УДК 004.62

РАЗРАБОТКА УРОВНЯ РАБОТЫ С БАЗОЙ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА MYBATIS В ВЕБ-ПРИЛОЖЕНИИ «ОНЛАЙН СЕРВИС ДЛЯ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЯ НА JAVA»

П.А. СТАНКЕВИЧ
(Представлено: Ю.Н. КРАВЧЕНКО)

Расматривается разработка уровня работы с базой данных с использованием фреймворка MyBatis в веб-приложении, написанном на языке программирования Java. Данное веб-приложение реализованно с использованием множества фреймворков, одними из которых стали Spring Framework, Spring Security и MyBatis.

На стадии проектирования объемных веб-приложений часто возникает вопрос, как необходимо работать с уровнем базы данных. Многие останавливаются на стандартных средствах работы с данными, вследствие чего замедляют скорость работы веб-приложения, делают исходные файлы веб-приложения грамосткими и сложно читаемыми с множеством повторений одинаковых фрагментов кода, а также нарушают безопасность работы с данными.

Для решения данных проблем существуют специализированные фреймворки, которые с более высокой скоростью обрабатывают запросы пользователей, и имеют большую степень защищенности от SQL-инъекцый.

В данной работе пойдет речь о фреймворке MyBatis, который был использован в веб-приложении **«Онлайн сервис для обучения программирования на Java».**

При проектировании «Онлайн сервиса для обучения программирования на Java» была взята за основу архитектура, представленная на рисунке 1. Для ее реализации необходимо было выбрать способ работы с базой данных. Выбранный способ должен был обладать такими характеристиками, как:

- работа под Java SE 8;
- взаимодействие с фреймворками Spring Framework и Apache Commons DBCP;
- высокую скорость работы.

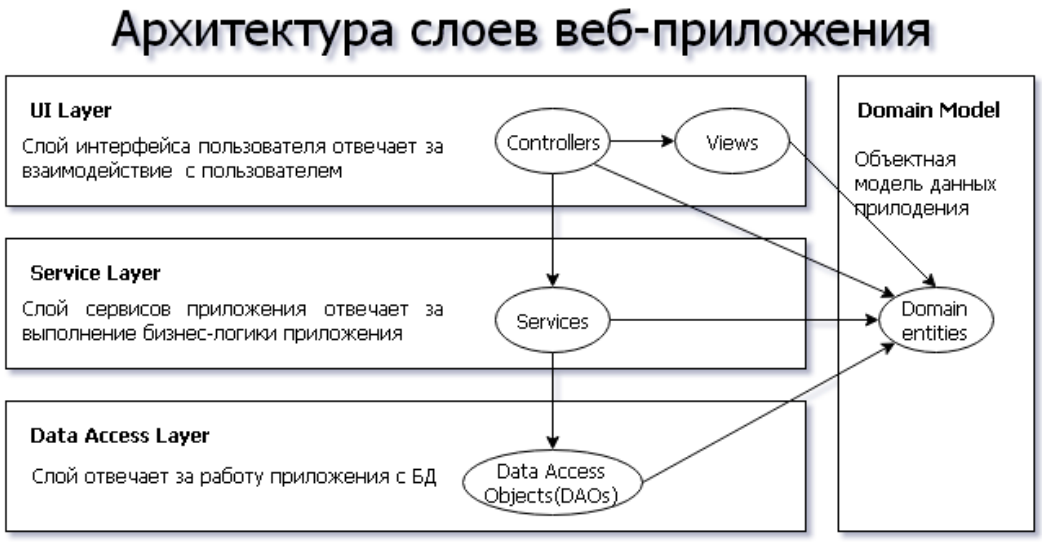

Рисунок 1. – Архитектура разработанного веб-приложения

После рассмотрения возможных вариантов был выбран MyBatis как основное средство на уровне доступа к данным. *MyBatis* – это фреймворк для связывания данных, который упрощает работу с реляционными базами данных в объектно-ориентированных приложениях [1].

Из истории MyBatis можно подчеркнуть, что изначально он имел название iBATIS, который разрабатывал Клинтон Бегин в 2001 году. Тогда основное внимание уделялось разработке криптографических программных решений. Первым программным продуктом, выпущенным iBATIS, был Secrets, он был предназначен для шифрования личных данных и использовался как инструмент подписи документов. Вскоре после выпуска Secrets проект iBATIS стал фокусироваться на Интернете и других интернет-

технологиях. Благодаря этому были разработаны несколько интересных программных продуктов, в том числе веб-фреймворк Axle, который был альтернативой JSP.

В начале 2002 года Microsoft опубликовала документ, в котором утверждается, что .Net был в 10 раз быстрее и в 4 раза более производительным, чем J2EE. Понимая, что это не так, Клинтон быстро отреагировал, и 1 июля 2002 года вышел аналог программного продукта Pet Store, который получил название JPetStore. Основываясь на тех же требованиях к магазину Pet Store, JPetStore продемонстрировал, что Java может быть не только более продуктивной, чем .NET, но также может достигнуть лучшей архитектуры, чем была использована в реализации Microsoft.

