

Spring Boot实践，开发社区核心功能

牛客Java高级工程师 第三章

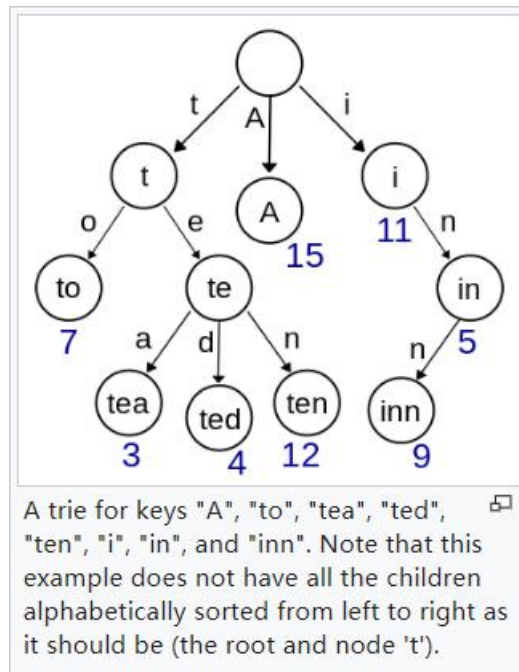
1. 过滤敏感词

- 前缀树

- 名称: Trie、字典树、查找树
- 特点: 查找效率高, 消耗内存大
- 应用: 字符串检索、词频统计、字符串排序等

- 敏感词过滤器

- 定义前缀树
- 根据敏感词, 初始化前缀树
- 编写过滤敏感词的方法

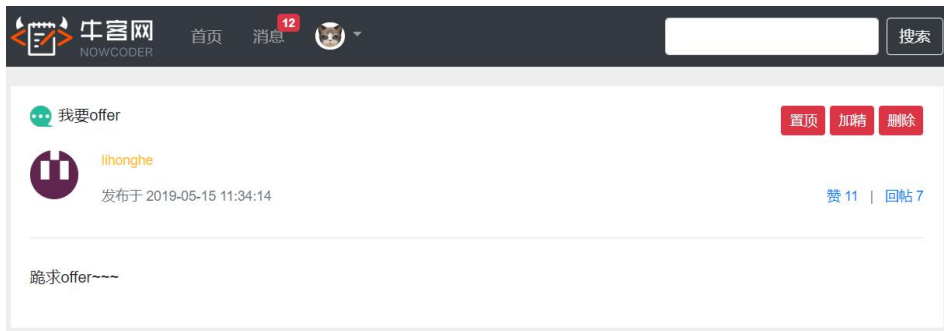


► 2. 发布帖子

- AJAX
 - Asynchronous JavaScript and XML
 - 异步的JavaScript与XML，不是一门新技术，只是一个新的术语。
 - 使用AJAX，网页能够将增量更新呈现在页面上，而不需要刷新整个页面。
 - 虽然X代表XML，但目前JSON的使用比XML更加普遍。
 - <https://developer.mozilla.org/zh-CN/docs/Web/Guide/AJAX>
- 示例
 - 使用jQuery发送AJAX请求。
- 实践
 - 采用AJAX请求，实现发布帖子的功能。

► 3. 帖子详情

- DiscussPostMapper
- DiscussPostService
- DiscussPostController
- index.html
 - 在帖子标题上增加访问详情页面的链接
- discuss-detail.html
 - 处理静态资源的访问路径
 - 复用index.html的header区域
 - 显示标题、作者、发布时间、帖子正文等内容



► 4. 事务管理

回顾

- 什么是事务
 - 事务是由N步数据库操作序列组成的逻辑执行单元，这系列操作要么全执行，要么全放弃执行。
- 事务的特性 (ACID)
 - 原子性 (Atomicity)：事务是应用中不可再分的最小执行体。
 - 一致性 (Consistency)：事务执行的结果，须使数据从一个一致性状态，变为另一个一致性状态。
 - 隔离性 (Isolation)：各个事务的执行互不干扰，任何事务的内部操作对其他的事务都是隔离的。
 - 持久性 (Durability)：事务一旦提交，对数据所做的任何改变都要记录到永久存储器中。

► 4. 事务管理

事务的隔离性

- 常见的并发异常
 - 第一类丢失更新、第二类丢失更新。
 - 脏读、不可重复读、幻读。
- 常见的隔离级别
 - Read Uncommitted: 读取未提交的数据。
 - Read Committed: 读取已提交的数据。
 - Repeatable Read: 可重复读。
 - Serializable: 串行化。

