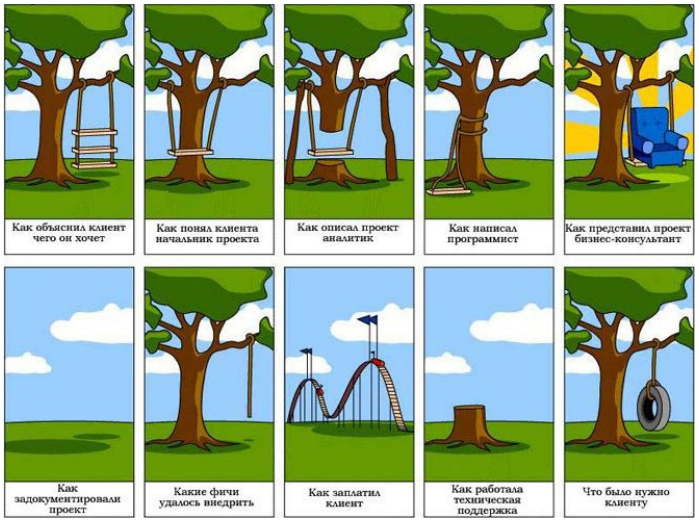
2014

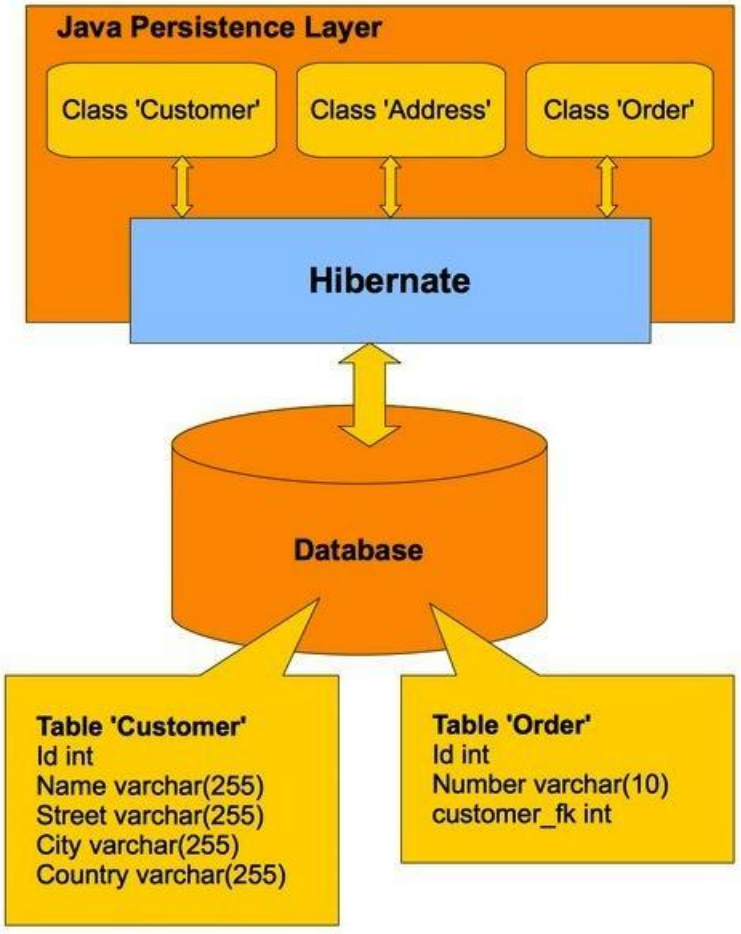
Hibernate.
Углубленный курс. Специализация

