```
@Entity
@Table(name="person")
class Person {
  @Id
  @GeneratedValue(strategy=
         GenerationType.IDENTITY)
  int id;
  String fullname;
  // accessors
}
```

```
@Entity
@Table(name="address")
class Address {
  @Id
  @GeneratedValue(strategy=
    GenerationType.IDENTITY)
  int addressNo;
  String addressLine1;
  // accessors
}
```

```
#Proxy / DAO
class CRUDInvocationHandler implements InvocationHandler {

  @Autowired
  private JpaTemplate jpaTemplate;


  public Object invoke(Object proxy, Method method, Object[] args) {
    if(method.eq("save")){
      jpaTemplate.persist(args[0]);
      return args[0];
    } else if(method.eq("update")) {
      jpaTemplate.merge(args[0]);
    }else if(method.eq("delete")) {
      jpaTemplate.remove(args[0]);
    }else if(method.eq("find")) {
      Object entity = jpaTemplate.find(proxy.getClass(), args[0]);
      return entity;
    }
  }
}
```

```
@Repository
class PersonDao {
  @Autowired
  JpaTemplate jpaTemplate;

  public int savePerson(Person person) {
    jpaTemplate.persist(person);
    return person.getId();
  }

  public void updatePerson(Person person){
    jpaTemplate.merge(person);
  }

  pubic void delete(int id) {
    Person person =
jpaTemplate.find(Person.class, id);
    jpaTemplate.remove(person);
  }
}
```

```
@Repository
class AddressDao {
  @Autowired
  JpaTemplate jpaTemplate;

  public int saveAddress(Address address){
    jpaTemplate.persist(address);
    return address.getAddressNo();
  }
  public void updateAddress(Address
address) {
    jpaTemplate.merge(address);
  }
  public void deleteAddress(int addressNo){
    Address address =
jpaTemplate.find(Address.class, addressNo);
    jpaTemplate.remove(address);
  }
}
```

```
interface PersonDao {
  Person save(Person person);
  int update(Person person);
  void delete(Person person);
  Person find(Class<?> classType, int id);
}
```

```
interface AddressDao {
  Address save(Address address);
  int update(Address address);
  void delete(Address address);
  Address find(Class<?> classType, long id);
}
```

```
PersonDao personDao = Proxy.newProxyInstance(PersonDao.class.getClassLoader(),
                                new Class<?>[] {Person.class}, new CRUDInvocationHandler());
```

```
Person person = new Person();
// populate data
person = personDao.save(person);
```