### Java 8

#### Lambda表达式简介。

要理解Lambda表达式首先必须明白行为参数化的意义；在软件工程中，用户的需求随时会改变，行为参数化可以帮助您应对不断变化的需求。Lambda表达式就是行为参数化概念的一种解决方案。

#### 什么是函数式接口。

只定义了一个抽象方法的接口，称作函数式接口，函数式接口可以描述Lambda表达式的签名；常用的函数式接口有Predicate — 定义了一个test方法、Consumer — 接收一个泛型参数没有任何返回值、

Funtion — 接收一个泛型T对象，返回一个泛型R对象。

#### 什么方法引用。

方法引用可以看做是Lambda表达式的一种简写方式。

#### 什么是默认方法。

接口中使用关键字 default 修饰的方法，称作默认方法。我们知道向已经发布的接口添加一个方法是非常麻烦的事情，因为所有的子类都必须实现这个方法；有了默认方法之后，我们在已经发布的接口中添加默认方法，则之前改接口的子类不会受到任何影响。

#### 流是什么。

通过声明的方式来处理集合，我们可以把它看做是遍历数据集的高级迭代器，此外流是可以并行的处理的。

#### 流和集合的区别。

集合和流之间的差异就在于什么时候进行计算，集合中的元素必须先计算出来才能添加到集合中。与之相比，流则是概念上的数据结构(不能添加或者删除)，其元素是按需计算的，只有在需要的时候才会计算值。

打个比方吧，流好比就是DVD里面的电影；而流就好比在互联网上观看电影，只需要提前下载用户观看位置的几帧就可以了。

流只能遍历一次，如果重新遍历一次，则抛出流已经被操作或者关闭的异常。

Java流采用了内部迭代的方式(集合采用的是外部迭代的方式)，内部迭代的优点在于可以在内部做很多优化处理。

#### 流提供了那些功能。

流提供了诸如筛选、匹配、映射、分组、分区等等功能。

#### 构建流的方式。

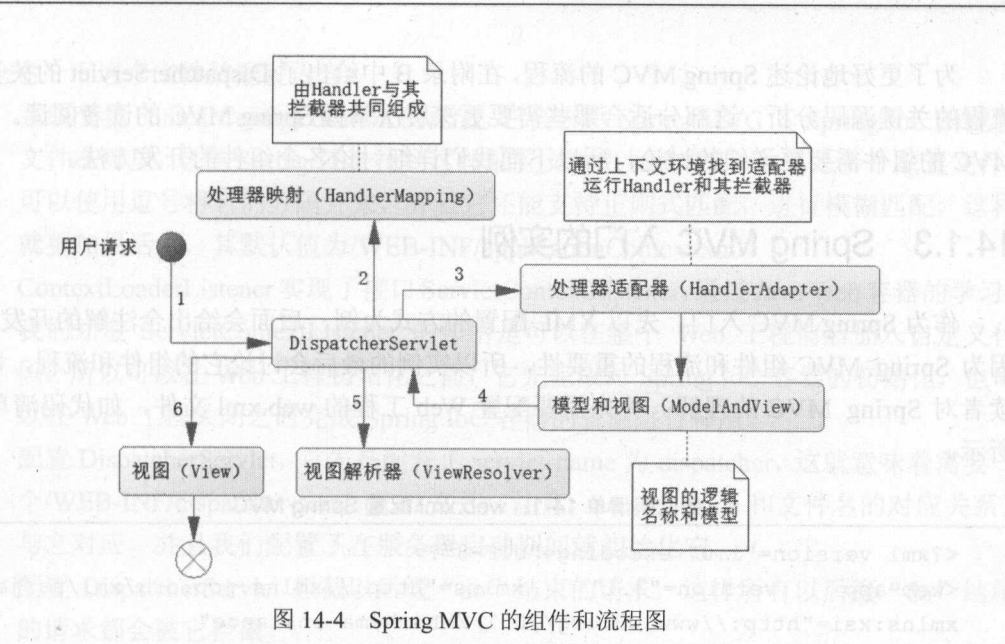
1. 通过值创建流。
2. 通过数组和集合创建流。
3. 通过文件创建流。

### Spring MVC

#### Spring MVC 的组件和流程。

Spring MVC是一种基于Servlet的技术，它提供了核心控制器DispatcherServlet; HandlerMapping、HandlerAdapter、ModelAndView、ViewResolver、View等相关组件。

