


SpringMVC-@Transaction声明事务的使用

 zhanglbjames (/u/4210f4fa15ae) +关注

0.2 2017.06.22 10:31* 字数 3566 阅读 942 评论 1 喜欢 3

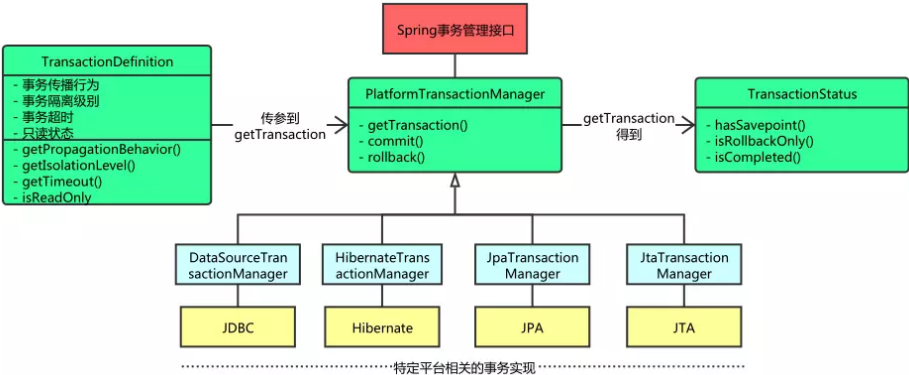
(/u/4210f4fa15ae)

1- 事务ACID

- 事务由一系列操作组成的，保证所有操作整体原子执行，完整的事务满足ACID特性
- 原子性（Atomicity）：事务是一个原子操作，由一系列动作组成。事务的原子性确保动作要么全部完成，要么完全不起作用。
 - 一致性（Consistency）：一旦事务完成（不管成功还是失败），系统必须确保它所建模的业务处于一致的状态，而不会是部分完成部分失败。在现实中的数据不应该被破坏。
 - 隔离性（Isolation）：可能有许多事务会同时处理相同的数据，因此每个事务都应该与其他事务隔离开来，防止数据损坏。
 - 持久性（Durability）：一旦事务完成，无论发生什么系统错误，它的结果都不应该受到影响，这样就能从任何系统崩溃中恢复过来。通常情况下，事务的结果被写到持久化存储器中。

2- 事务管理器

Spring并不直接管理事务，而是提供了多种事务管理器，他们将事务管理的职责委托给Hibernate或者JTA等持久化机制所提供的相关平台框架的事务来实现。



2.1- PlatformTransactionManager

org.springframework.transaction.PlatformTransactionManager是Spring事务管理器的接口，通过这个接口，Spring为各个平台如JDBC、Hibernate等都提供了对应的事务管理器，但是具体的实现就是各个平台自己的事情了。

(/apps/redirect?utm_source=side-banner-click)

PlatformTransactionManager接口

```
public interface PlatformTransactionManager()...{
    // 由TransactionDefinition得到TransactionStatus对象
    TransactionStatus getTransaction(TransactionDefinition definition)
        throws TransactionException;
    // 提交
    void commit(TransactionStatus status) throws TransactionException;
    // 回滚
    void rollback(TransactionStatus status) throws TransactionException;
}
```

提供了获取TransactionDefinition的方法以及根据TransactionStatus相应的状态进行提交或者回滚

TransactionStatus事务状态接口

上面讲到的调用PlatformTransactionManager接口的getTransaction()的方法得到的是TransactionStatus接口的一个实现，这个接口的内容如下：

```
public interface TransactionStatus{
    boolean isNewTransaction(); // 是否是新的事务
    boolean hasSavepoint(); // 是否有恢复点
    void setRollbackOnly(); // 设置为只回滚
    boolean isRollbackOnly(); // 是否为只回滚
    boolean isCompleted(); // 是否已完成
}
```

