

【济南中心】JavaEE就业班同步笔记第三阶段：Hibernate-part01

小鲁哥哥 • 2017-6-4 14:52:47

【济南中心】JavaEE就业班同步笔记第三阶段： Hibernate-part01

1.1 Hibernate的学习路线

Hibernate的入门和流程

Hibernate的一级缓存，对象的状态

Hibernate的关联关系映射

Hibernate的查询方式和检索策略

1.2 CRM的管理的CRUD的操作

1.2.1 Hibernate的概述

1.2.1.1 什么是Hibernate

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，它将POJO与数据库表建立映射关系，是一个全自动的orm框架，hibernate可以自动生成SQL语句，自动执行，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。

Hibernate是一个持久层的ORM框架。

1.2.1.2 什么是ORM

ORM：Object Relational Mapping。对象关系映射。开发语言用的是Java，面向对象的（Object）。使用的数据库是关系型数据库（Relational）。就是将对象与数据库中的表建立一种映射关系，操作对象就可以操作这个表。


1.2.2 Hibernate的入门

1.2.2.1 步骤一：下载Hibernate开发包

下载地址：

<https://sourceforge.net/projects/hibernate/files/hibernate-orm/5.0.7.Final/>

1.2.2.2 步骤二：Hibernate的目录结构

名称	修改日期	类型	大小
 documentation	2016/1/13 12:45	文件夹	
 lib	2016/1/13 12:45	文件夹	
 project	2016/1/13 12:45	文件夹	
 changelog.txt	2016/1/13 12:33	文本文档	46 KB
 hibernate_logo.gif	2013/4/17 11:37	GIF 文件	2 KB
 lgpl.txt	2013/4/17 11:37	文本文档	26 KB

