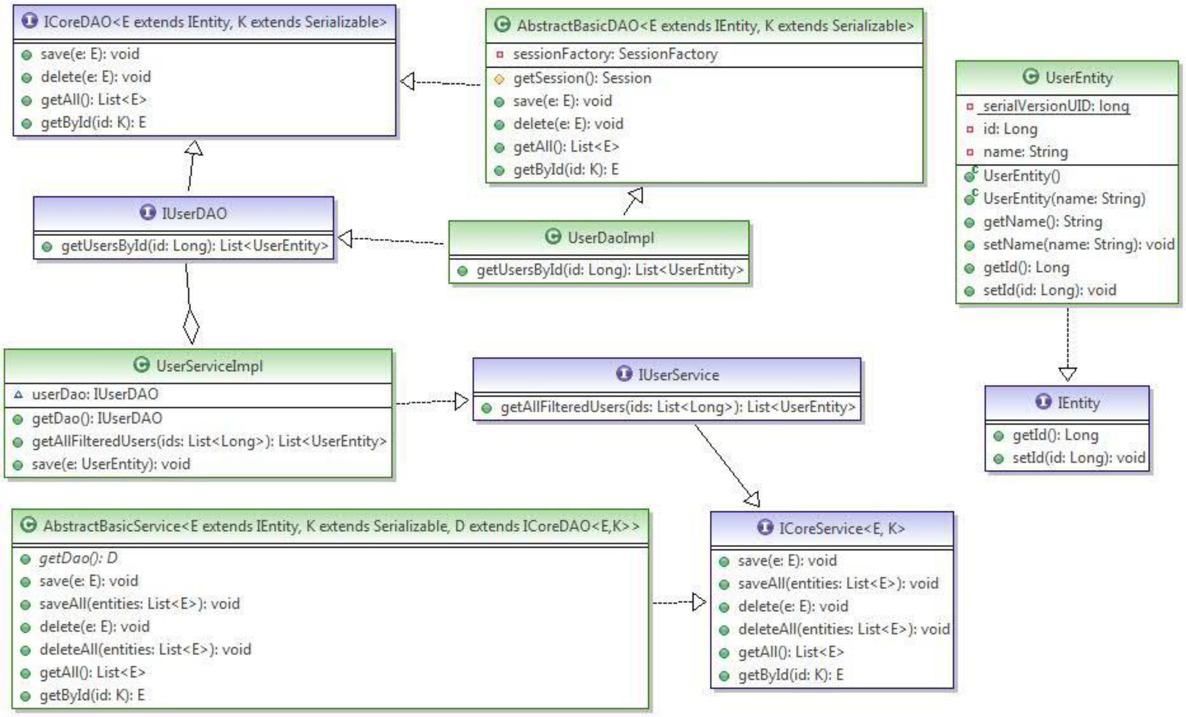
DAO – Service – Layer (Service – Repository –Entity) Layer Architecture

UML Architecture Diagram


ICoreDAO<E extends IEntity, K extends Serializable>
save(e: E): void
delete(e: E): void
getAll(): List<E>
getById(id: K): E
AbstractBasicDAO<E extends IEntity, K extends Serializable>
sessionFactory: SessionFactory
getSession(): Session
save(e: E): void
delete(e: E): void
getAll(): List<E>
getById(id: K): E
UserEntity
serialVersionUID: long
id: Long
name: String
UserEntity()
UserEntity(name: String)
getName(): String
setName(name: String): void
getId(): Long
setId(id: Long): void
IUserDAO
getUsersById(id: Long): List<UserEntity>
UserDaoImpl
getUsersById(id: Long): List<UserEntity>
UserServiceImpl
userDao: IUserDAO
getDao(): IUserDAO
getAllFilteredUsers(ids: List<Long>): List<UserEntity>
save(e: UserEntity): void
IUserService
getAllFilteredUsers(ids: List<Long>): List<UserEntity>
IEntity
getId(): Long
setId(id: Long): void
AbstractBasicService<E extends IEntity, K extends Serializable, D extends ICoreDAO<E,K>>
getDao(): D
save(e: E): void
saveAll(entities: List<E>): void
delete(e: E): void
deleteAll(entities: List<E>): void
getAll(): List<E>
getById(id: K): E
ICoreService<E, K>
save(e: E): void
saveAll(entities: List<E>): void
delete(e: E): void
deleteAll(entities: List<E>): void
getAll(): List<E>
getById(id: K): E

Maven Configuration(pom.xml)

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>dao-service-layer</groupId>
  <artifactId>dao-service-layer</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>many-to-many</name>
  <url>http://maven.apache.org</url>

  <properties>
             <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
             <spring.version>4.1.2.RELEASE</spring.version>
             <spring.security.version>3.2.3.RELEASE</spring.security.version>
       </properties>

  <dependencies>
             <dependency>
                    <groupId>junit</groupId>
                    <artifactId>junit</artifactId>
                    <version>4.0</version>
                    <scope>test</scope>
             </dependency>
             <dependency>
                    <groupId>org.springframework</groupId>
```

