УДК 004.657

MYBATIS КАК ИНСТРУМЕНТ ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ В JAVA-ПРИЛОЖЕНИЯХ

Д.Ю. ДЁКИН
(Представлено: канд. физ.-мат. наук, доц. Ю.Ф. ПАСТУХОВ)

Рассматривается MyBatis фреймворк для отображения данных из базы на доменные объекты в Java-приложениях. Представлен краткий обзор базовых возможностей MyBatis фреймворка и его базовой конфигурации. Приведены примеры маппингов и запросов к базе данных, вызова запросов из Java-кода.

При проектировании и разработке приложений практически всегда приходится иметь дело с базами данных. Поэтому встает вопрос о выборе того или иного ORM или persistence фреймворка, который будет служить для отображения данных из базы в пользовательские объекты. Существует достаточно большое количество различных ORM, таких как Hibernate, JPA, EclipseLink и др. Каждый из них имеет свои достоинства и недостатки. Большинство фреймворков поддерживают JPA спецификацию.

Рассматриваемый в данной статье фреймворк является альтернативой JPA спецификации.

MyBatis является persistence фреймворком, который отображает SQL запросы на доменные объекты. Данный фреймворк (так же, как и ORM фреймворки) ликвидирует необходимость писать шаблонный и ненужный код для уже давно решенных задач. MyBatis позволяет отображать данные из базы в примитивные Java-типы и POJO (Plain Old Java Object, простой Java-объект). MyBatis может быть сконфигурирован как при помощи XML, так и при помощи аннотаций. Для удобства и простоты последующие примеры будут приводиться на XML.

MyBatis хорошо подходит для тех случаев, когда в приложении имеется достаточно много различных сложных выборок из нескольких таблиц (много JOIN-ов). Все дело в том, что MyBatis позволяет отображать SQL-запросы на сущности. Поэтому производительность запроса полностью определяется разработчиком. Это позволяет полностью контролировать, какие запросы и как именно будут отправляться в БД.

Для базовой конфигурации необходимо указать источник данных (от англ. data source) с именем драйвера и url к БД, логином и паролем. Также в конфигурационном файле указываются type alias (доменные объекты), mapper (xml-файлы, которые хранят SQL-запросы). Пример приведен в листинге 1.

Листинг 1 – Пример конфигурационного файла

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
<properties resource="config.properties">
<property name="schema" value="db_schema"/>
</properties>

<dataSource type="POOLED">
<property name="driver" value="${driver}"/>
<property name="url" value="${url}"/>
<property name="username" value="${username}"/>
<property name="password" value="${password}"/>
</dataSource>
<typeAliases>
  <typeAlias type="by.psu.domain.User"/>
  <typeAlias type="by.psu.domain.Role"/>
</typeAliases>

                        <mappers>
        <mapper resource="by/psu/repository/IUserMapper.xml"/>
        <mapper resource="by/psu/repository/IRoleMapper.xml"/>
       </mappers>
 </configuration>
```

Для того чтобы показать, как происходит отображение результатов выборок на доменные объекты, введем следующие простые типы: User, Role (методы доступа к полям опущены для экономии места). Данные классы представленны в листинге 2.

Листинг 2 – Описание классов User и Role

```
class User {
  private String id;
  private String name;
  private List<Role> roles;
}
```

```
class Role {
  private String id;
  private String name;
}
```

Теперь для данной простой иерархии классов введем так называемые маппинги (отоброжаения), представленные в листинге 3.

Листинг 3 – Пример маппинга на доменные классы

```
<resultMap type="User" id="userMap">
  <result id="id" column="u_id"/>
  <result property="name" column="u_name"/>
  <collection property="roles" ofType="Role" resultMap="roleMap"/>
</resultMap>