► 4. 事务管理

第一类丢失更新

某一个事务的回滚，
导致另外一个事务已更新的数据丢失了。

时刻	事务1	事务2
T1	Read: N = 10	
T2		Read: N = 10
T3		Write: N = 9
T4		Commit: N = 9
T5	Write: N = 11	
T6	Rollback: N = 10	

► 4. 事务管理

第二类丢失更新

某一个事务的提交，
导致另外一个事务已更新的数据丢失了。

时刻	事务1	事务2
T1	Read: N = 10	
T2		Read: N = 10
T3		Write: N = 9
T4		Commit: N = 9
T5	Write: N = 11	
T6	Commit: N = 11	

► 4. 事务管理

脏读

某一个事务，
读取了另外一个事务未提交的数据。

时刻	事务1	事务2
T1	Read: N = 10	
T2	Write: N = 11	
T3		Read: N = 11
T4	Rollback: N = 10	

► 4. 事务管理

不可重复读

某一个事务，

对同一个数据前后读取的结果不一致。

时刻	事务1	事务2
T1	Read: N = 10	
T2		Read: N = 10
T3	Write: N = 11	
T4	Commit: N = 11	
T5		Read: N = 11

► 4. 事务管理

幻读

某一个事务，

对同一个表前后查询到的行数不一致。

时刻	事务1	事务2
T1		Select: id < 10 (1,2,3)
T2	Insert: id = 4	
T3	Commit: id = (1,2,3,4)	
T4		Select: id < 10 (1,2,3,4)

► 4. 事务管理

事务隔离级别

隔离级别	第一类丢失更新	脏读	第二类丢失更新	不可重复读	幻读
Read Uncommitted	Y	Y	Y	Y	Y
Read Committed	N	N	Y	Y	Y
Repeatable Read	N	N	N	N	Y
Serializable	N	N	N	N	N

► 4. 事务管理

实现机制

- 悲观锁（数据库）

- 共享锁（S锁）

事务A对某数据加了共享锁后，其他事务只能对该数据加共享锁，但不能加排他锁。

- 排他锁（X锁）

事务A对某数据加了排他锁后，其他事务对该数据既不能加共享锁，也不能加排他锁。

- 乐观锁（自定义）

- 版本号、时间戳等

在更新数据前，检查版本号是否发生变化。若变化则取消本次更新，否则就更新数据（版本号+1）。

► 4. 事务管理

Spring事务管理

- 声明式事务
 - 通过XML配置，声明某方法的事务特征。
 - 通过注解，声明某方法的事务特征。
- 编程式事务
 - 通过 TransactionTemplate 管理事务，并通过它执行数据库的操作。

Table of Contents

[◀ Back to index](#)

1. Transaction Management

- 1.1. Advantages of the Spring Framework's Transaction Support Model
- 1.2. Understanding the Spring Framework Transaction Abstraction
- 1.3. Synchronizing Resources with Transactions
- 1.4. Declarative transaction management
- 1.5. Programmatic Transaction Management
- 1.6. Choosing Between Programmatic and Declarative Transaction Management
- 1.7. Transaction-bound Events
- 1.8. Application server-specific integration
- 1.9. Solutions to Common Problems
- 1.10. Further Resources
- 2. DAO Support
- 3. Data Access with JDBC
- 4. Object Relational Mapping (ORM) Data Access
- 5. Marshalling XML by Using Object-XML Mappers
- 6. Appendix

► 5. 显示评论

- 数据层
 - 根据实体查询一页评论数据。
 - 根据实体查询评论的数量。
- 业务层
 - 处理查询评论的业务。
 - 处理查询评论数量的业务。
- 表现层
 - 显示帖子详情数据时，
同时显示该帖子所有的评论数据。



► 6. 添加评论

- 数据层
 - 增加评论数据。
 - 修改帖子的评论数量。
- 业务层
 - 处理添加评论的业务：
先增加评论、再更新帖子的评论数量。
- 表现层
 - 处理添加评论数据的请求。
 - 设置添加评论的表单。

在这里畅所欲言你的看法吧!