TransactionDefinition事务定义的接口



而TransactionDefinition接口内容如下：
属性：

```
package org.springframework.transaction;

import java.sql.Connection;

public interface TransactionDefinition {

    // 传播属性
    int PROPAGATION_REQUIRED = 0;

    int PROPAGATION_SUPPORTS = 1;

    int PROPAGATION_MANDATORY = 2;

    int PROPAGATION_REQUIRES_NEW = 3;

    int PROPAGATION_NOT_SUPPORTED = 4;

    int PROPAGATION_NEVER = 5;

    int PROPAGATION_NESTED = 6;

    // 隔离级别
    int ISOLATION_DEFAULT = -1;

    int ISOLATION_READ_UNCOMMITTED = Connection.TRANSACTION_READ_UNCOMMITTED;

    int ISOLATION_READ_COMMITTED = Connection.TRANSACTION_READ_COMMITTED;

    int ISOLATION_REPEATABLE_READ = Connection.TRANSACTION_REPEATABLE_READ;

    int ISOLATION_SERIALIZABLE = Connection.TRANSACTION_SERIALIZABLE;

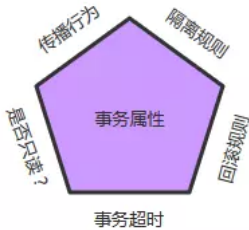
    // 连接超时
    int TIMEOUT_DEFAULT = -1;
    // 方法略
    ...
}
```

方法：

```
public interface TransactionDefinition {

    // 返回事务的传播行为
    int getPropagationBehavior();
    // 返回事务的隔离级别，事务管理器根据它来控制另外一个事务可以看到本事务内的哪些数据
    int getIsolationLevel();
    // 返回事务必须在多少秒内完成
    int getTimeout();
    // 事务是否只读，事务管理器能够根据这个返回值进行优化，确保事务是只读的
    boolean isReadOnly();
}
```

3- TransactionDefinition接口定义的事务的五个属性



3.1- 传播行为

当事务方法被另一个事务方法调用时，必须指定事务应该如何传播，没有默认值。例如：方法可能继续在现有事务中运行，也可能开启一个新事务，并在自己的事务中运行

(/apps/redirect?utm_source=side-banner-click) x

我们可以看 org.springframework.transaction.annotation.Propagation 枚举类中定义了6个表示传播行为的枚举值：

```
public enum Propagation {
    REQUIRED(0),
    SUPPORTS(1),
    MANDATORY(2),
    REQUIRES_NEW(3),
    NOT_SUPPORTED(4),
    NEVER(5),
    NESTED(6);
}
```

- REQUIRED：如果当前存在事务，则加入该事务；如果当前没有事务，则创建一个新的
 - 事务。
 - SUPPORTS：如果当前存在事务，则加入该事务；如果当前没有事务，则以非事务的方式继续运行。
 - MANDATORY：如果当前存在事务，则加入该事务；如果当前没有事务，则抛出异常。
 - REQUIRES_NEW：创建一个新的
 - 事务，如果当前存在事务，则把当前事务挂起。
 - NOT_SUPPORTED：以非事务方式运行，如果当前存在事务，则把当前事务挂起。
 - NEVER：以非事务方式运行，如果当前存在事务，则抛出异常。
 - NESTED：如果当前存在事务，则创建一个事务作为当前事务的嵌套事务来运行；如果当前没有事务，则该取值等价于 REQUIRED。
- 指定方法：通过使用 propagation 属性设置，例如：
- @Transactional(propagation = Propagation.REQUIRED)

3.2- 隔离级别

隔离级别是指若干个并发的

事务之间的隔离程度，与我们开发时候主要相关的场景包括：脏读取、重复读、幻读。

并发事务引起的问题

在典型的应用程序中，多个事务并发运行，经常会操作相同的数据来完成各自的任务。并发虽然是必须的，但可能会导致一下的问题。

- 脏读（Dirty reads）——脏读发生在一个事务读取了另一个事务改写但尚未提交的数据时。如果改写在稍后被回滚了，那么第一个事务获取的数据就是无效的。
- 不可重复读（Nonrepeatable read）——不可重复读发生在一个事务执行相同的查询两次或两次以上，但是每次都得到不同的数据时。这通常是因为另一个并发事务在两次查询期间进行了更新。
- 幻读（Phantom read）——幻读与不可重复读类似。它发生在一个事务（T1）读取了几行数据，接着另一个并发事务（T2）插入了一些数据时。在随后的查询中，第一个事务（T1）就会发现多了一些原本不存在的记录。



不可重复读与幻读的区别

(/apps/redirect?
utm_source=side-
banner-click)

- 不可重复读的重点是修改：
同样的条件, 你读取过的数据, 再次读取出来发现值不一样了, 在一个事务中前后两次读取的结果并不一致, 导致了不可重复读。
- 幻读的重点在于新增或者删除：
同样的条件, 第1次和第2次读出来的记录数不一样
- 从总的结果来看, 似乎不可重复读和幻读都表现为两次读取的结果不一致。但如果你从控制的角度来看, 两者的区别就比较大：
对于前者, 只需要锁住满足条件的记录。
对于后者, 要锁住满足条件及其相近的记录。