<resultMap type="Role" id="roleMap">
  <result id="id" column="r_id"/>
  <result property="name" column="r_name"/>
</resultMap>
```

Как можно видеть из листинга, маппинги довольно просты.

Таким образом, мы задаем правило того, как должны будут заполняться наши объекты из выборки данных. MyBatis будет брать значение колонки и вставлять его в объект при помощи соответствующего setter-а (метода установки значения).

В данном примере используется коллекция ролей. Однако если есть необходимость реализовать ассоциацию (один-к-одному связь), то в таком случае в маппере будет использоваться **association** тег.

Идентификаторы мепперов должны быть уникальными (аттрибут *id* в теге *resultMap*).

Для наглядности на рисунке 1 представлена схема БД для данных сущностей.

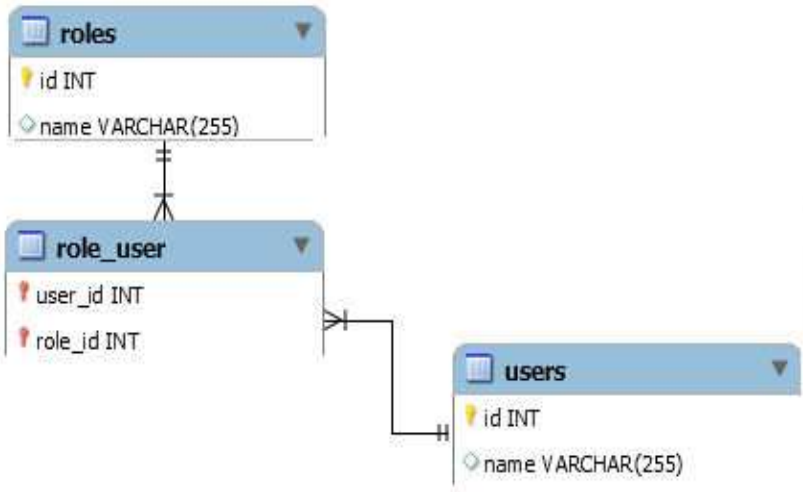

Рисунок 1. – Схема базы данных

Можно привести пример обычного запроса, который должен вернуть пользователя со всеми его ролями по его id (листинг 4).

Листинг 4 – Пример запроса выборки данных

```
<select id="getUserById" resultMap="userMap">
  SELECT
        u.id AS u_id,
        u.name AS u_name,
        r.id AS r_id,
        r.name AS r_name
  FROM users AS u
  INNER JOIN user_role AS ur
        ON u.id = ur.user_id
  INNER JOIN roles AS r
        ON ur.role_id = r.id
  WHERE u.id = #{id}
</select>
```

Псевдонимы для выбираемых колонок должны полностью совпадать со значениями аттрибутов *colomn* в мапперах. В данном примере используется элемент в фигурных скобках “#{id}” – это тот параметр, который мы передаем в данный запрос. Как это вызывается из Java-кода можно увидеть в листинге 5.

Листинг 5 – Пример вызвоа запроса из Java-кода

```
class User {
  private SqlSessionTemplate template;
  public User getUserById(String id) {
        return template.selectOne("IUserMapper.getUserById", id);
  }
}
```

SqlSessionTemplate является тем классом, через который происходит обращение к БД. В примере выше мы выбираем один элемент по id, который принимаем в качестве параметра метода.

Метод *selectOne* первым параметром принимает имя маппера и id запроса, который требуется выполнить, а вторым параметром уже сам параметр запроса.

Параметры, передаваемые в запросы, могут быть не только примитивными типами, но и сложными пользовательскими объектами. В таком случае для обращения к полям этих параметров следует обращаться через точку. Например, для параметра типа User, переданного в запрос, можем обратиться к его имени так: #{user.name}.

Для выполнения вставки, удаления, обновления данных используются теги insert, delete, update, соответственно. Кроме того, имеется возможность переиспользовать части SQL запросов, как представлено в листинге 6.

Листинг 6 – Пример переиспользования фрагмента SQL-запроса

```
<sql id="filtered">
  FROM users AS u
  INNER JOIN user_role AS ur
        ON u.id = ur.user_id
  INNER JOIN roles AS r
        ON ur.role_id = r.id
  WHERE u.id = #{id}
</sql>
-----
<select id="getUserById" resultMap="userMap">
  SELECT
        u.id AS u_id,
        u.name AS u_name,
        r.id AS r_id,
        r.name AS r_name
  <include refid="filtered"/>
</select>
```

Когда один и тот же фрагмент SQL-запроса используется в нескольких местах, уместно будет вынести этот фрагмент, как показано выше, и ссылаться на него в остальных местах. Это экономит место, а также способствует уменьшению объема работы, если данный фрагмент запроса в какой-то момент необходимо изменить. При таком подходе изменения будут вноситься только в одном месте, что экономит еще и время на внесение изменений.

Заключение

Представлен краткий обзор базовых возможностей MyBatis фреймворка и его базовой конфигурации. Также представлены примеры маппингов и запросов к БД.

Приведен пример вызова запросов из Java-кода.

Как видно, MyBatis не упраздняет наличие SQL-кода в проекте, однако он делает его написание и сопровождение гораздо легче.

Данная работа может служить отправной точкой в изучении фреймворка MyBatis.

ЛИТЕРАТУРА

1. Хабрахабр [Электронный ресурс] MyBatis как более быстрая альтернатива Hibernate. – Режим доступа: https://habrahabr.ru/post/247885. – Дата доступа: 27.09.2017.
2. Tuturialspoint [Электронный ресурс] MYBATIS Tutorial. – Режим доступа: http://www.tutorialspoint.com/mybatis/. – Дата доступа: 27.09.2017.
3. MyBatis [Электронный ресурс] Introduction. – Режим доступа: http://www.mybatis.org/mybatis-3/. – Дата доступа: 27.09.2017.