回 帖

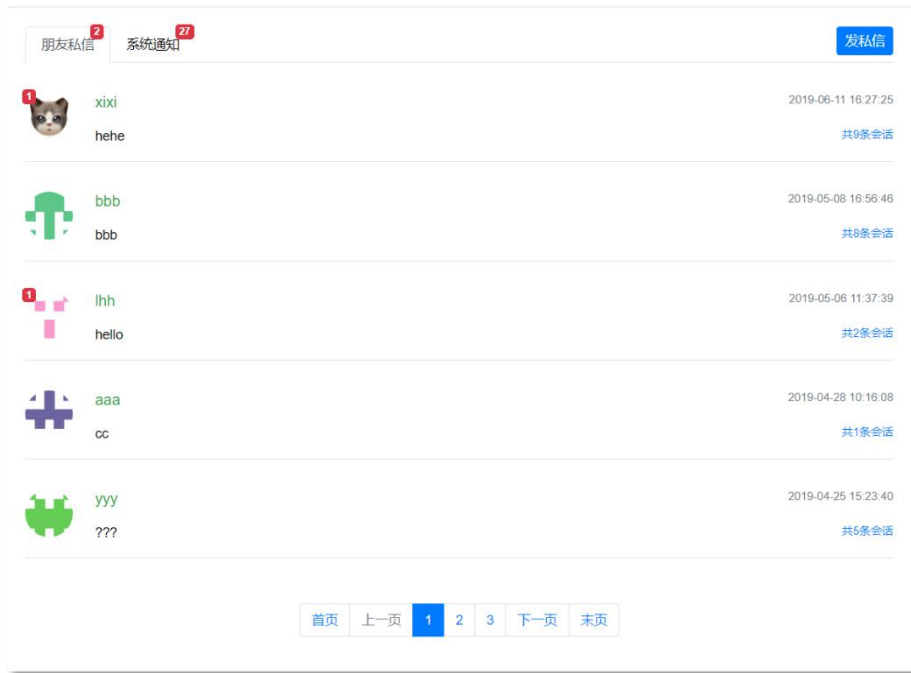
▶ 7. 私信列表

- 私信列表

- 查询当前用户的会话列表，
每个会话只显示一条最新的私信。
- 支持分页显示。

- 私信详情

- 查询某个会话所包含的私信。
- 支持分页显示。



► 8. 发送私信

- 发送私信
 - 采用异步的方式发送私信。
 - 发送成功后刷新私信列表。
- 设置已读
 - 访问私信详情时，
将显示的私信设置为已读状态。

发私信

发给:

内容:

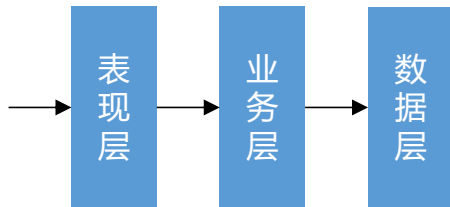
取消

发送

► 9. 统一处理异常

- @ControllerAdvice

- 用于修饰类，表示该类是Controller的全局配置类。
- 在此类中，可以对Controller进行如下三种全局配置：
异常处理方案、绑定数据方案、绑定参数方案。



- @ExceptionHandler

- 用于修饰方法，该方法会在Controller出现异常后被调用，用于处理捕获到的异常。

- @ModelAttribute

- 用于修饰方法，该方法会在Controller方法执行前被调用，用于为Model对象绑定参数。

- @DataBinder

- 用于修饰方法，该方法会在Controller方法执行前被调用，用于绑定参数的转换器。

► 10. 统一记录日志

需求

帖子
模块

评论
模块

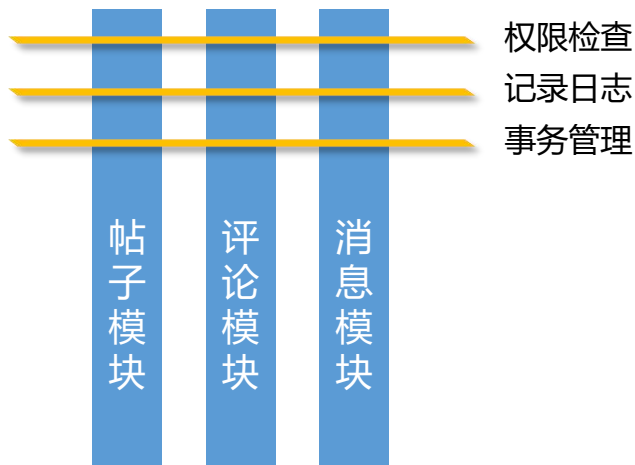
消息
模块

```
public class AlphaService {  
  
    public void doSomething() {  
        记录日志 ...  
        处理业务 ...  
    }  
  
}
```