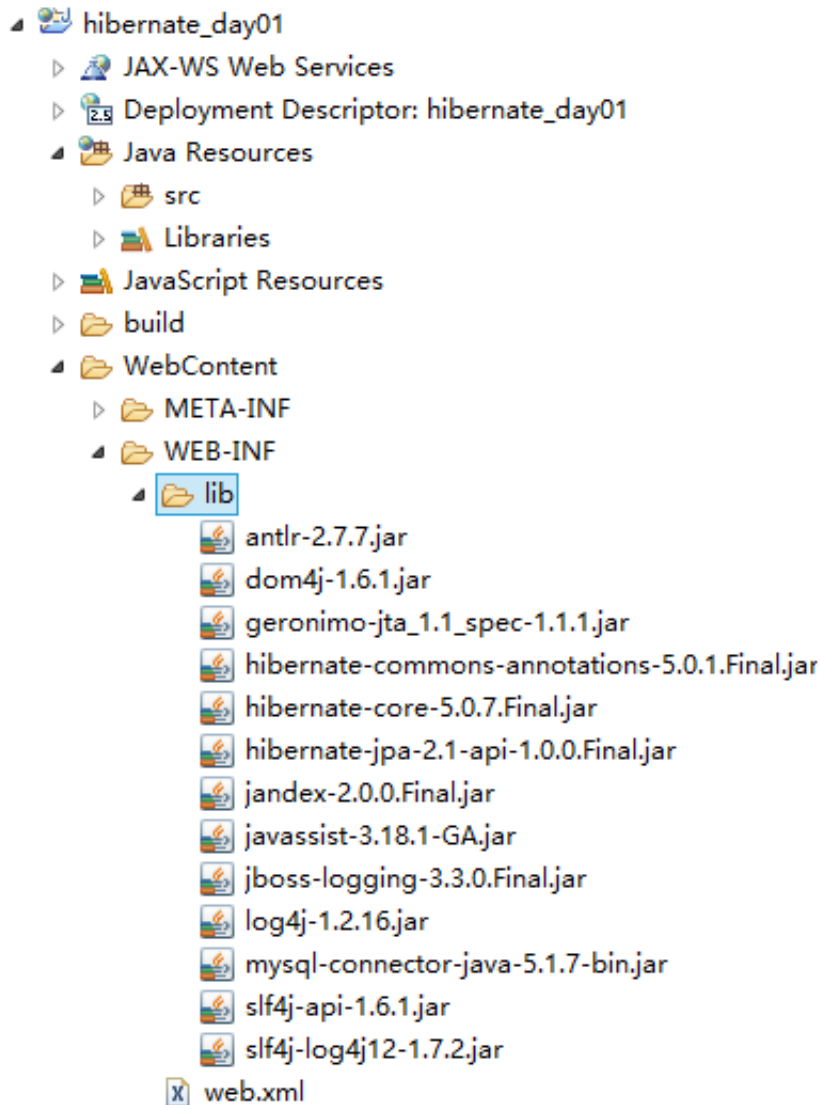
1.2.2.3 步骤三：创建WEB工厂，引入jar包

创建web工程。

D:\hibernate-release-5.0.7.Final\lib\required*.jar

数据库驱动包

日志记录的包



1.2.2.4 步骤四：创建数据库和表

```
[mw_shl_code=sql,true]CREATE TABLE `cst_customer` (  
  `cust_id` bigint(32) NOT NULL AUTO_INCREMENT COMMENT '客户编号(主键)',  
  `cust_name` varchar(32) NOT NULL COMMENT '客户名称(公司名称)',  
  `cust_source` varchar(32) DEFAULT NULL COMMENT '客户信息来源',  
  `cust_industry` varchar(32) DEFAULT NULL COMMENT '客户所属行业',  
  `cust_level` varchar(32) DEFAULT NULL COMMENT '客户级别',  
  `cust_phone` varchar(64) DEFAULT NULL COMMENT '固定电话',  
  `cust_mobile` varchar(16) DEFAULT NULL COMMENT '移动电话',  
  PRIMARY KEY (`cust_id`))
```

) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;[/mw_shl_code]

1.2.2.5 步骤五：创建实体类

```
[mw_shl_code=java,true]public class Customer {  
    private Long cust_id;  
    private String cust_name;  
    private String cust_source;  
    private String cust_industry;  
    private String cust_level;  
    private String cust_phone;  
    private String cust_mobile;  
    public Long getCust_id() {  
        return cust_id;  
    }  
    public void setCust_id(Long cust_id) {  
        this.cust_id = cust_id;  
    }  
    public String getCust_name() {  
        return cust_name;  
    }  
    public void setCust_name(String cust_name) {  
        this.cust_name = cust_name;  
    }  
    public String getCust_source() {  
        return cust_source;  
    }  
    public void setCust_source(String cust_source) {  
        this.cust_source = cust_source;  
    }  
    public String getCust_industry() {  
        return cust_industry;  
    }  
    public void setCust_industry(String cust_industry) {
```

```

        this.cust_industry = cust_industry;
    }
    public String getCust_level() {
        return cust_level;
    }
    public void setCust_level(String cust_level) {
        this.cust_level = cust_level;
    }
    public String getCust_phone() {
        return cust_phone;
    }
    public void setCust_phone(String cust_phone) {
        this.cust_phone = cust_phone;
    }
    public String getCust_mobile() {
        return cust_mobile;
    }
    public void setCust_mobile(String cust_mobile) {
        this.cust_mobile = cust_mobile;
    }
}

```

}/{mw_shl_code]

1.2.2.6 步骤六：创建映射

映射文件通常有一个命名规则：类名.hbm.xml

引入约束：

```

[mw_shl_code=xml,true]<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

```

```

<?xml version="1.0" encoding="UTF-8"?>

```

```

<!DOCTYPE hibernate-mapping PUBLIC

```

```

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <!-- ORM:Object Relational Mapping,将实体类O和数据库的表R 建立映射关系 -->
  <class name="com.itheima.hibernate.domain.Customer" table="cst_customer">
    <!-- 类中的属性与表中的主键对应 -->
    <id name="cust_id" column="cust_id">
      <generator class="native"/>
    </id>
    <!-- 类中的属性与表中的字段对应 -->
    <property name="cust_name" column="cust_name"/>
    <property name="cust_source" column="cust_source"/>
    <property name="cust_industry" column="cust_industry"/>
    <property name="cust_level" column="cust_level"/>
    <property name="cust_phone" column="cust_phone"/>
    <property name="cust_mobile" column="cust_mobile"/>
  </class>
</hibernate-mapping>[/mw_shl_code]