我们可以看 `org.springframework.transaction.annotation.Isolation` 枚举类中定义了五个表示隔离级别的值：

```
public enum Isolation {  
    DEFAULT(-1),  
    READ_UNCOMMITTED(1),  
    READ_COMMITTED(2),  
    REPEATABLE_READ(4),  
    SERIALIZABLE(8);  
}
```

- **DEFAULT**：这是默认值，表示使用底层数据库的默认隔离级别。对大部分数据库而言，通常这值就是：**READ_COMMITTED**，**Mysql**为**REPEATABLE_READ**。
- **READ_UNCOMMITTED**：该隔离级别表示一个事务可以读取另一个事务修改但还没有提交的数据。该级别不能防止脏读和不可重复读，因此很少使用该隔离级别。
- **READ_COMMITTED**：该隔离级别表示一个事务只能读取另一个事务已经提交的数据。该级别可以防止脏读，这也是大多数情况下的推荐值。
- **REPEATABLE_READ**：该隔离级别表示一个事务在整个过程中可以多次重复执行某个查询，并且每次返回的记录都相同。即使在多次查询之间有新增的数据满足该查询，这些新增的记录也会被忽略。该级别可以防止脏读和不可重复读。
- **SERIALIZABLE**：所有的事务依次逐个执行，这样事务之间就完全不可能产生干扰，也就是说，该级别可以防止脏读、不可重复读以及幻读。但是这将严重影响程序的性能。通常情况下也不会用到该级别。

指定方法：通过使用 `isolation` 属性设置，例如：

```
@Transactional(isolation = Isolation.DEFAULT)
```

3.3- 只读

事务的第三个特性是它是否为只读事务。如果事务只对后端的数据库进行该操作，数据库可以利用事务的只读特性来进行一些特定的优化。通过将事务设置为只读，你就可以给数据库一个机会，让它应用它认为合适的优化措施。

3.4- 事务超时

为了使应用程序很好地运行，事务不能运行太长的时间。因为事务可能涉及对后端数据库的锁定，所以长时间的事务会不必要的占用数据库资源。**事务超时就是事务的一个定时器，在特定时间内事务如果没有执行完毕，那么就会自动回滚，而不是一直等待其结束。**



2. 显示的指明事务管理器

如果你项目做的比较大，添加的持久化依赖比较多，我们还是会选择人为的指定使用哪个事务管理器

(/apps/redirect?
utm_source=side-
banner-click)

```
@EnableTransactionManagement
@SpringBootApplication
public class ProfiledemoApplication {

    @Bean // 其中 dataSource 框架会自动为我们注入
    public PlatformTransactionManager txManager(dataSource dataSource) {
        return new DataSourceTransactionManager(dataSource);
    }

    @Bean
    public Object testBean(PlatformTransactionManager platformTransactionManager) {
        System.out.println(">>>>>>> " + platformTransactionManager.getClass().getName());
        return new Object();
    }

    public static void main(String[] args) {
        SpringApplication.run(ProfiledemoApplication.class, args);
    }
}
```

在Spring容器中，我们手工注解@Bean 将被优先加载，框架不会重新实例化其他的 PlatformTransactionManager 实现类。

4.3- 多事务管理器的配置

对于同一个工程中存在多个事务管理器进行配置

```
@EnableTransactionManagement // 开启注解事务管理, 等同于xml配置文件中的 <tx:annotation-driven />
@SpringBootApplication
public class ProfiledemoApplication implements TransactionManagementConfigurer {

    @Resource(name="txManager2")
    private PlatformTransactionManager txManager2;

    // 创建事务管理器1
    @Bean(name = "txManager1")
    public PlatformTransactionManager txManager1(DataSource dataSource) {
        return new DataSourceTransactionManager(dataSource);
    }

    // 创建事务管理器2
    @Bean(name = "txManager2")
    public PlatformTransactionManager txManager2(EntityManagerFactory factory) {
        return new JpaTransactionManager(factory);
    }

    // 实现接口 TransactionManagementConfigurer 方法, 其返回值代表在拥有多个事务管理器的情况下默认使用的事务管理器
    @Override
    public PlatformTransactionManager annotationDrivenTransactionManager() {
        return txManager2;
    }

    public static void main(String[] args) {
        SpringApplication.run(ProfiledemoApplication.class, args);
    }
}
```

