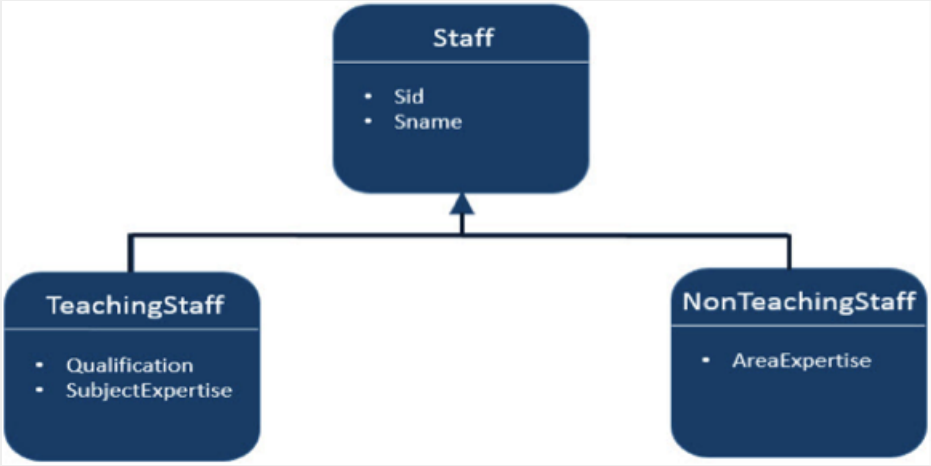
JPA - Advanced Mappings

JPA is a library which is released with java specification. Therefore, it supports all object oriented concepts for entity persistence. Till now we are done with the basics of object relational mapping. This chapter takes you through the advanced mappings between objects and relational entities.

Inheritance Strategies

Inheritance is the core concept of object oriented language, therefore we can use inheritance relationships or strategies between entities. JPA support three types of inheritance strategies such as SINGLE_TABLE, JOINED_TABLE, and TABLE_PER_CONCRETE_CLASS.

Let us consider an example of Staff, TeachingStaff, NonTeachingStaff classes and their relationships as follows:



In the above shown diagram Staff is an entity and TeachingStaff and NonTeachingStaff are the sub entities of Staff. Here we will discuss the above example in all three strategies of inheritance.

Single Table strategy

Single-Table strategy takes all classes fields (both super and sub classes) and map them down into a single table known as SINGLE_TABLE strategy. Here discriminator value plays key role in differentiating the values of three entities in one table.

Let us consider the above example, TeachingStaff and NonTeachingStaff are the sub classes of class Staff. Remind the concept of inheritance (is a mechanism of inheriting the properties of super class by sub class) and therefore sid, sname are the fields which belongs to both TeachingStaff and NonTeachingStaff. Create a JPA project. All the modules of this project as follows:

Creating Entities

Create a package named **'com.tutorialspoint.eclipselink.entity'** under **'src'** package. Create a new java class named **Staff.java** under given package. The Staff entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import java.io.Serializable;

import javax.persistence.DiscriminatorColumn;
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.Table;

@Entity
@Table
```

```
@Inheritance( strategy = InheritanceType.SINGLE_TABLE )
@DiscriminatorColumn( name = "type" )

public class Staff implements Serializable {
   @Id
   @GeneratedValue( strategy = GenerationType.AUTO )

   private int sid;
   private String sname;

   public Staff( int sid, String sname ) {
      super( );
      this.sid = sid;
      this.sname = sname;
   }

   public Staff( ) {
      super( );
   }

   public int getSid( ) {
      return sid;
   }

   public void setSid( int sid ) {
      this.sid = sid;
   }

   public String getSname( ) {
      return sname;
   }

   public void setSname( String sname ) {
      this.sname = sname;
   }
}
```

In the above code **@DescriminatorColumn** specifies the field name **(type)** and the values of it shows the remaining (Teaching and NonTeachingStaff) fields.

Create a subclass (class) to Staff class named **TeachingStaff.java** under the **com.tutorialspoint.eclipselink.entity** package. The TeachingStaff Entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue( value="TS" )
public class TeachingStaff extends Staff {

   private String qualification;
   private String subjectexpertise;

   public TeachingStaff( int sid, String sname,

   String qualification,String subjectexpertise ) {
      super( sid, sname );
      this.qualification = qualification;
      this.subjectexpertise = subjectexpertise;
```

```
    }

    public TeachingStaff( ) {
       super( );
    }

    public String getQualification( ){
       return qualification;
    }

    public void setQualification( String qualification ){
       this.qualification = qualification;
    }

    public String getSubjectexpertise( ) {
       return subjectexpertise;
    }

    public void setSubjectexpertise( String subjectexpertise ){
       this.subjectexpertise = subjectexpertise;
    }
 }
```