```

1.2.2.7 步骤七：创建Hibernate的核心配置文件

```

[mw_shl_code=xml,true]<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <!-- 连接数据库的信息 -->
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  >
    <property name="hibernate.connection.url">jdbc:mysql:///hibernate_day01</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">123</property>
  </session-factory>
</hibernate-configuration>

```

```

<!-- 数据库的方言:根据底层的数据库生成不同的SQL -->
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<!-- 配置显示SQL -->
<property name="hibernate.show_sql">true</property>
<!-- 配置格式化SQL -->
<property name="hibernate.format_sql">true</property>
<!-- 配置hbm2ddl -->
<property name="hibernate.hbm2ddl.auto">update</property>

<!-- 加载映射文件 -->
<mapping resource="com/itheima/hibernate/domain/Customer.hbm.xml"/>
</session-factory>
</hibernate-configuration>[/mw_shl_code]

```

1.2.2.8 步骤八：编写测试方法

```

[/mw_shl_code=java,true]public class HibernateDemo1 {

    @Test
    /**
     * 保存操作
     */
    public void demo1(){
        // 加载Hibernate的核心配置文件.
        Configuration configuration = new Configuration().configure();
        // 创建一个SessionFactory的对象.
        SessionFactory sessionFactory = configuration.buildSessionFactory();
        // 创建Session(相当于JDBC中的Connection)
        Session session = sessionFactory.openSession();
        // 开启事务:
        Transaction transaction = session.beginTransaction();
        // 完成操作:
        Customer customer = new Customer();
    }
}

```

```
customer.setCust_name("柳岩");
```

```
session.save(customer);
```

```
// 提交事务
```

```
transaction.commit();
```

```
// 释放资源
```