```
				<artifactId>spring-core</artifactId>
				<version>${spring.version}</version>
			</dependency>

			<dependency>
				<groupId>org.springframework</groupId>
				<artifactId>spring-web</artifactId>
				<version>${spring.version}</version>
			</dependency>

			<dependency>
				<groupId>org.springframework</groupId>
				<artifactId>spring-orm</artifactId>
				<version>${spring.version}</version>
			</dependency>

			<dependency>
				<groupId>org.springframework</groupId>
				<artifactId>spring-jdbc</artifactId>
				<version>${spring.version}</version>
			</dependency>

			<dependency>
				<groupId>mysql</groupId>
				<artifactId>mysql-connector-java</artifactId>
				<version>5.1.30</version>
			</dependency>

			<!-- C3P0 library -->
			<dependency>
				<groupId>com.mchange</groupId>
				<artifactId>c3p0</artifactId>
				<version>0.9.5</version>
			</dependency>

			<dependency>
				<groupId>org.hibernate</groupId>
				<artifactId>hibernate-core</artifactId>
				<version>4.3.7.Final</version>
			</dependency>

		</dependencies>
</project>
```

SQL Configuration

```
drop table IF EXISTS user;

CREATE TABLE IF NOT EXISTS user (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(255) DEFAULT NULL,
  PRIMARY KEY (id)
) ;

select * from user;
```

Spring-Hibernate Configuration (app-context.xml)

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:jee="http://www.springframework.org/schema/jee"
       xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:jdbc="http://www.springframework.org/schema/jdbc"
       xmlns:osgi="http://www.springframework.org/schema/osgi"
xmlns:security="http://www.springframework.org/schema/security"
       xsi:schemaLocation="http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
          http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
          http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd
          http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-
jee-3.0.xsd
          http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-
3.0.xsd
          http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-
jdbc-3.0.xsd
          http://www.springframework.org/schema/osgi http://www.springframework.org/schema/osgi/spring-
osgi.xsd
          http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-3.0.3.xsd">

       <tx:annotation-driven transaction-manager="transactionManager" />
       <context:annotation-config/>
       <context:component-scan base-package="com.ddlab.rnd"/>

    <bean id="dataSourceInternal" class="com.mchange.v2.c3p0.ComboPooledDataSource"
             destroy-method="close">
             <property name="driverClass" value="com.mysql.jdbc.Driver" />
             <property name="jdbcUrl" value="jdbc:mysql://localhost/test" />
             <property name="user" value="deba" />
             <property name="password" value="deba" />

             <!-- these are C3P0 properties -->
             <property name="acquireIncrement" value="5" />
             <property name="initialPoolSize" value="5" />
             <property name="minPoolSize" value="5" />
             <property name="maxPoolSize" value="20" />

       </bean>

       <!-- This is the lazy DataSource proxy that interacts with the target DataSource once a real
statement is sent to the database. Users use this DataSource to set up their Hibernate session factory,
which in turn forces the Hibernate second-level cache and also everything that interacts with that
Hibernate session factory to use it. -->
    <bean id="dataSource" class="org.springframework.jdbc.datasource.LazyConnectionDataSourceProxy">
        <property name="targetDataSource"><ref bean="dataSourceInternal" /></property>
    </bean>
    <bean id="sessionFactory" class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="hibernateProperties">
            <value>
                hibernate.id.new_generator_mappings=true,
                hibernate.show_sql=true
                <!-- hibernate.hbm2ddl.auto=update -->
                <!-- hibernate.current_session_context_class=thread -->
            </value>
        </property>
        <property name="annotatedClasses">
            <list>
                <value>com.ddlab.rnd.entity.UserEntity</value>
            </list>
        </property>
    </bean>

    <bean id="userService" class="com.ddlab.rnd.service.impl.UserServiceImpl"/>

    <bean id="transactionManager"
       class="org.springframework.orm.hibernate4.HibernateTransactionManager">
     <property name="sessionFactory" ref="sessionFactory" />
  </bean>

</beans>
```

Java Source Code

Entity Layer

IEntity.java

```
package com.ddlab.rnd.entity;
import java.io.Serializable;

public interface IEntity extends Serializable {

    public Long getId();

    public void setId(Long id);
}
```

UserEntity.java

```
package com.ddlab.rnd.entity;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "user")
public class UserEntity implements IEntity {

    private static final long serialVersionUID = 5921262165555614502L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;
```

```
        @Column(name = "name")
        private String name;

        public UserEntity() {

        }

        public UserEntity(String name) {
                this.name = name;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public Long getId() {
                return id;
        }

        public void setId(Long id) {
                this.id = id;
        }
}
```

DAO Layer

ICoreDAO.java

```
package com.ddlab.rnd.dao;
import java.io.Serializable;
import java.util.List;
import com.ddlab.rnd.entity.IEntity;

public interface ICoreDAO<E extends IEntity , K extends Serializable> {

        public void save(E e);

        public void delete(E e);

        public List<E> getAll();

    public E getById(K id);

}
```