Иван Спресов
Юлий Слабко

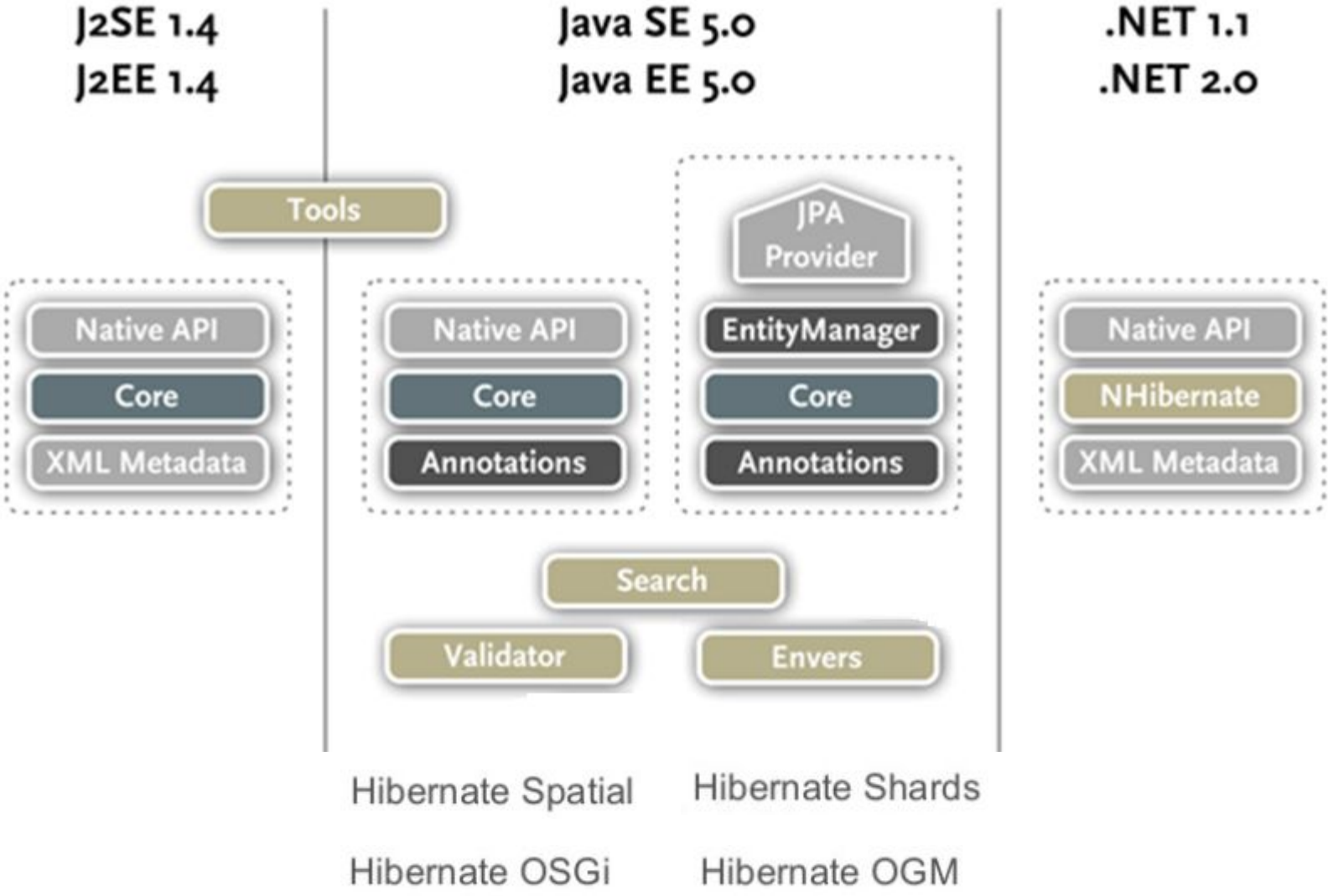


Несколько слов о Hibernate

Несколько слов о Hibernate

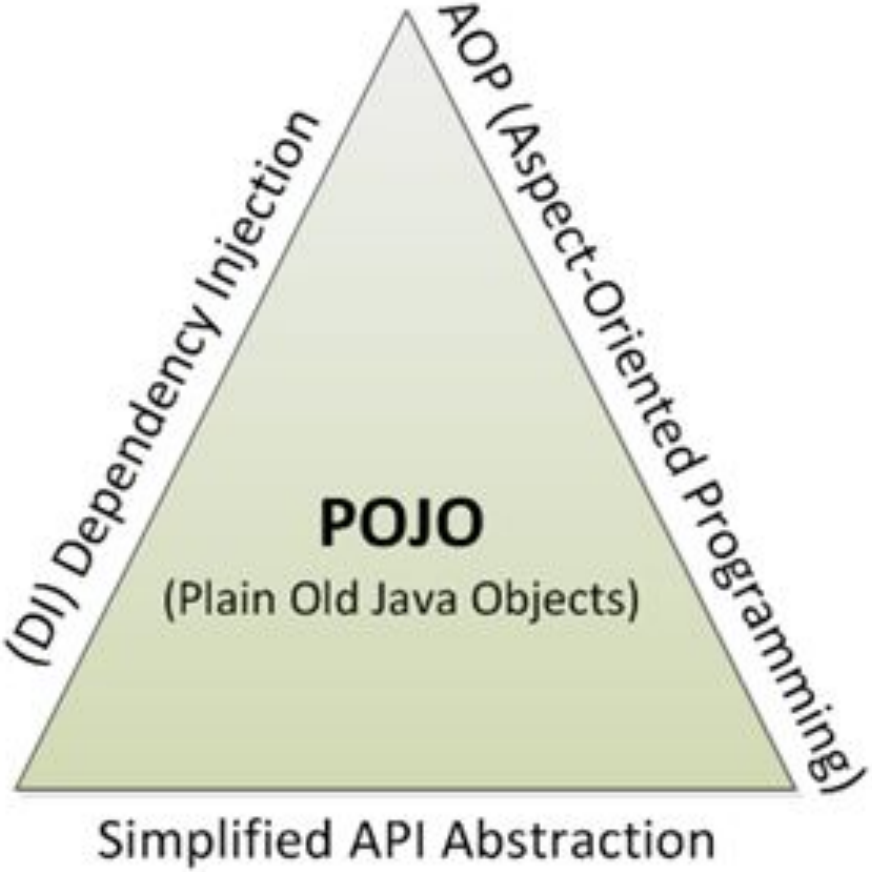


Несколько слов о Hibernate

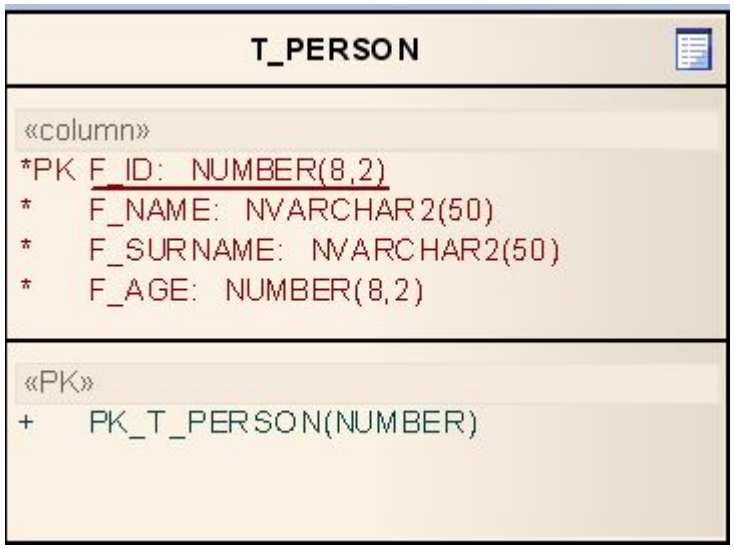