► 10. 统一记录日志

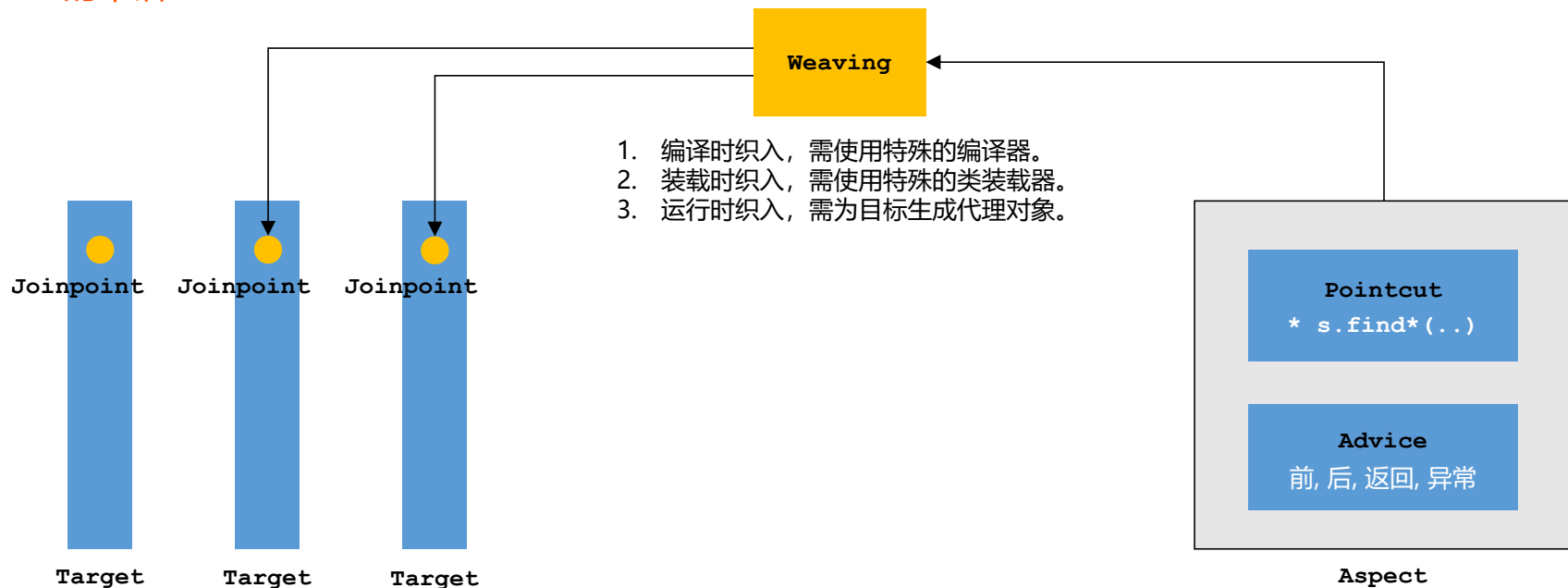
AOP的概念

- Aspect Oriented Programing,
即面向方面（切面）编程。
- AOP是一种编程思想，是对OOP的补充，
可以进一步提高编程的效率。



► 10. 统一记录日志

AOP的术语



► 10. 统一记录日志

AOP的实现

- AspectJ
 - AspectJ是语言级的实现，它扩展了Java语言，定义了AOP语法。
 - AspectJ在编译期织入代码，它有一个专门的编译器，用来生成遵守Java字节码规范的class文件。
- Spring AOP
 - Spring AOP使用纯Java实现，它不需要专门的编译过程，也不需要特殊的类装载器。
 - Spring AOP在运行时通过代理的方式织入代码，只支持方法类型的连接点。
 - Spring支持对AspectJ的集成。

► 10. 统一记录日志

Spring AOP

- JDK动态代理
 - Java提供的动态代理技术，可以在运行时创建接口的代理实例。
 - Spring AOP默认采用此种方式，在接口的代理实例中织入代码。
- CGLib动态代理
 - 采用底层的字节码技术，在运行时创建子类代理实例。
 - 当目标对象不存在接口时，Spring AOP会采用此种方式，在子类实例中织入代码。

Thanks