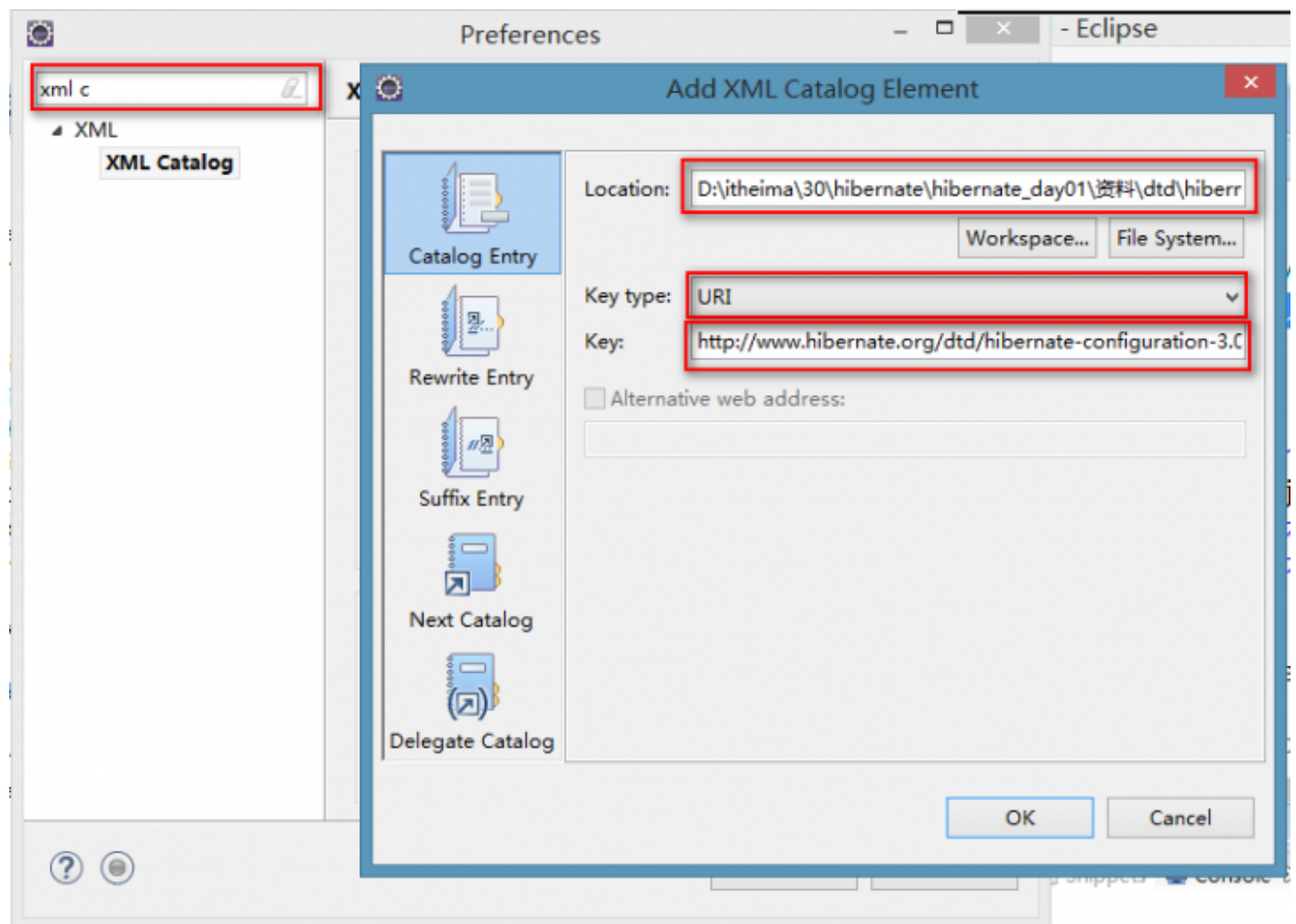
```
session.close();
```

```
}
```

```
}[mw_shl_code]
```

1.2.3 Hibernate的常用的配置

1.2.3.1 Hibernate的配置文件的提示



1.2.3.2 Hibernate的映射文件的配置

class的配置

```
<!--
    class标签：用来描述类与表的映射关系。
    * name      :类的全路径的。(Object)
    * table     :数据库中的表名称。(Relational)
        * 如果类名和表名称一致的话，那么table可以省略。
    * catalog   :数据库名称
-->
<class name="com.itheima.hibernate.domain.Customer" table="cst_customer">
```

Id和property的配置

```
<!--
    id标签：用来描述类中的OID与表中的主键建立映射。
    * name      :类中的属性名称。
    * column    :表中的字段名称。
        * 如果类中的属性名称和表中的字段名称一致，column可以省略。
    * length    :字段的长度。(Hibernate自动建表的时候)
    * type      :数据类型。
        * Java的数据类型
        * Hibernate的数据类型
        * SQL的数据类型
-->
<id name="cust_id" column="cust_id">
    <!-- 主键生成策略： -->
    <generator class="native"/>
</id>
```

<!--

property标签：用来描述类中的属性与表中的字段建立映射

- * name :类中的属性名称。
- * column :表中的字段名称。
 - * 如果类中的属性名称和表中的字段名称一致，column可以省略。
- * length :字段的长度。（Hibernate自动建表的时候）
- * type :数据类型。
 - * Java的数据类型
 - * Hibernate的数据类型
 - * SQL的数据类型

-->

1.2.3.3 Hibernate的核心配置文件的配置

Hibernate的核心配置文件有两种方式：

【属性文件的方式】

需要在src下创建一个hibernate.properties文件

配置：key=value

[mw_shl_code=xml,true]hibernate.connection.driver_class=com.mysql.jdbc.Drvier[/mw_shl_code]