Spring MVC 流程如下：



1. 初始化的时候根据配置文件生成HandlerMapping。
2. DispatcherServlet 首先通过请求和事先解析好的 HandlerMapping配置，找到对应的处理器（ Handler），这样就准备开始运行处理器和拦截器组成的执行链而运行处理器需要有个对应的环境这样它就有了一个处理器的适配器（HandlerAdapter ），通过这个适配器就能运行对应的处理器及其拦截器，这里的处理器包含了控制器的内容和其他增强的功能。
3. 处理器返回ModelAndView给DispacherServlet 。
4. 通过视图解析器得到View，默认返回的是jstlView视图。

#### 2.Spring MVC 接收请求参数的方式有哪些？

1、 不需要任何注解，只需要前、后的参数名保持一致即可接收参数。这正是Spring MVC的智能之处。

2、 使用＠RequestParam注解接收参数。

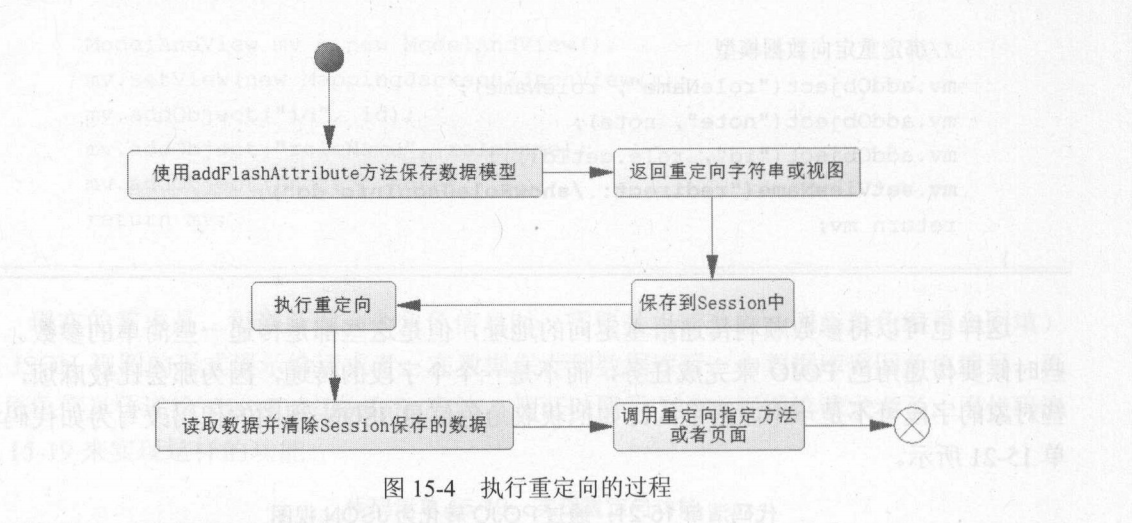
3、 使用 URL 传递参数（@PathVariable）。

4、 使用JSON传递参数（＠RequestBody）。

#### 3.Spring MVC 重定向和重定向时的数据传输？

Spring MVC有一个约定，当返回的字符串中有一个redirect的时候，它就会认为需要的是一个重定向。

在URL重定向的时候，并不能有效传递对象，Spring MVC为我们提供了flash属性，用来在重定向的时候传递对象。其原理如下：

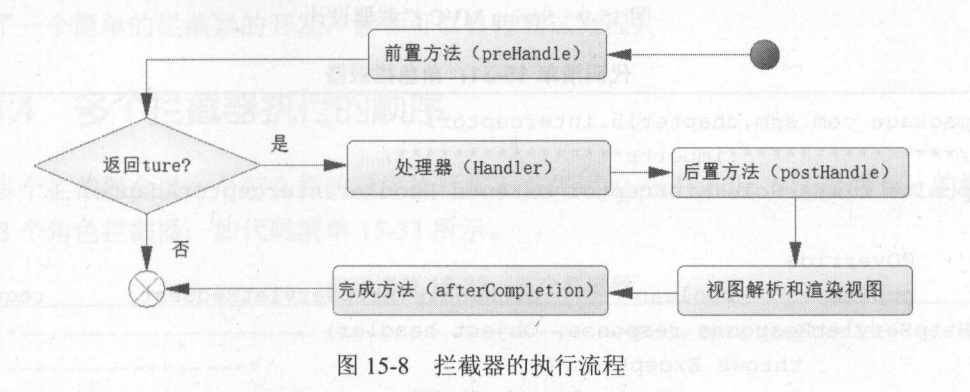