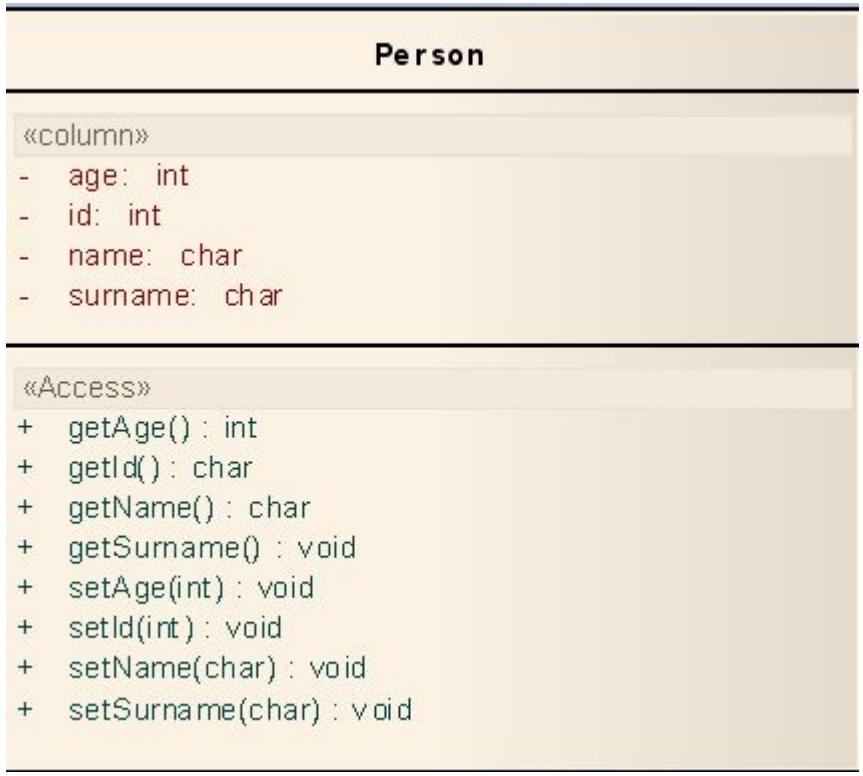
ВАШИ ВОПРОСЫ?

POJO - Plain Old Java Object



POJO - Plain Old Java Object





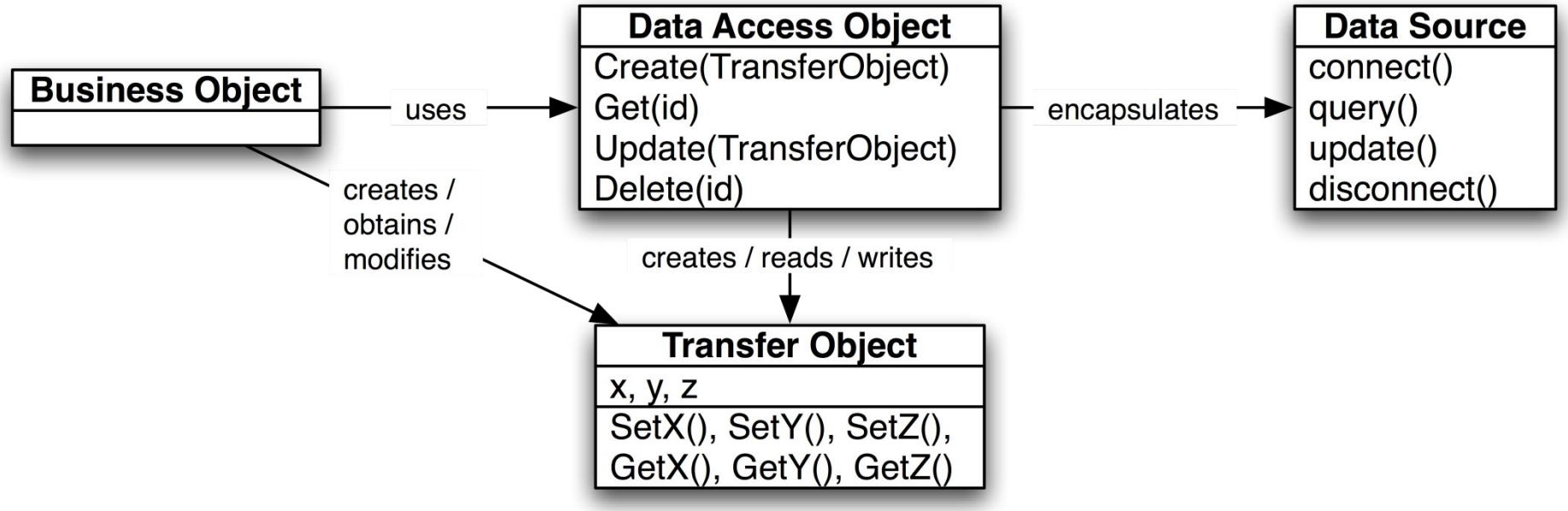
POJO - Plain Old Java Object

```
@Entity
@Table
public class Person implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    @Column
    private Integer age;
    @Column
    private String name;
    @Column
    private String surname;

    public Person() {
    }
```