启动类实现TransactionManagementConfigurer接口, 通过@Bean注解注入两个事务管理器, 实现annotationDrivenTransactionManager接口方法, 指定value缺省值情况下提供的默认事务管理器, 如下:

```
@Component
public class DevSendMessage implements SendMessage {

    // 使用value具体指定使用哪个事务管理器
    @Transactional(value="txManager1")
    @Override
    public void send() {
        System.out.println(">>>>>>>Dev Send()<<<<<<<");
        send2();
    }

    // 在存在多个事务管理器的情况下, 如果使用value具体指定
    // 则默认使用方法 annotationDrivenTransactionManager() 返回的事务管理器
    @Transactional
    public void send2() {
        System.out.println(">>>>>>>Dev Send2()<<<<<<<");
    }
}
```

注:

如果Spring容器中存在多个 PlatformTransactionManager 实例, 并且没有实现接口 TransactionManagementConfigurer 指定默认值, 我们在方法上使用注解 @Transactional 的时候, 就必须要用value指定, 如果不指定, 则会抛出异常。

4.4- @Transaction 注解使用的注意事项

@Transactional属性

(/apps/redirect?
utm_source=side-
banner-click)



属性	类型	描述
value	String	可选的限定描述符，指定使用的事务管理器
propagation	enum: Propagation	可选的事务传播行为设置
isolation	enum: Isolation	可选的事务隔离级别设置
readOnly	boolean	读写或只读事务，默认读写
timeout	int (in seconds granularity)	事务超时时间设置
rollbackFor	Class对象数组，必须继承自Throwable	导致事务回滚的异常类数组
rollbackForClassName	类名数组，必须继承自Throwable	导致事务回滚的异常类名字数组
noRollbackFor	Class对象数组，必须继承自Throwable	不会导致事务回滚的异常类数组
noRollbackForClassName	类名数组，必须继承自Throwable	不会导致事务回滚的异常类名字数组

(/apps/redirect?utm_source=side-banner-click)

noRollback示例

```
// 执行哪些异常不会回滚示例
@Transactional(noRollbackForClassName={"RollBackException1,RollBackException2"})
或者
@Transactional(noRollbackFor={RollBackException1.class,RollBackException2.class})
```

@Transactional 可以作用于接口、接口方法、类以及类方法上。当作用于类上时，该类的所有 public 方法将都具有该类型的事务属性，同时，我们也可以在方法级别使用该标注来覆盖类级别的定义。

最佳实践：

- 虽然 @Transactional 注解可以作用于接口、接口方法、类以及类方法上，但是 Spring 建议不要在接口或者接口方法上使用该注解，因为这只有在使用基于接口的代理时它才会生效。
- @Transactional 注解应该只被应用到 public 方法上，这是由 Spring AOP 的本质决定的。如果你在 protected、private 或者默认可见性的方法上使用 @Transactional 注解，这将被忽略，也不会抛出任何异常。
- 默认情况下，只有来自外部的方法调用才会被AOP代理捕获，也就是，类内部方法调用本类内部的其他方法并不会引起事务行为，即使被调用方法使用@Transactional注解进行修饰。

事务异常排查列表：

1. 数据库本身是否支持事务，如mysql的myisam引擎就不支持事务
2. 多数据源，是否指定了正确的事务管理器
3. 数据源配置是否正确（MyBatis的SqlSessionFactoryBean引用的数据源与DataSourceTransactionManager引用的数据源是否一致）
4. 私有方法不会生效事务（AOP默认不代理私有方法，也不会抛出异常）
5. 被调用方法不会生效事务（this指针调用方法，并没有使用AOP代理来执行方法）
6. 不指明回滚的异常，方法内用户抛出的checked异常不会生效回滚（默认只有unchecked异常发生回滚）

MyBatis自动参与到spring事务管理中，无需额外配置，只要org.mybatis.spring.SqlSessionFactoryBean引用的数据源与DataSourceTransactionManager引用的数据源一致即可，否则事务管理会不起作用。



@Transaction示例

方法上注解属性会覆盖类注解上的相同属性

```
// 事务属性值覆盖
@Transactional(readonly = true)
public class DefaultFooService implements FooService {

    public Foo getFoo(String fooName) {
        // do something
    }

    // these settings have precedence for this method
    // 方法上注解属性会覆盖类注解上的相同属性
    @Transactional(readonly = false, propagation = Propagation.REQUIRES_NEW)
    public void updateFoo(Foo foo) {
        // do something
    }
}
```