#### 4.Spring MVC 保存并获取属性参数？

Spring MVC 提供了5个注解，从request、session中读取数据或者保存数据到request、session中；它们分别是@requestAttribute(获取http请求中的属性值)、 @SessionAttribute（获取session中的属性值）、 @SessionAttributes(保存属性值到session中)、＠CookieValue(获取Cookie中的属性值)、＠RequestHeader(获取request头中的属性值)。

#### 5.Spring MVC 拦截器？

Spring MVC 拦截器需要实现HandlerinterceptorAdapter接口，下面是单个拦截器的执行流程（拦截器可以有多个）：



#### Spring MVC 表单校验？

Spring MVC提供了两种校验方式，一是使用 JSR 303 注解验证输入内容，二是使用Spring MVC提供的验证器（自定义的验证器必须实现Validator接口）。

#### Spring MVC 数据模型？

所谓数据模型，实际上就是需要展示在前端页面上的数据，Spring MVC 通过 Model或ModelMap来构建数据模型； Model 和 ModelMap在本质上并没有区别。

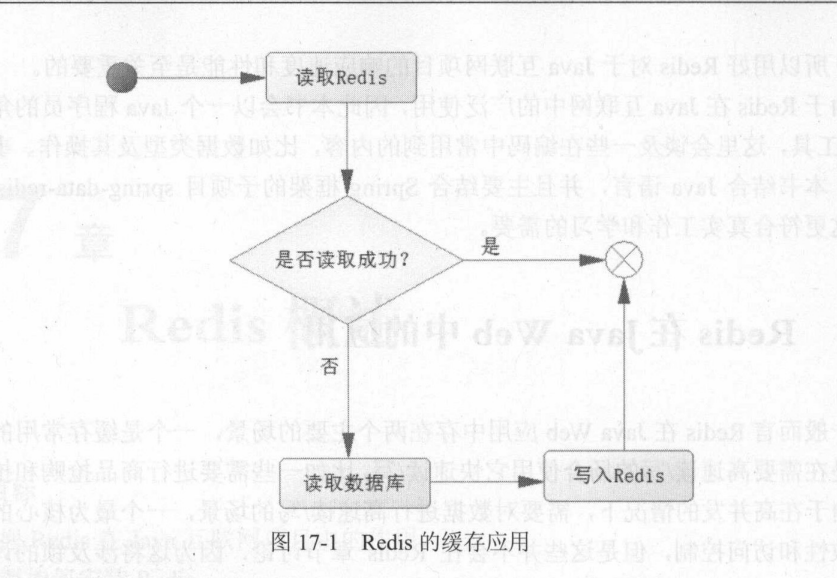
#### Spring MVC 有哪些视图？

Spring MVC视图可以分作逻辑视图、非逻辑视图；JstlView、 IntemalResourceView（**逻辑视图**） 是父子类关系，所以它们可以归为一类，它们主要是为了 JSP 渲染而服务的；MappingJackson2J son View （**非逻辑视图**）则是一个 JSON 视图 ；当然Spring MVC还有其它等等一些不常用的视图。