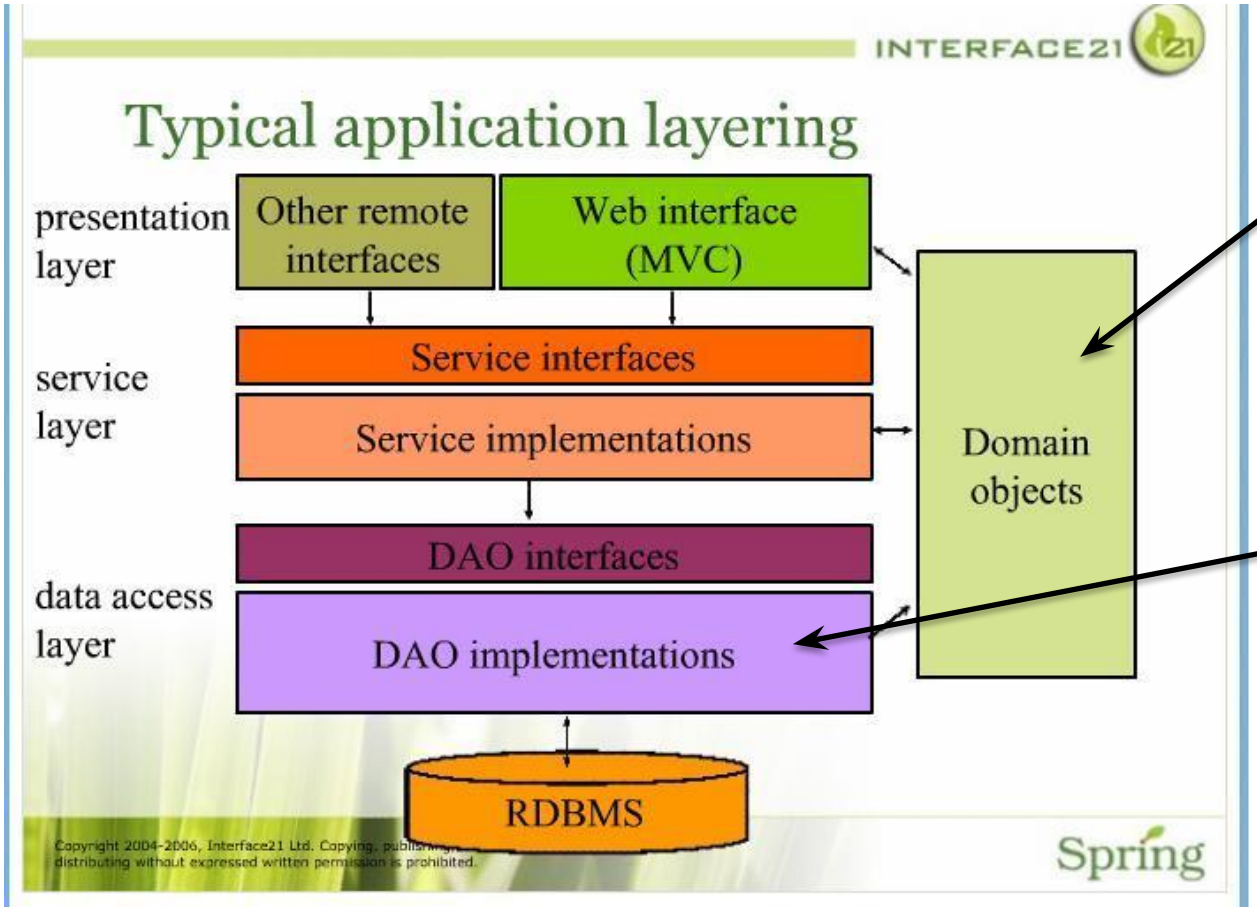
POJO - Plain Old Java Object

```java
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Person)) return false;

    Person person = (Person) o;

    if (!age.equals(person.age)) return false;
    if (!id.equals(person.id)) return false;
    if (!name.equals(person.name)) return false;
    if (!surname.equals(person.surname)) return false;

    return true;
}

@Override
public int hashCode() {
    int result = id.hashCode();
    result = 31 * result + name.hashCode();
    result = 31 * result + surname.hashCode();
    result = 31 * result + age.hashCode();
    return result;
}
```

