ForumUser extends AbstractEntity {
    @OneToMany(cascade=REMOVE, mappedBy="author")
    private List<ForumPost> posts;
}

...
public class ForumPost extends AbstractEntity {
    @ManyToOne(optional=false)
    private ForumUser author;
}
```

НАСЛЕДЯВАНЕ

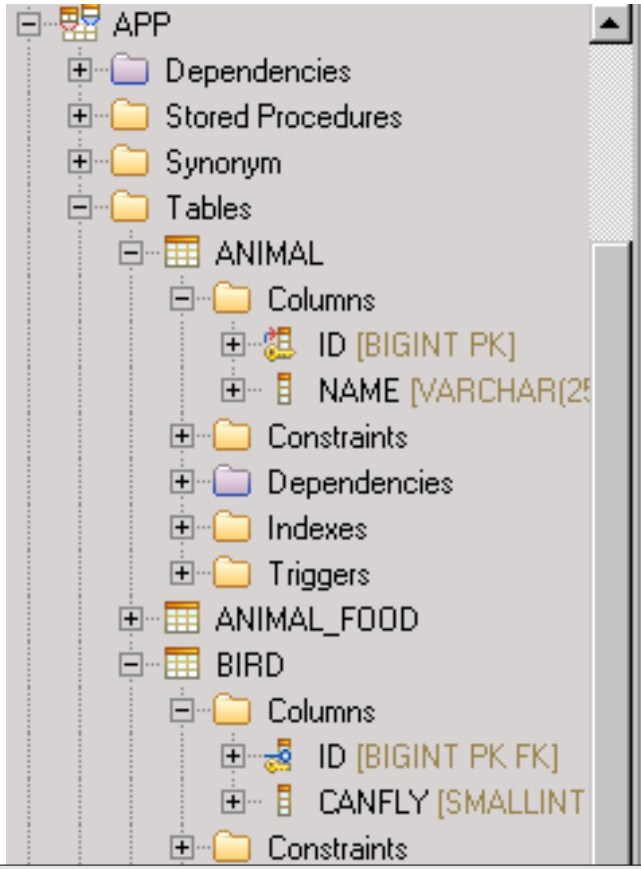
- Ако даден клас, който съхранява данни, се наследява от много други, но за него не трябва да се създава отделна таблица се ползва `@javax.persistence.MappedSuperclass`

```
@MappedSuperclass
public abstract class AbstractEntity {
    @Id
    @GeneratedValue
    private long id;

    public long getId() {
        return id;
    }
}

@Entity
public class Food extends AbstractEntity {}
```

НАСЛЕДЯВАНЕ

- За съхранение на данните при наследяване на класове са възможни няколко стратегии:
  - Отделна таблица за всяко едно дете от йерархията
    - `TABLE_PER_CLASS`
    - Поддръжката ѝ не е задължителна
  - Таблица за всеки клас от йерархията
    - `JOINED`
    - Таблиците за децата от йерархията съдържат само новите полета
  - Една таблица за цялата йерархия от класове
    - `SINGLE_TABLE`
    - По подразбиране
    - Стойностите, които не присъстват за даден тип са `null`
- Всяка една от тях има преимущества и недостатъци

ПРИМЕР




```
@Entity
public class Bird extends Animal {

    private boolean canFly;

    public Bird() {
        super(AnimalType.BIRD);
    }

    public void setCanFly(boolean canFly) {
        this.canFly = canFly;
    }

    public boolean isCanFly() {
        return canFly;
    }
}
```

```
@Entity
@Inheritance(strategy=InheritanceType.JOINED)
public class Animal extends AbstractEntity {}