Create a subclass (class) to Staff class named **NonTeachingStaff.java** under the **com.tutorialspoint.eclipselink.entity** package. The NonTeachingStaff Entity class is shown as follows:

```
 package com.tutorialspoint.eclipselink.entity;

 import javax.persistence.DiscriminatorValue;
 import javax.persistence.Entity;

 @Entity
 @DiscriminatorValue( value = "NS" )

 public class NonTeachingStaff extends Staff {
    private String areaexpertise;

    public NonTeachingStaff( int sid, String sname, String areaexpertise ) {
       super( sid, sname );
       this.areaexpertise = areaexpertise;
    }

    public NonTeachingStaff( ) {
       super( );
    }

    public String getAreaexpertise( ) {
       return areaexpertise;
    }

    public void setAreaexpertise( String areaexpertise ){
       this.areaexpertise = areaexpertise;
    }
 }
```

Persistence.xml

Persistence.xml file contains the configuration information of database and registration information of entity classes. The xml file is shown as follows:

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

<persistence version="2.0" xmlns="http://java.sun.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_2_0.xsd">

   <persistence-unit name="Eclipselink_JPA" transaction-type="RESOURCE_LOCAL">

      <class>com.tutorialspoint.eclipselink.entity.Staff</class>
      <class>com.tutorialspoint.eclipselink.entity.NonTeachingStaff</class>
      <class>com.tutorialspoint.eclipselink.entity.TeachingStaff</class>

      <properties>
         <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/jpadb"/>
         <property name="javax.persistence.jdbc.user" value="root"/>
         <property name="javax.persistence.jdbc.password" value="root"/>
         <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
         <property name="eclipselink.logging.level" value="FINE"/>
         <property name="eclipselink.ddl-generation" value="create-tables"/>
      </properties>

   </persistence-unit>
</persistence>
```

Service class

Service classes are the implementation part of business component. Create a package under **'src'** package named **'com.tutorialspoint.eclipselink.service'**.

Create a class named SaveClient.java under the given package to store Staff, TeachingStaff, and NonTeachingStaff class fields. The SaveClient class is shown as follows:

```
package com.tutorialspoint.eclipselink.service;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import com.tutorialspoint.eclipselink.entity.NonTeachingStaff;
import com.tutorialspoint.eclipselink.entity.TeachingStaff;

public class SaveClient {

   public static void main( String[ ] args ) {

      EntityManagerFactory emfactory = Persistence.createEntityManagerFactory( "Eclipselink_JPA" );
      EntityManager entitymanager = emfactory.createEntityManager( );
      entitymanager.getTransaction( ).begin( );

      //Teaching staff entity
      TeachingStaff ts1=new TeachingStaff(1,"Gopal","MSc MEd","Maths");
      TeachingStaff ts2=new TeachingStaff(2, "Manisha", "BSc BEd", "English");

      //Non-Teaching Staff entity
      NonTeachingStaff nts1=new NonTeachingStaff(3, "Satish", "Accounts");
      NonTeachingStaff nts2=new NonTeachingStaff(4, "Krishna", "Office Admin");

      //storing all entities
      entitymanager.persist(ts1);
      entitymanager.persist(ts2);
      entitymanager.persist(nts1);
```

