基于注解的一对一查询

一、@One注解

1、select 相当于resultMap里的select

2、fetchType

true　　表示支持延迟加载

false　即可取消延迟加载

二、步骤

在基于xml的基础上，删掉Mapper.xml文件（IAccountDao.xml）

1、仍需要配置映射

```
public class User implements Serializable{

    private Integer userId;
    private String userName;
    private String userAddress;
    private String userSex;
    private Date userBirthday;

    //一对多关系映射：一个用户对应多个账户
    private List<Account> accounts;

    public List<Account> getAccounts() {
        return accounts;
    }

    public void setAccounts(List<Account> accounts) {
        this.accounts = accounts;
    }

    public Integer getUserId() {
        return userId;
    }

    public void setUserId(Integer userId) {
        this.userId = userId;
```

```
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getUserAddress() {
        return userAddress;
    }

    public void setUserAddress(String userAddress) {
        this.userAddress = userAddress;
    }

    public String getUserSex() {
        return userSex;
    }

    public void setUserSex(String userSex) {
        this.userSex = userSex;
    }

    public Date getUserBirthday() {
        return userBirthday;
    }

    public void setUserBirthday(Date userBirthday) {
        this.userBirthday = userBirthday;
    }

    @Override
    public String toString() {
        return "User{" +
                "userId=" + userId +
                ", userName='" + userName + '\'' +
                ", userAddress='" + userAddress + '\'' +
                ", userSex='" + userSex + '\'' +
                ", userBirthday=" + userBirthday +
                '}';
    }
}
```

2、编写IAccountDao 接口

```
public interface IAccountDao {

    /**
     * 查询所有账户，并且获取每个账户所属的用户信息
     * @return
     */
    @Select("select * from account")
    @Results(id="accountMap",value = {
            @Result(id=true,column = "id",property = "id"),
            @Result(column = "uid",property = "uid"),
            @Result(column = "money",property = "money"),
            //实现一对一的映射
            @Result(property = "user",column =
"uid",one=@One(select="com.itheima.dao.IUserDao.findById",fetchType=
FetchType.EAGER))
    })

    List<Account> findAll();

    /**
     * 根据用户id查询账户信息
     * @param userId
     * @return
     */
    @Select("select * from account where uid = #{userId}")
    List<Account> findAccountByUid(Integer userId);
}
```

、测试

```
public class AccountTest {
    private InputStream in;
    private SqlSessionFactory factory;
    private SqlSession session;
    private IAccountDao accountDao;

    @Before
    public  void init()throws Exception{
        in = Resources.getResourceAsStream("SqlMapConfig.xml");
        factory = new SqlSessionFactoryBuilder().build(in);
        session = factory.openSession();
        accountDao = session.getMapper(IAccountDao.class);
    }
```

```
    @After
    public  void destroy()throws  Exception{
        session.commit();
        session.close();
        in.close();
    }

    @Test
    public  void  testFindAll(){
        List<Account> accounts = accountDao.findAll();
        for(Account account : accounts){
            System.out.println("----每个账户的信息-----");
            System.out.println(account);
            System.out.println(account.getUser());
        }
    }
}
```