属性文件的方式：结构不是很清晰而且这种方式不能加载映射文件。需要手动编写代码才能加载映射文件。

【XML配置文件的方式】

数据库的基本参数：

```
<!-- 连接数据库的信息 -->
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql:///hibernate_day01</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">123</property>
```

Hibernate的属性

```
<!-- 数据库的方言:根据底层的数据库生成不同的SQL -->
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<!-- 配置显示SQL -->
<property name="hibernate.show_sql">true</property>
<!-- 配置格式化SQL -->
<property name="hibernate.format_sql">true</property>
<!-- 配置hbm2ddl -->
<property name="hibernate.hbm2ddl.auto">update</property>
```

hibernate.hbm2ddl.auto有几个取值

- * none :不使用映射转成DDL。
- * create :如果原来有表将原有的表删除。每次都会新创建一个表。测试的时候使用。
- * create-drop :如果原来有表将原有的表删除。每次都会新创建一个表。执行完操作以后会将表删除掉。测试的时候使用。
- * update :如果数据库中有表使用原来的表。如果没有表会创建一个表。而且可以更新原有表结构。
- * validate :不会创建表。校验映射和表结构是否正确

1.2.4 Hibernate的常用的API

1.2.4.1 Configuration:配置对象

Configuration是一个配置对象，作用主要有两个，一个用来加载核心配置文件。另一个用来加载映射文件。

【加载核心配置文件】

加载hibernate.properties

```
[mw_shl_code=java,true]Configuration cfg = new Configuration();[/mw_shl_code]
```

加载hibernate.cfg.xml

```
[mw_shl_code=java,true]Configuration cfg = new Configuration().configure();[/mw_shl_code];
```

【加载映射文件】

```
[mw_shl_code=java,true]// 手动加载映射文件
```

```
// configuration.addResource("com/itheima/hibernate/domain/Customer.hbm.xml");
```

```
// configuration.addClass(Customer.class);[/mw_shl_code]
```

1.2.4.2 SessionFactory:Session工厂

SessionFactory负责管理Session，管理连接池，管理Hibernate二级缓存。采用了工厂模式，不是一个轻量级的对象。通常情况下一个项目只对应一个SessionFactory就够了。是线程安全的对象。

【抽取Hibernate的工具类】

```
[mw_shl_code=java,true]public class HibernateUtils {  
  
    private static final Configuration configuration;  
    private static final SessionFactory sessionFactory;  
  
    static{  
        configuration = new Configuration().configure();  
        sessionFactory = configuration.buildSessionFactory();  
    }  
  
    public static Session openSession(){  
        return sessionFactory.openSession();  
    }  
}[/mw_shl_code]
```

【配置C3P0连接池】

引入c3p0连接池的jar包：

配置C3P0连接池：

<!-- 配置C3P0连接池 -->

```
<property name="connection.provider_class">org.hibernate.connection.C3P0Connecti  
onProvider</property>
```

<!--在连接池中可用的数据库连接的最少数目 -->

```
<property name="c3p0.min_size">5</property>
```

<!--在连接池中所有数据库连接的最大数目 -->

```
<property name="c3p0.max_size">20</property>
```

<!--设定数据库连接的过期时间,以秒为单位,

如果连接池中的某个数据库连接处于空闲状态的时间超过了timeout时间,就会从连接池中清除 -->

```
<property name="c3p0.timeout">120</property>
```

<!--每3000秒检查所有连接池中的空闲连接 以秒为单位-->

```
<property name="c3p0.idle_test_period">3000</property>
```

1.2.4.3 Session:相当于Connection

Session是Hibernate程序与数据库之间的桥梁。完成CRUD的操作。Session是一个单线程的对象，内部维护了Hibernate的一级缓存。

【Session保存某个对象】

```
[mw_shl_code=java,true]@Test
/**
 * 保存客户
 */
public void demo1(){
    Session session = HibernateUtils.openSession();
    Transaction tx = session.beginTransaction();

    Customer customer = new Customer();
    customer.setCust_name("梁");

    session.save(customer);// 保存对象

    tx.commit();
    session.close();
}[/mw_shl_code]
```