事务管理器保证事务方法内的SQL在同一个SqlSession中，相当于：

```
// 事务管理器保证事务方法内的SQL 在同一个SqlSession中，相当于：
Main{
    Connection con=null;
    try{
        con = getConnection();
        con.setAutoCommit(false);

        // 方法调用
        updateFoo();

        // 提交事务
        con.commit();
    } catch(RuntimeException ex) {
        // 回滚事务
        con.rollback();
    } finally {
        // 释放资源
        closeCon();
    }
}
```

参考链接
Spring声明式事务为何不回滚 (<https://www.jianshu.com/p/f5fc14bde8a0>)

小礼物走一走，来简书关注我

赞赏支持

 JavaWeb相关 (/nb/12782352)

举报文章

© 著作权归作者所有



zhangbjames (/u/4210f4fa15ae) ♂

写了 144943 字，被 81 人关注，获得了 206 个喜欢

(/u/4210f4fa15ae)

+ 关注

zhangbjames@163.com



喜欢 | 3



更多分享

(/apps/redirect?utm_source=side-banner-click)



(/apps/redirect?utm_source=note-bottom-click)




登录 (/sign-in?utm_source=desktop&utm_medium=not-signed-in-comment-form)

1条评论

只看作者

按时间倒序 按时间正序

 zhanglbjames (/u/4210f4fa15ae) 作者


2楼 · 2017.07.11 11:50

(/u/4210f4fa15ae)

一起学习😊

赞 回复

被以下专题收入，发现更多相似内容

 spring (/c/2572fa7868d4?utm_source=desktop&utm_medium=notes-included-collection)

推荐阅读

更多精彩内容 > (/)

【简书钻笔记033】怎样让更多人关注你？一个月涨粉2000人（以我的简书...

(/p/d6a2247a1378?utm_campaign=maleskine&utm_content=note&utm_medium=pc_all_hots&utm_source=recommendation)

在讲我的简书怎么涨粉之前，如下图，目前日更33天，粉丝2159人 我先讲一下微博的二三事哈。据我所知，微博早年兴起之时，很多草根博主靠强大的执行力，无所不用其极，通过各种方式增加自己的粉丝...
加油鸭CPA (/u/8a771f2da27d?)

李清照：一路写尽诗情画意，半世过完颠沛流离（下） ...

(/p/d6a2247a1378?utm_campaign=maleskine&utm_content=note&utm_medium=pc_all_hots&utm_source=recommendation)

文/麦大人 05 《少有人走的路》作者在书中说，人生苦难重重，这是世界上最伟大的真理之一。前半生的李清照无疑极受上天垂怜，他们的生活如果一直...
麦大人 (/u/2b3ad4f2a058?)

砍柴书院征文|在生命的最低谷起跳 (/p/f606f269a355?...

(/p/f606f269a355?utm_campaign=maleskine&utm_content=note&utm_medium=pc_all_hots&utm_source=recommendation)

我有一位朋友，四年前被单位派往偏远的山区支教。接到通知后他极端不情愿，去找领导理论说，我刚好在工作上走上正轨，正处在事业的上升期，去那个鬼...
云岭观世 (/u/3bd75eab1500?)

报班 (/p/45def810f881?utm_campaign=maleskine&utm_content=note&...

“小明，这个学期要不要给你报个培训班，补补后进的课？”“妈，我用不着报那个，我的每门功课都很平均，根本就没有拖后腿的。”“你的语文50分，数学50分，科学50分……门门都50分，是够平均的……你真好意...

过期废柴 (/u/b74a0510f9f2?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

(/apps/redirect?utm_source=side-banner-click)

×

午夜遐想 (/p/ff397ac1fe57?utm_campaign=maleski...

文/苏若儿 一些残留的记忆总喜欢躲在时光的角落在夜深人静时，轻轻铺展成画卷里，风与云的追逐 清晰的昨日，如星辰散发微妙的光，在眼眸流淌倚着月...

苏若儿 (/u/9b487134d84b?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

(/p/ff397ac1fe57?utm_campaign=maleskine&utm_content=note&utm_...

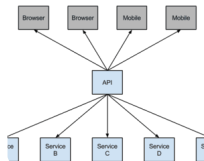
Spring boot参考指南 (/p/67a0e41dfe05?utm_campaign=maleskine&ut...