JPetStore использовал интересный уровень сохранения, который быстро привлек внимание сообщества с открытым исходным кодом. Вследствие чего были пораждены iBATIS Database Layer и Data Access Object (DAO). Вскоре после выпуска JPetStore вопросы и запросы к базам данных SQL Maps и DAO породили проект, который станет известен как уровень базы данных iBATIS. Уровень данных iBATIS включает в себя две структуры, упакованные вместе: SQL Maps и DAO.

В 2004 году проект присоединился к Apache Software Foundation и дальше разрабатывался в Apache на протяжении 6 лет.

За шесть лет многое изменилось в мире программного обеспечения с открытым исходным кодом. Все от практики развития инфраструктуры, лицензирования до технологий баз. В 2010 году основная команда разработчиков решила, что эти разработки стоит изучить и что iBATIS выиграет от некоторых значительных изменений. Это было серьезное решение, и этот шаг означал, что необходимо оставить имя iBATIS в прошлом [2].

Основным назначением фреймворка считается возможность отвязать Java классы от различных СУБД. Т.е. идея такова – описываем преобразование или маппинг таблиц базы данных или запросов в атрибуты Java классов посредством XML файла и, делая вызовы соответствующих методов MyBatis-a, просто заполняем данными эти Java классы.

Рассмотрим реализацию уровня работы с данными (Data Access Layer) на примере сущности Группа. Реализация сущности (Java-класс) представлена в листинге 1.

Листинг 1 – КлассGroup

```
1 package by.psu.learnjava.entity;
2
3 public class Group extends Entity {
4
5    private static final long serialVersionUID = 1L;
6
7    private User owner;
8    private String accessKey;
9    private String name;
10     private List<User> members;
11
12     public User getOwner() {
13        return owner;
14     }
15
16     public void setOwner(User owner) {
17  this.owner = owner;
18     }
19  ...
20  }
```

Из листинга 1 можно сделать вывод, что класс наследуются от некоторого класса Entity (Сущность). Класс Group имеет поля характерные для своей реализации, а также имеет методы получения и изменений полей, гетеры и сетеры (geters, seters). В листингах вместо троеточий пропущенны гетеры и сетеры для отстальных полей.

Далее необходимо создать слой Data Access Object (DAO). DAO – это шаблон проектирования, который предостовляетопределенные возможности независимо от того, какой механизм хранения используется и без необходимости специальным образом соответствовать этому механизму хранения. В качестве примера расмотрим интерфейс GroupDao, приведенный в листинге 2.

Листинг 2 – Интерфейс GroupDao

```
21 package by.psu.learnjava.dao;
22
23 public interface GroupDao extends CrudDao<Long, Group> {
24
25    List<Group>findAll();
26
27    List<Group>findForUserId(Long id);
28
29    List<User>findMembersByGroupId(Long id);
30
31    Group findByToken(String token);
32
33    void deleteMember(Map<String, Object> map);
34
35    void updateGroupList(Map<String, Object> map);
36 }
```

Из листинга 2 можно заметить, что интерфейс расширяется с помощью CrudDao, и имеет дополнительно 6 опираций, поиска всех групп, поиска групп по индетификатору пользователя, поиск пользователей по индетификатору группы, поиск группы по токен-ключу, удаление пользователя из группы, а также обновление списка пользователей в группе.

Каждый уважающий себя программист знает, нет смысла интерфейса без его реализации. Нужно создать класс, который наследуется от SqlSessionDaoSupport и реализует интерфейс GroupDao. SqlSessionDaoSupport – это класс MyBatis'а в нем есть объект DataSource, в котором хранятся данные подключений. Для реализации DataSource в данном примере используется библиотека ApacheCommonsDBCP, а именно класс BasicDataSource. Реализация класса представлена в листинге 3.

Листинг 3 – КлассGroupMapper

```
1 package by.psu.learnjava.dao.mybatis;
2
3 public class GroupMapper extends SqlSessionDaoSupport implements GroupDao {
4
5    @Override
6    public void create(Group obj) {
7 getSqlSession().insert("Group.create", obj);
8    }
9
10    @Override
11    public Group read(Long id) {
12       return getSqlSession().selectOne("Group.read", id);
13    }
14
15    @Override
16    public void update(Group obj) {
17 getSqlSession().update("Group.update", obj);
18    }
19
20    @Override
21    public void delete(Long id) {
22 getSqlSession().delete("Group.delete", id);
23    }
24
25    @Override
26    public List<Group>findAll() {
27       return getSqlSession().selectList("Group.findAll");
28    }
29
```

```
30    @Override
31    public List<Group>findForUserId(Long id) {
32       return getSqlSession().selectList("Group.findForUserId", id);
33    }
34
35    @Override
36    public List<User>findMembersByGroupId(Long id) {
37       return getSqlSession().selectList("Group.findMembersByGroupId", id);
38    }
39
40    @Override
41    public Group findByToken(String token) {
42       return getSqlSession().selectOne("Group.findByToken", token);
43    }
44
45    @Override
46    public void deleteMember(Map<String, Object> map) {
47 getSqlSession().delete("Group.deleteMember", map);
48    }
49
50    @Override
51    public void updateGroupList(Map<String, Object> map) {
52 getSqlSession().update("Group.updateGroupList", map);
53    }
54 }
```

Из приведенного примера в листинге 3 можно заметить, что в каждом методе вызывается getSqlSession, после чего могут быть вызваны различные методы, благодоря которым будут изменены или полученны данные. Но в параметрах у всех методов указанна ссылка, которая указывает, в каком xmlмаппере лежит нужный запрос к БД.