### Reids3.0

#### Redis在Java Web中的应用。

* 1. **缓存。**

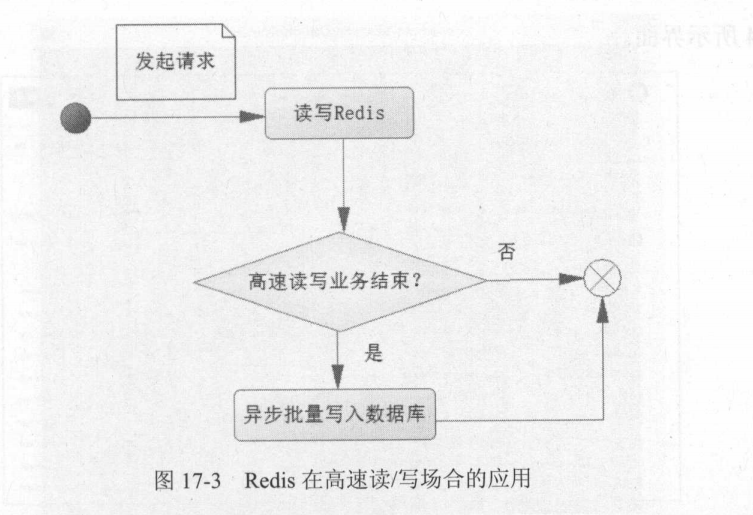


第一次读取数据的时候，读取redis的数据就会失败，此时会触发程序读取数据库，把数据读取出来，写入redis。

第二次及以后读取数据时，就直接读取redis.

* 1. **高速读/写场合。**

在互联网应用中往往存在着一些需要高速读/写的场合，比如商品的秒杀、抢红包、春运抢票等等。



假设面对的是一个商品秒杀的场景，从上面的流程来看，一个用户抢购商品，绝大部分的场合都是在操作内存数据库，所以性能会更加优越。只有在商品抢购一空后才会触发系统把Redis缓存的数据写入数据库磁盘中。

而在现实中需要高速响应的系统比上面的分析更加复杂，需要解决高并发下数据的安全性和一致性、有效请求、无效请求、事务一致性等诸多问题；

我们可以通过验证码、短信、单位时间内限制用户的访问量等形式过滤无效的请求。

#### Redis 的优点。

速度快、高可用和分布式(redis3.0 提供了redis的集群实现)、客户端语言多、简单稳定(单线程)、功能丰富（支持5种数据结构、提供发布订阅功能、流水线PipeLine、Lua脚本）。

#### Redis 能代替传统关系型数据库吗？

暂时无法代替，Redis是基于内存的数据库，无法支持大容量存储、没有像SQL那样强大的脚本语言、数据结构比较简单等等这些原因，注定其无法替代传统的关系型数据库。

#### Redis 的数据类型？

字符串、哈希表、列表、集合、有序集合、基数六种数据类型。

#### Redis 事物简单介绍

redis事务执行分作三步。开启事务（multi）、命令进入队列、执行事务（exec）。

redis可以使用watch命令监控事务，通过判断监控的key值是否一致，从而决定是否回滚事务。

redis的事物回滚只是对命令是否正确做判断，对于命令执行过程中产生的异常无法回滚，这正是redis事物的缺陷。

Redis事务不支持集群操作。

#### Redis 流水线（pipelined）

使用multi...exec事务命令是有系统开销的，因为它会检测对应的锁和序列化命令。有时候我们希望在没有任何附加条件的场景下去使用队列批量执行一系列命令，从而提高系统的性能，这就是流水线（pipelined）技术。

#### Redis 和 数据库结合会产生什么问题。

会产生数据不一致的情况，比如下面两种情况：

1. 对于Redis的事物，通过学习应该清楚它并不是很严格的，如果发生异常回滚事件，那么Redis的数据就可能和数据库不太一致。
2. 如下图：