DAO Architect

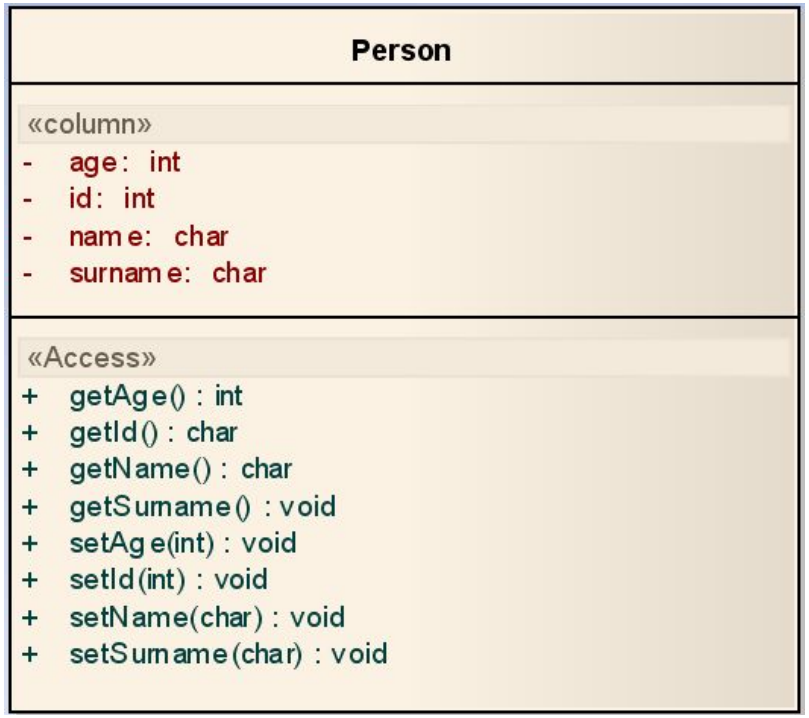


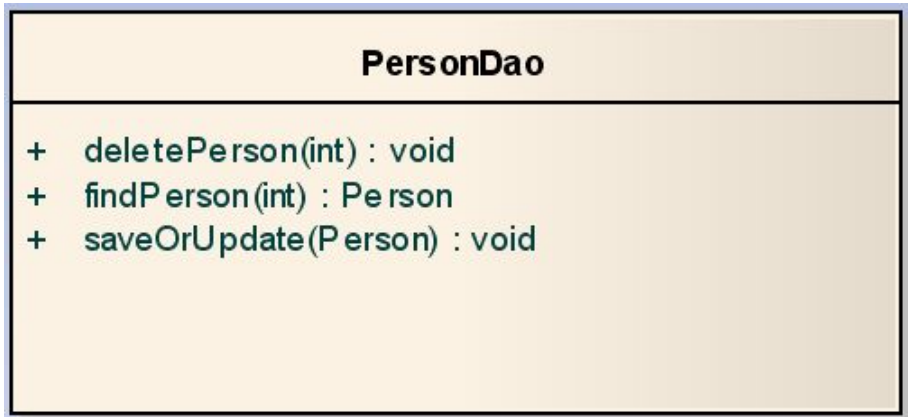
DAO Architect

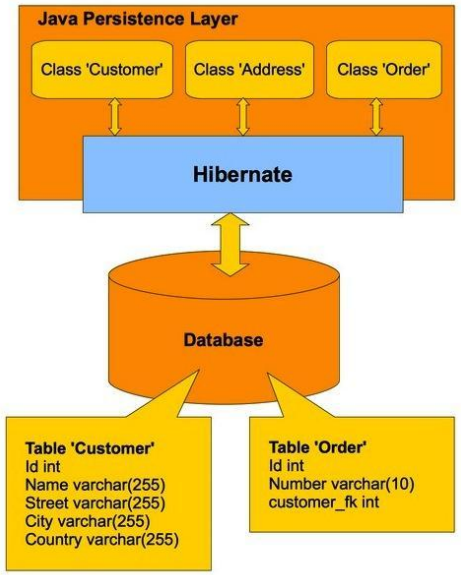


ВАШИ ВОПРОСЫ?