```
        entitymanager.persist(nts2);

        entitymanager.getTransaction().commit();

        entitymanager.close();
        emfactory.close();
    }
}
```

After compilation and execution of the above program you will get notifications in the console panel of Eclipse IDE. Check MySQL workbench for output. The output in a tabular format is shown as follows:

Sid	Type	Sname	Areaexpertise	Qualification	Subjectexpertise
1	TS	Gopal		MSC MED	Maths
2	TS	Manisha		BSC BED	English
3	NS	Satish	Accounts		
4	NS	Krishna	Office Admin		

Finally you will get single table which contains all three class's fields and differs with discriminator column named **'Type'** (field).

Joined table Strategy

Joined table strategy is to share the referenced column which contains unique values to join the table and make easy transactions. Let us consider the same example as above.

Create a JPA Project. All the project modules shown as follows:

Creating Entities

Create a package named **'com.tutorialspoint.eclipselink.entity'** under **'src'** package. Create a new java class named **Staff.java** under given package. The Staff entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import java.io.Serializable;

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.Table;

@Entity
@Table
@Inheritance( strategy = InheritanceType.JOINED )

public class Staff implements Serializable {

   @Id
   @GeneratedValue( strategy = GenerationType.AUTO )

   private int sid;
   private String sname;

   public Staff( int sid, String sname ) {
      super( );
      this.sid = sid;
```

```
      this.sname = sname;
   }

   public Staff( ) {
      super( );
   }

   public int getSid( ) {
      return sid;
   }

   public void setSid( int sid ) {
      this.sid = sid;
   }

   public String getSname( ) {
      return sname;
   }

   public void setSname( String sname ) {
      this.sname = sname;
   }
}
```

Create a subclass (class) to Staff class named **TeachingStaff.java** under the **com.tutorialspoint.eclipselink.entity** package. The TeachingStaff Entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@PrimaryKeyJoinColumn(referencedColumnName="sid")

public class TeachingStaff extends Staff {
   private String qualification;
   private String subjectexpertise;

   public TeachingStaff( int sid, String sname,

   String qualification,String subjectexpertise ) {
      super( sid, sname );
      this.qualification = qualification;
      this.subjectexpertise = subjectexpertise;
   }

   public TeachingStaff( ) {
      super( );
   }

   public String getQualification( ){
      return qualification;
   }

   public void setQualification( String qualification ){
      this.qualification = qualification;
   }

   public String getSubjectexpertise( ) {
      return subjectexpertise;
```

```
    }

    public void setSubjectexpertise( String subjectexpertise ){
       this.subjectexpertise = subjectexpertise;
    }
}
```

Create a subclass (class) to Staff class named **NonTeachingStaff.java** under the **com.tutorialspoint.eclipselink.entity** package. The NonTeachingStaff Entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@PrimaryKeyJoinColumn(referencedColumnName="sid")

public class NonTeachingStaff extends Staff {
   private String areaexpertise;

   public NonTeachingStaff( int sid, String sname, String areaexpertise ) {
      super( sid, sname );
      this.areaexpertise = areaexpertise;
   }

   public NonTeachingStaff( ) {
      super( );
   }

   public String getAreaexpertise( ) {
      return areaexpertise;
   }

   public void setAreaexpertise( String areaexpertise ) {
      this.areaexpertise = areaexpertise;
   }
}
```

Persistence.xml

Persistence.xml file contains the configuration information of database and registration information of entity classes. The xml file is shown as follows:

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

<persistence version = "2.0" xmlns = "http://java.sun.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_2_0.xsd">

   <persistence-unit name = "Eclipselink_JPA" transaction-type = "RESOURCE_LOCAL">
      <class>com.tutorialspoint.eclipselink.entity.Staff</class>
      <class>com.tutorialspoint.eclipselink.entity.NonTeachingStaff</class>
      <class>com.tutorialspoint.eclipselink.entity.TeachingStaff</class>

      <properties>
         <property name = "javax.persistence.jdbc.url" value = "jdbc:mysql://localhost:3306/jpadb"/>
         <property name = "javax.persistence.jdbc.user" value = "root"/>
         <property name = "javax.persistence.jdbc.password" value = "root"/>
```

```
         <property name = "javax.persistence.jdbc.driver" value = "com.mysql.jdbc.Driver"/>
         <property name = "eclipselink.logging.level" value = "FINE"/>
         <property name = "eclipselink.ddl-generation" value = "create-tables"/>
      </properties>

   </persistence-unit>
</persistence>
```

Service class

Service classes are the implementation part of business component. Create a package under **'src'** package named **'com.tutorialspoint.eclipselink.service'**.

Create a class named SaveClient.java under the given package to store Staff, TeachingStaff, and NonTeachingStaff class fields. Then SaveClient class as follows:

```
package com.tutorialspoint.eclipselink.service;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import com.tutorialspoint.eclipselink.entity.NonTeachingStaff;
import com.tutorialspoint.eclipselink.entity.TeachingStaff;

public class SaveClient {
   public static void main( String[ ] args ) {
      EntityManagerFactory emfactory = Persistence.createEntityManagerFactory( "Eclipselink_JPA" );
      EntityManager entitymanager = emfactory.createEntityManager( );
      entitymanager.getTransaction( ).begin( );

      //Teaching staff entity
      TeachingStaff ts1 = new TeachingStaff(1,"Gopal","MSc MEd","Maths");
      TeachingStaff ts2 = new TeachingStaff(2, "Manisha", "BSc BEd", "English");

      //Non-Teaching Staff entity
      NonTeachingStaff nts1 = new NonTeachingStaff(3, "Satish", "Accounts");
      NonTeachingStaff nts2 = new NonTeachingStaff(4, "Krishna", "Office Admin");

      //storing all entities
      entitymanager.persist(ts1);
      entitymanager.persist(ts2);
      entitymanager.persist(nts1);
      entitymanager.persist(nts2);

      entitymanager.getTransaction().commit();
      entitymanager.close();
      emfactory.close();
   }
}
```

After compilation and execution of the above program you will get notifications in the console panel of Eclipse IDE. For output check MySQL workbench as follows:

Here three tables are created and the result of **staff** table in a tabular format is shown as follows:

Sid	Dtype	Sname
1	TeachingStaff	Gopal
2	TeachingStaff	Manisha
3	NonTeachingStaff	Satish
4	NonTeachingStaff	Krishna

The result of **TeachingStaff** table in a tabular format is shown as follows:

Sid	Qualification	Subjectexpertise
1	MSC MED	Maths
2	BSC BED	English

In the above table sid is the foreign key (reference field form staff table) The result of **NonTeachingStaff** table in tabular format is shown as follows:

Sid	Areaexpertise
3	Accounts
4	Office Admin

Finally the three tables are created using their fields respectively and SID field is shared by all three tables. In staff table SID is primary key, in remaining (TeachingStaff and NonTeachingStaff) tables SID is foreign key.

Table per class strategy

Table per class strategy is to create a table for each sub entity. The staff table will be created but it will contain null records. The field values of Staff table must be shared by TeachingStaff and NonTeachingStaff tables.

Let us consider the same example as above. All modules of this project are shown as follows:

Creating Entities

Create a package named **'com.tutorialspoint.eclipselink.entity'** under **'src'** package. Create a new java class named **Staff.java** under given package. The Staff entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import java.io.Serializable;

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.Table;

@Entity
@Table
@Inheritance( strategy = InheritanceType.TABLE_PER_CLASS )

public class Staff implements Serializable {

   @Id
   @GeneratedValue( strategy = GenerationType.AUTO )

   private int sid;
```

```
   private String sname;

   public Staff( int sid, String sname ) {
      super( );
      this.sid = sid;
      this.sname = sname;
   }

   public Staff( ) {
      super( );
   }

   public int getSid( ) {
      return sid;
   }

   public void setSid( int sid ) {
      this.sid = sid;
   }

   public String getSname( ) {
      return sname;
   }

   public void setSname( String sname ) {
      this.sname = sname;
   }
}
```

Create a subclass (class) to Staff class named **TeachingStaff.java** under the **com.tutorialspoint.eclipselink.entity** package. The TeachingStaff Entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
public class TeachingStaff extends Staff {
   private String qualification;
   private String subjectexpertise;

   public TeachingStaff( int sid, String sname, String qualification, String subjectexpertise ) {
      super( sid, sname );
      this.qualification = qualification;
      this.subjectexpertise = subjectexpertise;
   }

   public TeachingStaff( ) {
      super( );
   }

   public String getQualification( ){
      return qualification;
   }

   public void setQualification( String qualification ) {
      this.qualification = qualification;
   }
```

```
   public String getSubjectexpertise( ) {
      return subjectexpertise;
   }

   public void setSubjectexpertise( String subjectexpertise ){
      this.subjectexpertise = subjectexpertise;
   }
}
```

Create a subclass (class) to Staff class named **NonTeachingStaff.java** under the **com.tutorialspoint.eclipselink.entity** package. The NonTeachingStaff Entity class is shown as follows:

```
package com.tutorialspoint.eclipselink.entity;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
public class NonTeachingStaff extends Staff {
   private String areaexpertise;

   public NonTeachingStaff( int sid, String sname, String areaexpertise ) {
      super( sid, sname );
      this.areaexpertise = areaexpertise;
   }

   public NonTeachingStaff( ) {
      super( );
   }

   public String getAreaexpertise( ) {
      return areaexpertise;
   }

   public void setAreaexpertise( String areaexpertise ) {
      this.areaexpertise = areaexpertise;
   }
}
```

Persistence.xml

Persistence.xml file contains the configuration information of database and registration information of entity classes. The xml file is shown as follows:

```
<?xml version="1.0" encoding = "UTF-8"?>
<persistence version = "2.0" xmlns = "http://java.sun.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_2_0.xsd">

   <persistence-unit name = "Eclipselink_JPA" transaction-type = "RESOURCE_LOCAL">
      <class>com.tutorialspoint.eclipselink.entity.Staff</class>
      <class>com.tutorialspoint.eclipselink.entity.NonTeachingStaff</class>
      <class>com.tutorialspoint.eclipselink.entity.TeachingStaff</class>

      <properties>
         <property name = "javax.persistence.jdbc.url" value = "jdbc:mysql://localhost:3306/jpadb"/>
         <property name = "javax.persistence.jdbc.user" value = "root"/>
         <property name = "javax.persistence.jdbc.password" value = "root"/>
         <property name = "javax.persistence.jdbc.driver" value = "com.mysql.jdbc.Driver"/>
```

```
            <property name = "eclipselink.logging.level" value = "FINE"/>
            <property name = "eclipselink.ddl-generation" value="create-tables"/>
         </properties>

      </persistence-unit>
   </persistence>
```

Service class

Service classes are the implementation part of business component. Create a package under **'src'** package named **'com.tutorialspoint.eclipselink.service'**.

Create a class named **SaveClient.java** under the given package to store Staff, TeachingStaff, and NonTeachingStaff class fields. The SaveClient class is shown as follows:

```
package com.tutorialspoint.eclipselink.service;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

import com.tutorialspoint.eclipselink.entity.NonTeachingStaff;
import com.tutorialspoint.eclipselink.entity.TeachingStaff;

public class SaveClient {
   public static void main( String[ ] args ) {
      EntityManagerFactory emfactory = Persistence.createEntityManagerFactory( "Eclipselink_JPA" );
      EntityManager entitymanager = emfactory.createEntityManager( );
      entitymanager.getTransaction( ).begin( );

      //Teaching staff entity
      TeachingStaff ts1 = new TeachingStaff(1,"Gopal","MSc MEd","Maths");
      TeachingStaff ts2 = new TeachingStaff(2, "Manisha", "BSc BEd", "English");

      //Non-Teaching Staff entity
      NonTeachingStaff nts1 = new NonTeachingStaff(3, "Satish", "Accounts");
      NonTeachingStaff nts2 = new NonTeachingStaff(4, "Krishna", "Office Admin");

      //storing all entities
      entitymanager.persist(ts1);
      entitymanager.persist(ts2);
      entitymanager.persist(nts1);
      entitymanager.persist(nts2);

      entitymanager.getTransaction().commit();
      entitymanager.close();
      emfactory.close();
   }
}
```

After compilation and execution of the above program you will get notifications in the console panel of Eclipse IDE. For output, check MySQL workbench as follows:

Here the three tables are created and the **Staff** table contains null records.

The result of **TeachingStaff** in a tabular format is shown as follows:

Sid	Qualification	Sname	Subjectexpertise
1	MSC MED	Gopal	Maths
2	BSC BED	Manisha	English

The above table TeachingStaff contains fields of both Staff and TeachingStaff Entities.

The result of **NonTeachingStaff** in a tabular format is shown as follows:

Sid	Areaexpertise	Sname
3	Accounts	Satish
4	Office Admin	Krishna

The above table NonTeachingStaff contains fields of both Staff and NonTeachingStaff Entities.