【Session查询某个对象】

```
[mw_shl_code=java,true] @Test
/**
 * 查询某个对象
 * 面试：get方法和load方法的区别？
 * * get方法采用的是立即加载，执行到该行代码的时候，马上发送SQL语句进行查询。
查询之后返回的是真实对象本身。
 * * 查询一个找不到的对象返回null.
 * * load方法采用的是延迟加载(lazy)，执行到改行的代码的时候，不会马上发送SQL语句
，
 * 只有真正使用这个对象的时候（使用这个对象的普通属性的时候）才会发送SQL语句
。
 * load方法返回的是代理对象。（产生的是Customer的子类对象）
```

```

*    查询一个找不到的对象抛出异常： ObjectNotFoundException
*/
public void demo2(){
    Session session = HibernateUtils.openSession();
    Transaction tx = session.beginTransaction();
    // 调用session.get()方法查询某个对象。
//    Customer customer = session.get(Customer.class, 100l);// 马上发送SQL语句.

    // 调用session.load()方法查询某个对象。
    Customer customer = session.load(Customer.class, 300l);// 不会马上发送SQL语句.

    System.out.println(customer);

    tx.commit();
    session.close();
}[/mw_shl_code]
【Session中修改某个对象的方法】
[/mw_shl_code=java,true]@Test
/**
 * 修改某个对象
 */
public void demo3(){
    Session session = HibernateUtils.openSession();
    Transaction tx = session.beginTransaction();

    // 1.直接创建对象进行修改.(没有设置的属性会被修改为null)
    /*Customer customer = new Customer();
    customer.setCust_id(4l);
    customer.setCust_name("梁如花");

    session.update(customer);*/

```

// 2.先查询再修改. (推荐)

```
Customer customer = session.get(Customer.class, 4l);
customer.setCust_name("梁如花");
```

```
session.update(customer);
```

```
tx.commit();
session.close();
```

[/mw_shl_code]

【Session删除某个对象的方法】

[mw_shl_code=java,true] @Test

```
/**
```

```
 * 删除某个对象
```

```
 */
```

```
public void demo4(){
```

```
    Session session = HibernateUtils.openSession();
```

```
    Transaction tx = session.beginTransaction();
```

// 1.直接new 对象 删除

回复帖子...

回帖

```
customer.setCust_id(2l);
```

```
session.delete(customer);*/
```

// 2.先查询, 在删除 (推荐)

```
Customer customer = session.get(Customer.class, 3l);
```

```
session.delete(customer);
```

```
tx.commit();
```

```
session.close();
```

[/mw_shl_code]

1.2.4.4 Transaction:事务对象

Hibernate3的时候事务自动提交的参数是false（不自动提交）。Hibernate5可以自动提交，但是默认每个操作都是在一个事务中。一般需要手动开启事务。

```
[mw_shl_code=shell,true]commit();  
rollback();[/mw_shl_code]
```



回帖

— 9条回帖 —



liujinlong666

谢谢分享

沙发 • 2017-6-5 06:58:07

回帖



甲壳虫1993

谢谢楼主分享，有视频吗

藤椅 • 2017-6-5 15:47:38

回帖



dhj

感谢无私分享！！！！

板凳 • 2017-6-11 17:00:07

回帖



a851699

可以，很强势

报纸 • 2017-6-16 22:16:22

回帖



某某徐

commit();;;;;;;;;

地板 • 2017-6-20 21:16:49

回帖



apple4412

赞赞，希望继续分享

7# • 2017-6-21 21:33:19

回帖



孙阳超

谁有cas单点登录的视频啊,可以发我一份不

8# • 2017-6-22 15:58:31

回帖



sshsshssh

666666666666666666

9# • 2017-7-20 00:02:13

回帖



k275424

谢谢分享

10# • 2017-7-23 22:15:02

回帖