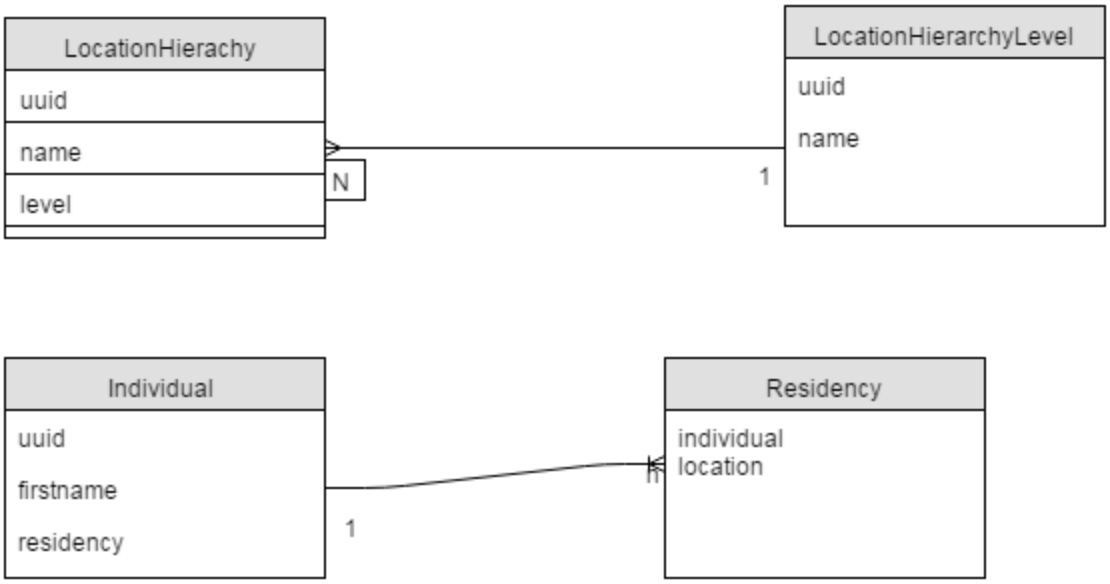
The Hibernate ORM is used to perform CRUD operations on the OpenHDS entities. In the hibernate.cfg.xml file the domain model classes like Individual, Visit etc are configured using the mapping tags to map the model with the corresponding database table. The connection attributes for the hibernate session is passed using the property tags. The hibernate.SessionFactory is used to build and retrieve the sessions with the database and to perform the CRUD operations. The GenericDao and Dao interfaces act like the gateway to the database. All the database related methods are encapsulated using these two interfaces. Both the Dao's have the different implementations. The GenericDao contains generic methods in order to perform update, delete, refresh, finders using the entityType. Whereas, the class BaseDao perform these operations using the primary keys.

The sessionFactory bean of type springframework.orm.hibernate3.annotation.AnnotationSessionFactory is configured in the daoApplicationContext in order to support the annotation based metadata for the mapping.The annotated class names are passed using the property tag. The Models are annotated with the @javax.persistence.Entity to indicate as the database entity. @Table annotation is used to map the entity with corresponding table in the database. For example: the individual class is annotated with @Entity which represents that individual is an entity. The table name individual represents that the individual entity is mapped to the individual table in the database .

```
@Entity
@Table(name=”individual”)
Class Individual()
{
}
```

Apart from the above JPA annotations at the class level, the annotations such as @NotNull, @Serachable, @Cascade, @ManyToOne, @OneToMany are set to represent the nature of the particular entity type. In OpenHDS, there are entities which have ManyToOne and OneToMany relationship between them. A ManyToOne relationship on the particular field of an entity class indicate that the many entities can have a relationships with the annotated entities.
for example many Memberships can have a same social group. In order to define this relationship the membership entity have the socialGroup field annotated with @ManyToOne. Similarly, many locations hierarchy can have same location hierarchy levels such as country, region etc.
A OneToMany relationship specifies that an entity can have many relationships with entity annotated with @OneToMany. An individual can have multiple residencies. Similarly a location can have multiple residencies.


LocationHierachy
uuid
name
level
N
1
LocationHierarchyLevel
uuid
name
Individual
uuid
firstname
residency
1
Residency
individual
location