СОХРАНЕНИЕ ДАННЫХ В СУБД С ПОМОЩЬЮ HIBERNATE

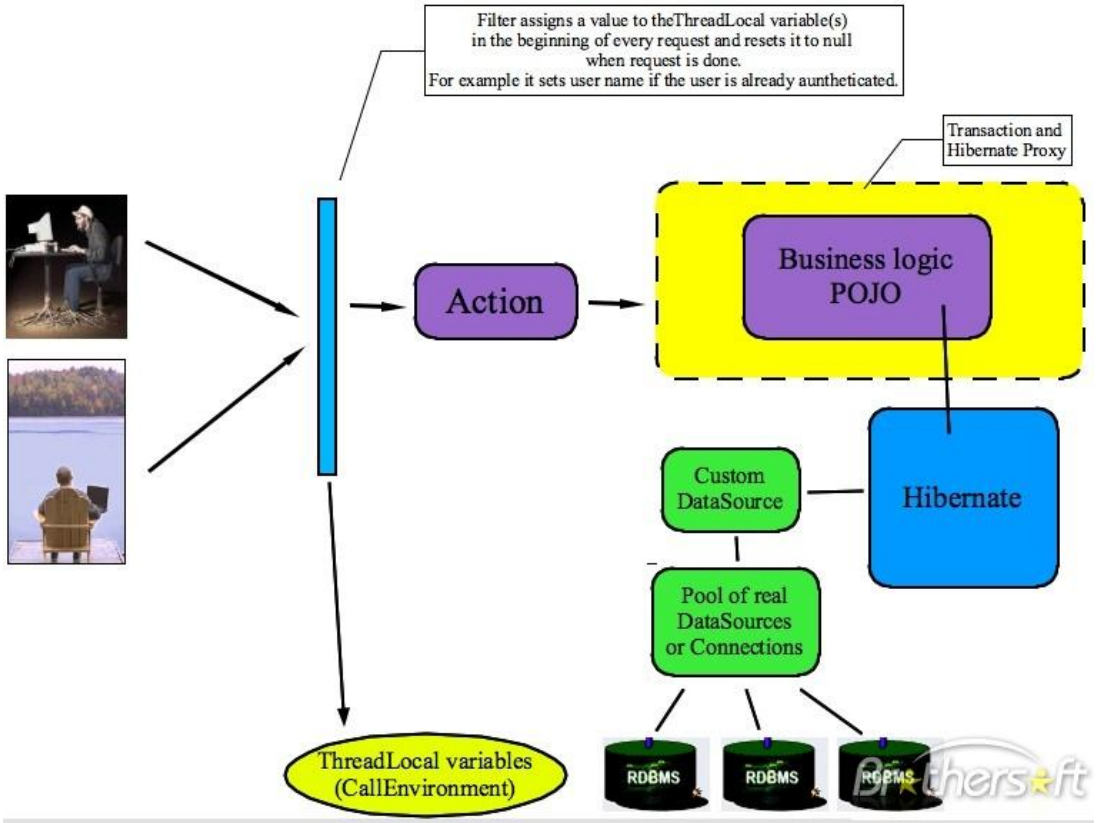






Структура проекта

- hibernate-annotation-first **[hibernate-annotation]**
  - .idea
  - src
    - main
      - java
        - by.academy.it
          - db
            - exceptions
              - BaseDao
              - Dao
              - PersonDao
            - loader
              - MenuLoader
              - PersonLoader
            - pojos
              - Person
            - util
              - HibernateUtil
      - resources
        - hibernate.cfg.xml
        - log4j.properties
  - test
  - hibernate-annotation.iml
  - managePerson.iml
  - pom.xml



POM-файл

```
<groupId>by.it</groupId>
<artifactId>hibernate-annotation</artifactId>
<version>1.0</version>
<packaging>jar</packaging>
<name>managePerson</name>
<build...>
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <hibernate.version>4.3.11.Final</hibernate.version>
</properties>
<repositories...>
<dependencies>
    <dependency...>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>1.5.6</version>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>${hibernate.version}</version>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-entitymanager</artifactId>
        <version>${hibernate.version}</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.34</version>
    </dependency>
</dependencies>
```

Файл настройки Hibernate (hibernate.cfg.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/personDB</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">pwd</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.connection.pool_size">10</property>
        <property name="show_sql">true</property>
        <property name="hibernate.hbm2ddl.auto">create</property>