IUserDAO.java

```
package com.ddlab.rnd.dao;
import java.util.List;
import com.ddlab.rnd.entity.UserEntity;

public interface IUserDAO extends ICoreDAO<UserEntity, Long> {

        public List<UserEntity> getUsersById(Long id);
}
```

AbstractBasicDAO.java

```
package com.ddlab.rnd.dao;
import java.io.Serializable;
import java.util.List;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import com.ddlab.rnd.entity.IEntity;

public abstract class AbstractBasicDAO<E extends IEntity, K extends Serializable> implements
ICoreDAO<E, K> {

    @Autowired
    @Qualifier("sessionFactory")
    private SessionFactory sessionFactory;

    protected Session getSession() {
        return sessionFactory.getCurrentSession();
    }

      public void save(E e) {
             System.out.println("Going to save--->"+e);
             getSession().save(e);
      }

      public void delete(E e) {

             System.out.println("Going to delete--->"+e);
      }

      public List<E> getAll() {
             String hql = "FROM UserEntity";
             return getSession().createQuery(hql).list();
      }

      public E getById(K id) {
             String hql = "FROM UserEntity u where u.id =:id";
             Query query = getSession().createQuery(hql);
             query.setParameter("id", id);
             return (E) query.list().get(0);
      }
}
```

UserDAOImpl.java

```
package com.ddlab.rnd.dao.impl;
import java.util.List;
import org.springframework.stereotype.Repository;
import com.ddlab.rnd.dao.AbstractBasicDAO;
import com.ddlab.rnd.dao.IUserDAO;
import com.ddlab.rnd.entity.UserEntity;

@Repository(value = "userDao")
public class UserDaoImpl extends AbstractBasicDAO<UserEntity,Long> implements IUserDAO {

      public List<UserEntity> getUsersById(Long id) {
             System.out.println("UserDaoImpl : getUsersById");
             return null;
      }
}
```

Service Layer

ICoreService.java

```
package com.ddlab.rnd.service;
import java.util.List;

public interface ICoreService<E,K> {

	public void save(E e);

	public void saveAll(List<E> entities);

	public void delete(E e);

	public void deleteAll(List<E> entities);

	public List<E> getAll();

	public E getById(K id);
}
```

IUserService.java

```
package com.ddlab.rnd.service;
import java.util.List;
import com.ddlab.rnd.entity.UserEntity;

public interface IUserService extends ICoreService<UserEntity, Long> {

	public List<UserEntity> getAllFilteredUsers(List<Long> ids);
}
```

AbstractBasicService.java

```
package com.ddlab.rnd.service;
import java.io.Serializable;
import java.util.List;
import org.springframework.transaction.annotation.Transactional;
import com.ddlab.rnd.dao.ICoreDAO;
import com.ddlab.rnd.entity.IEntity;
@Transactional
public abstract class AbstractBasicService<E extends IEntity, K extends Serializable, D extends
ICoreDAO<E, K>> implements ICoreService<E, K> {

	public abstract D getDao();

	public void save(E e) {
		getDao().save(e);
	}

	public void saveAll(List<E> entities) {

	}

	public void delete(E e) {
		getDao().delete(e);
	}

	public void deleteAll(List<E> entities) {

	}
```

```
      public List<E> getAll() {
             return getDao().getAll();
      }

      public E getById(K id) {
             return getDao().getById(id);
      }
}
```

UserService.java

```
package com.ddlab.rnd.service.impl;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;
import com.ddlab.rnd.dao.IUserDAO;
import com.ddlab.rnd.entity.UserEntity;
import com.ddlab.rnd.service.AbstractBasicService;
import com.ddlab.rnd.service.IUserService;

@Service(value = "userService")
public class UserServiceImpl extends AbstractBasicService<UserEntity, Long, IUserDAO> implements
IUserService {

    @Autowired
    @Qualifier("userDao")
    IUserDAO userDao;

    @Override
    public IUserDAO getDao() {
        return userDao;
    }

      public List<UserEntity> getAllFilteredUsers(List<Long> ids) {
             return null;
      }

      @Override
      public void save(UserEntity e) {
             super.save(e);
      }
}
```

Test.java (Test Program)

```java
package com.ddlab.rnd.test;
import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.ddlab.rnd.entity.UserEntity;
import com.ddlab.rnd.service.IUserService;

public class Test {

        public static void main(String[] args) {
                ApplicationContext context = new ClassPathXmlApplicationContext("app-context.xml");
                IUserService userService = (IUserService) context.getBean("userService");
                UserEntity userEntity = new UserEntity("Deba1");
                userService.save(userEntity);

                List<UserEntity> usersList = (List<UserEntity>) userService.getAll();
                for(UserEntity user : usersList) {
                        System.out.println(user.getId()+"---->"+user.getName());
                }

                UserEntity user1 = userService.getById(1L);
                System.out.println("User Name :::"+user1.getName());
        }
}
```