@Entity
public class Bird extends Animal {}
```

УПРАВЛЕНИЕ

- За основните операции се използва интерфейс наречен `EntityManager`
- Ползва настройки от `META-INF/persistence.xml`
  - Описват се entity класовете
  - Настройки за конкретната реализация
- Предоставя набор от функции за:
  - Съхранение и промяна на обекти
    - `em.persist(entity);`
  - Извличане на обекти (по първичен ключ)
    - `em.find(type, id);`
  - Изтриване на обекти
    - `em.remove(entity);`
  - Поддръжка на транзакции
  - Изпълнение на заявки

ПОДДРЪЖКА НА ТРАНЗАКЦИИ

- Транзакция се отваря посредством `begin()`
- Транзакция се завършва посредством `commit()`
- Състоянието се възстановява посредством `rollback()`

```
synchronized public boolean removeEntity(AbstractEntity
                                                    entity) {
    final EntityTransaction tx = em.getTransaction();
    try {
        tx.begin();
        em.remove(entity);
        tx.commit();
        return true;
    } catch (final RuntimeException e) {
        return false;
    } finally {
        if (tx.isActive()) {
            tx.rollback();
        }
    }
}
```

ИЗПЪЛНЕНИЕ НА ЗАЯВКИ

- Служат за извличане, промяна и изтриване на данни
- Заявките биват два типа:
  - Именувани
  - Неименувани
- Могат да се указва максимален брой резултати
- Могат да се добавят параметри (добавяне на : пред името й)

```
Map<String, String> dbProperties = new HashMap<String,
                                                String>();

dbProperties.put("hibernate.connection.url", connectionUrl);
dbProperties.put("hibernate.hbm2ddl.auto", "create");

EntityManagerFactory emf = Persistence.
   createEntityManagerFactory("SimpleForumJSF", dbProperties);

EntityManager em = emf.createEntityManager();
em.createNamedQuery("getPosts").setMaxResults(10).
                                getResultList();
```

Допълнителни настройки, характерни за JPA реализацията

ПРИМЕР НЕИМЕНУВАНИ ЗАЯВКИ

```
return (List<ForumPost>) em.createQuery(
          "SELECT p FROM Posts p WHERE p.author=:author")
          .setParameter("author", author).setMaxResults(10)
          .getResultList();
```

Задаване на стойност на параметър

Параметър

ИМЕНУВАНИ ЗАЯВКИ

- Задават се чрез анотации

```
@Entity(name="Posts")
@NamedQueries({
    @NamedQuery(
          name="getPosts",//get all posts
          query="SELECT p FROM Posts p"
          ),
    @NamedQuery(
          name="getPostsByAuthor",//get posts by author
          query="SELECT p FROM Posts p WHERE p.author=:author"
          ),
})
public class ForumPost extends AbstractEntity {}

...
return(List<ForumPost>)em.createNamedQuery("getPostsByAuthor").
          setParameter("author", author).
          getResultList();
```

Параметър

Задаване на стойност на параметър

ДОПЪЛНИТЕЛНИ ХАРАКТЕРИСТИКИ

•Посредством анотации може да се зададат различни характеристики на полетата и таблиците:

- Имена на таблици и колони
- Задължителни полета
- Уникални полета
- Дължина на поле
- и други

```
@Entity(name="Users")
public class ForumUser extends AbstractEntity {

    @Column(unique=true, nullable=false, length=25, name="usr")
    private String username;

...
}
```

ПРИМЕР
persistence.xml

```
<persistence version="1.0">
   <persistence-unit name="Biology">
      <class>biology.model.AbstractEntity</class>
      <class>biology.model.Animal</class>
      <exclude-unlisted-classes/>
      <properties>
         <property name="hibernate.show_sql" value="false" />
         <property name="hibernate.format_sql" value="false" />
         <property name="hibernate.connection.username"
                                                value="app" />
         <property name="hibernate.connection.autocommit"
                                              value="false" />
         <property name="hibernate.dialect"
                  value="org.hibernate.dialect.DerbyDialect" />
         <property name="hibernate.hbm2ddl.auto"
                                             value="validate" />
         <property name="hibernate.cache.provider_class"
                 value="org.hibernate.cache.NoCacheProvider" />
      </properties>
   </persistence-unit>
</persistence>
```