        <mapping class="by.academy.it.pojos.Person"/>
    </session-factory>
</hibernate-configuration>
```

Log4j.properties

```
log4j.rootLogger=INFO, CONSOLE
#log4j.rootLogger=INFO, FILE
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.Target=System.err
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.conversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p - %m%n
log4j.appender.FILE=org.apache.log4j.RollingFileAppender
log4j.appender.FILE.File=log4j.log
log4j.appender.FILE.MaxFileSize=512KB
log4j.appender.FILE.MaxBackupIndex=3
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.conversionPattern=%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p - %m%n
```

Конфигурация фабрики сессий

```
public class HibernateUtil {
    private static HibernateUtil util = null;
    private static Logger log = Logger.getLogger(HibernateUtil.class);
    private SessionFactory sessionFactory = null;
    private final ThreadLocal sessions = new ThreadLocal();

    private HibernateUtil() {
        try {
            Configuration configuration = new Configuration().configure();
            /*.configure(HibernateUtil.class.getResource("/hibernate.cfg.xml"));*/
            StandardServiceRegistryBuilder serviceRegistryBuilder = new StandardServiceRegistryBuilder();
            serviceRegistryBuilder.applySettings(configuration.getProperties());
            ServiceRegistry serviceRegistry = serviceRegistryBuilder.build();
            sessionFactory = configuration.buildSessionFactory(serviceRegistry);
        } catch (Throwable ex) {
            log.error("Initial SessionFactory creation failed." + ex);
            System.exit(0);
        }
    }
```

Конфигурация фабрики сессий

```
    public Session getSession () {
        Session session = (Session) sessions.get();
        if (session == null) {
            session = sessionFactory.openSession();
            sessions.set(session);
        }
        return session;
    }

    public static synchronized HibernateUtil getHibernateUtil(){
        if (util == null){
            util = new HibernateUtil();
        }
        return util;
    }
}
```

Контракт DOA. CRUD API

```
interface Dao<T> {
    void saveOrUpdate(T t) throws DaoException;

    T get(Serializable id) throws DaoException;

    T load(Serializable id) throws DaoException;

    void delete(T t) throws DaoException;
}
```

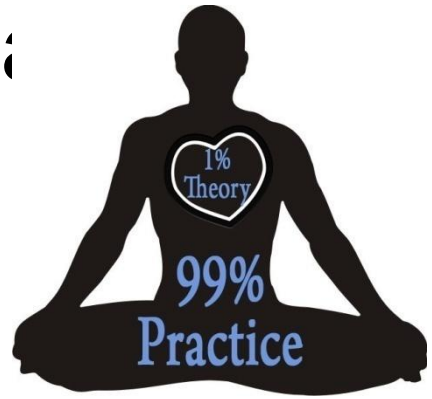
Операция сохранения сущности

```
public class BaseDao<T> implements Dao<T> {
    private static Logger log = Logger.getLogger(BaseDao.class);
    private Transaction transaction = null;
    public BaseDao() { }

    public void saveOrUpdate(T t) throws DaoException{
        try {
            Session session = util.getSession();
            transaction = session.beginTransaction();
            session.saveOrUpdate(t);
            log.info("saveOrUpdate(t):" + t);
            transaction.commit();
            log.info("Save or update (commit):" + t);
        } catch (HibernateException e) {
            log.error("Error save or update PERSON in Dao" + e);
            transaction.rollback();
            throw new DaoException(e);
        }
    }
```

Запуск и стартовое меню.

```
public class PersonLoader {
    public static HibernateUtil util = null;
    public static void main(String[] args) throws Exception {

        Locale.setDefault(Locale.US);
        util = HibernateUtil.getHibernateUtil();
        Person person = new Person(null, 35, "Yuli", "Slabko");
        Session session = util.getSession();
        session.saveOrUpdate(person);
        System.out.println("Start Menu");
        menu();
    }
}
```

ВАШИ ВОПРОСЫ?

Практика

Создайте стандартный проект maven. Настройте зависимости. Создайте mapping-файлы, конфигурацию и соберите проект. Проверьте, что все библиотеки есть в на