Using the Metamodel API to Model Entity Classes

The Metamodel API is used to create a metamodel of the managed entities in a particular persistence unit. For each entity class in a particular package, a metamodel class is created with a trailing underscore and with attributes that correspond to the persistent fields or properties of the entity class.

The following entity class, `com.example.Pet`, has four persistent fields: `id`,`name`, `color`, and `owners`:

```
package com.example;

...

@Entity
public class Pet {
  @Id
  protected Long id;
  protected String name;
  protected String color;
  @ManyToOne
  protected Set<Owner> owners;
  ...
}
```

The corresponding Metamodel class is:

```
package com.example;

...

@Static Metamodel(Pet.class)
public class Pet_ {

  public static volatile SingularAttribute<Pet, Long> id;
  public static volatile SingularAttribute<Pet, String> name;
  public static volatile SingularAttribute<Pet, String> color;
  public static volatile SetAttribute<Pet, Owner> owners;
}
```

The metamodel class and its attributes are used in Criteria queries to refer to the managed entity classes and their persistent state and relationships.

Using Metamodel Classes

Metamodel classes that correspond to entity classes are of the following type:

```
javax.persistence.metamodel.EntityType<T>
```

Metamodel classes are typically generated by annotation processors either at development time or at runtime. Developers of applications that use Criteria queries may generate static metamodel classes by using the persistence provider's annotation processor or may obtain the metamodel class by either calling the `getModel` method on the query root object or first obtaining an instance of the `Metamodel` interface and then passing the entity type to the instance's `entity` method.

The following code snippet shows how to obtain the `Pet` entity's metamodel class by calling `Root<T>.getModel`:

```
EntityManager em = ...;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery cq = cb.createQuery(Pet.class);
Root<Pet> pet = cq.from(Pet.class);
EntityType<Pet> Pet_ = pet.getModel();
```

The following code snippet shows how to obtain the `Pet` entity's metamodel class by first obtaining a metamodel instance by using`EntityManager.getMetamodel` and then calling `entity` on the metamodel instance:

```
EntityManager em = ...;
Metamodel m = em.getMetamodel();
```

```
EntityType<Pet> Pet_ = m.entity(Pet.class);
```

Note - The most common use case is to generate type-safe static metamodel classes at development time. Obtaining the metamodel classes dynamically, by
calling `Root<T>.getModel` or `EntityManager.getMetamodel` and then the `entity` method, doesn't allow for type-safety and doesn't allow the application to call persistent field or property names on the metamodel class.