一、通用Mapper

MP中的基本CRUD在内置的BaseMapper中都已得到了实现。

创建MapperTests测试类：

```
package com.atguigu.mybatisplus;

@SpringBootTest
public class MapperTests {

    @Resource
    private UserMapper userMapper;
}
```

1、Create

```
@Test
public void testInsert(){

    User user = new User();
    user.setName("Helen");
    user.setAge(18);
    //不设置email属性，则生成的动态sql中不包括email字段

    int result = userMapper.insert(user);
    System.out.println("影响的行数：" + result); //影响的行数
    System.out.println("id：" + user.getId()); //id自动回填
}
```

2、Retrieve

```
@Test
public void testSelect(){
```

```
    //按id查询
    User user = userMapper.selectById(1);
    System.out.println(user);

    //按id列表查询
    List<User> users = userMapper.selectBatchIds(Arrays.asList(1, 2, 3));
    users.forEach(System.out::println);

    //按条件查询
    Map<String, Object> map = new HashMap<>();
    map.put("name", "Helen"); //注意此处是表中的列名，不是类中的属性名
    map.put("age", 18);
    List<User> users1 = userMapper.selectByMap(map);
    users1.forEach(System.out::println);
}
```

3、Update

```
@Test
public void testUpdate(){

    User user = new User();
    user.setId(1L);
    user.setAge(28);

    //注意：update时生成的sql自动是动态sql
    int result = userMapper.updateById(user);
    System.out.println("影响的行数：" + result);
}
```

4、Delete

```
@Test
public void testDelete(){

    int result = userMapper.deleteById(5);
    System.out.println("影响的行数：" + result);
```

```
}
```

二、通用Service

MP中有一个接口 IService和其实现类 ServiceImpl，封装了常见的业务层逻辑

1、创建Service接口

创建 service 包，创建 UserService，继承 IService

```
package com.atguigu.mybatisplus.service;

public interface UserService extends IService<User> {

}
```

2、创建Service实现类

创建 impl 包，创建 UserServiceImpl，继承 ServiceImpl，实现 UserService

```
package com.atguigu.mybatisplus.service.impl;

@Service
public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements

}
```

3、创建测试类

创建ServiceTests

```
package com.atguigu.mybatisplus;

```

```
@SpringBootTest
public class ServiceTests {

    @Resource
    private UserService userService;

}
```

4、测试记录数

```
@Test
public void testCount(){

    int count = userService.count();
    System.out.println("总记录数：" + count);
}
```

5、测试批量插入

```
@Test
public void testSaveBatch(){

    // SQL长度有限制，海量数据插入单条SQL无法实行，
    // 因此MP将批量插入放在了通用Service中实现，而不是通用Mapper
    ArrayList<User> users = new ArrayList<>();
    for (int i = 0; i < 5; i++) {
        User user = new User();
        user.setName("Helen" + i);
        user.setAge(10 + i);
        users.add(user);
    }
    userService.saveBatch(users);
}
```

三、自定义Mapper

当通用Mapper无法满足我们的需求时，我们可以自定义基于Mapper接口的xml文件，并在xml文件中配置SQL语句

1、接口方法定义

在UserMapper接口中定义如下方法

```
List<User> selectAllByName(String name);
```

2、创建xml文件

在resources目录中创建mapper目录，创建UserMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.

<mapper namespace="com.atguigu.mybatisplus.mapper.UserMapper">

   <sql id="Base_Column_List">
       id, name, age, email
   </sql>

    <select id="selectAllByName" resultType="com.atguigu.mybatisplus.entity.U
        select
        <include refid="Base_Column_List"/>
        from user
        where
        name = #{name}
    </select>
</mapper>
```

注意：MP中mapper目录是持久层映射文件的默认目录，如果是其他目录，需要配置mapper-locations，例如：

```
mybatis-plus.mapper-locations=classpath:xml/*.xml
```

3、测试条件查询

在MapperTests中创建如下测试用例

```
@Test
public void testSelectAllByName(){
    List<User> users = userMapper.selectAllByName("Helen");
    users.forEach(System.out::println);
}
```

四、自定义Service

1、添加接口方法

UserService中添加接口方法

```
List<User> listAllByName(String name);
```

2、实现接口方法

```
@Override
public List<User> listAllByName(String name) {
    // baseMapper对象指向当前业务的mapper对象
    return baseMapper.selectAllByName("Helen");
}
```

4、测试

ServiceTests中添加测试方法

```
@Test
public void testListAllByName(){
    List<User> users = userService.listAllByName("Helen");
    users.forEach(System.out::println);
}
```