Spring Boot 参考指南 介绍 转载自:https://www.gitbook.com/book/qbgbook/spring-boot-reference-guide-zh/details带目录浏览地址:http://www.maoyupeng.com/sprin...

毛宇鹏 (/u/d3ea915e1e0f?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/67a0e41dfe05?utm_campaign=maleskine&utm_content=note&utm_...

(/p/46fd0faecac1?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

Spring Cloud (/p/46fd0faecac1?utm_campaign=maleskine&utm_conte...

Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线）。分布式系统的协调导致了样板模式，使用Spring Cloud开发人员可...

卡卡罗2017 (/u/d90908cb0d85?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/46fd0faecac1?utm_campaign=maleskine&utm_content=note&utm_...

(/p/b4b7b7d3f1bd?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

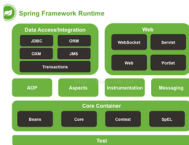
第五部分：数据访问 (/p/b4b7b7d3f1bd?utm_campaign=maleskine&utm...

这部分的参考文档涉及数据访问和数据访问层和业务或服务层之间的交互。
Spring的综合事务管理支持覆盖很多细节，然后覆盖了各种数据访问框架和...



utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)


(/p/094b68778ef4?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

无标题文章 (/p/094b68778ef4?utm_campaign=maleskine&utm_content...

spring官方文档：http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/一、Spring 框架概述Spring框架是一个轻量级的解决方案，可以一站式地构建企业级应用。...

牛马风情 (/u/f109f1576a46?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/05b70834dafe?


更新丢失	脏读	不可重复读	幻读	实现力
N	Y	Y	Y	排他写
N	N	Y	Y	锁间共
N	N	N	Y	共享读
N	N	N	N	?

(/apps/redirect?utm_source=side-banner-click)

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)


由浅入深谈论spring事务 (/p/05b70834dafe?utm_campaign=maleskine&...

很多人喜欢这篇文章，特此同步过来 由浅入深谈论spring事务 前言 这篇其实也要归纳到《常识》系列中，但这重点又是spring的介绍，故归档在spring系列中。工作很多年，除了学生时代学过，事务还真没有用过...

码农戏码 (/u/c7137c71b8f0?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

无标题文章 (/p/4ebcb13f0595?utm_campaign=maleskine&utm_content=...

不要去抱怨生活 .其实生活中的不好之处都是好的 要懂得去享受它 当我们去享受它的时候就不会抱怨了 把抱怨变成享受

我在看书而你在看手机 (/u/e55266508c78?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)


(/p/ccdb910ef78f?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

解析移动端车牌识别技术在警务通中的应用方式 (/p/ccdb910ef78f?utm_ca...

通常交警在处理路边违章停车的情况时，采用的是传统的手工录入车牌信息的方式，在面对庞大数量的汽车时显得力不从心，目前，我国警务通、停车场手持收费机等移动终端的使用比较普及，如果在这些终端上...

OCR智能识别 (/u/556186219d2b?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)


(/p/26dcfcb96f37?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)


isn't me (/p/26dcfcb96f37?utm_campaign=maleskine&utm_content=n...

回忆在散发恨的魔力，时间洗去伤口的血迹，一双手写不出传奇，一个背影去铺叙成悲剧；枫叶上你残留的气息，闭眼缓缓碾成了碎屑，你在渲染残余的演技，让你成为受伤的主角。不曾有的唯一，被我心痛着...

淑淑女子 (/u/5f883ae7f9d9?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)


读《三国》品人生 (/p/409768462525?utm_campaign=maleskine&utm_c...

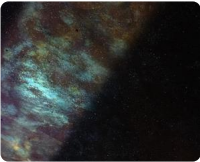
曹操代表欲望，刘备代表信仰，而孙权更多代表现实。三人构成了江山，而三者也构成了我们的人生，欲望、信仰和现实三足鼎立的人生。鲁肃在临死的时候交代给孙权，魏蜀吴三足鼎立之势中，魏和蜀孰弱便...

八佾舞庭 (/u/0814e4422634?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/284768fb7e64?
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
青春如光 (/p/284768fb7e64?utm_campaign=maleskine&utm_content=n...

青春如光 文/@姜来姜姜姜姜 青春是从什么时候算起的？女生第一次姨妈？男生
第一次遗精？还是彼此第一次的偷尝禁果？又或者是第一次冲撞老师，第一次...

 姜来姜姜姜姜 (/u/5057d819a3a4?)



(/apps/redirect?
utm_source=side-
banner-click) ×

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