Для работы с MyBatis необходимо создать файл конфигурации, его часть приведена в листинге 4. Он описывается с помощью xml файла, в котором содержится основной тег configuration, для его написания необхимо описать тег typeAliases и mappers. Тег typeAliases должен содержать в себе теги typeAlias, который содержит ссылку на тип, и альтернативную ссылку на этот тип. Тег mappers должен содержать пути к файлам с описаниям действий над сущностью.

Листинг 4 – Конфигурация MyBatis

```
1  <?xml version = "1.0" ?>
2  <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
3  "http://mybatis.org/dtd/mybatis-3-config.dtd">
4
5  <configuration>
6  <typeAliases>
7        ...
8  <typeAliastype="by.psu.learnjava.entity.Group" alias="group" />
9        ...
10 </typeAliases>
11 <mappers>
12       ...
13 <mapper resource="by/psu/learnjava/dao/mybatis/Group.xml" />
14       ...
15 </mappers>
16 </configuration>
```

После настройки файла конфинурации необходимо реализовать файл с описанием возможных действий над сущностью. Данный файл описывается в формате xml, в котором корневым тегом служит

mapper с указанием имени маппера. Внутри доступны resultMap, sql, select, delete, insert, update. Тег resultMap описывает сущность, sql позволяет хранить фрогменты sqlкода, select нужен для считывания данных, а delete, insert, update необходимы для удаления, добавления и обновления записей соответственно. Пример файла приведен в листинге 5.

Листинг 5 – Фрагмент файла Group.xml

```
<?xml version = "1.0" ?>
<!DOCTYPE mapper
   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="Group">

<resultMap type="group" id="group">
<id property="id" column="group_id" />
<result property="accessKey" column="group_access_key" />
<result property="name" column="group_name" />
<collection property="owner" javaType="User" resultMap="User.user"/>
</resultMap>

        <sql id="sql">
                SELECT
         g.id AS group_id,
g.access_key AS group_access_key,
         g.name AS group_name,
         u.id AS user_id,
u.email AS user_email,
u.passwd AS user_passwd,
u.login AS user_login,
u.unconfirmed_email AS user_unconfirmed_email,
         r.id AS role_id,
         r.name AS role_name,
         ls.id AS learn_speed_id,
ls.speed AS learn_speed_speed,
         ls.name AS learn_speed_name,
         p.id AS person_id,
         p.name AS person_name,
p.surname AS person_surname,
p.patronymic AS person_patronymic,
p.birthday AS person_birthday

      FROM
learn_java.groups AS g
         INNER JOIN learn_java.users AS u ON g.id_user=u.id
         INNER JOIN learn_java.roles AS r ON u.id_role=r.id
         INNER JOIN learn_java.learn_speed AS ls ON u.id_learn_speed=ls.id
         INNER JOIN learn_java.persons AS p ON u.id_person=p.id
        </sql>

<insert id="create" parameterType="group">
<selectKeykeyProperty="id" order="BEFORE" resultType="long">
         SELECT nextval('learn_java.groups_id_seq'::regclass)
</selectKey>
      INSERT INTO learn_java.groups(
         id, id_user, name)
      VALUES (#{id}, #{user.id}, #{name})
</insert>
```

```
51
52 <select id="read" parameterType="long" resultMap="group">
53                     <include refid="sql" />
54                     g.id = #{id}
55 </select>
56
57 <select id="findAll" resultMap="group">
58 <include refid="sql" />
59       ORDER BY g.id
60 </select>
61
62 <select id="findForUserId" parameterType="long" resultMap="group">
63 <include refid="sql" />
64       WHERE u.id = #{id}
65       ORDER BY g.id
66 </select>
67 ...
68 </mapper>
```

Заключение

Разработка веб-приложений с использованием паттерна DAO, а также фреймворка MyBatis актуальна, так как данный способ работы с данными давно доказал свою эффективность, а также не зависимость от выбранной СУБД. Но данный тип реализации уровня работы с данными лучше использовать с объемными проектами, которые, скорее всего, нужно будет поддерживать на протяжении некоторого времени.

ЛИТЕРАТУРА

1. ApacheAttic [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Apache_Attic. – Дата доступа: 20.09.2017.
2. TheMyBatisBlog [Электронный ресурс]. – Режим доступа: http://blog.mybatis.org/p/about.html. – Дата доступа: 20.09.2017.