Spring快速入门

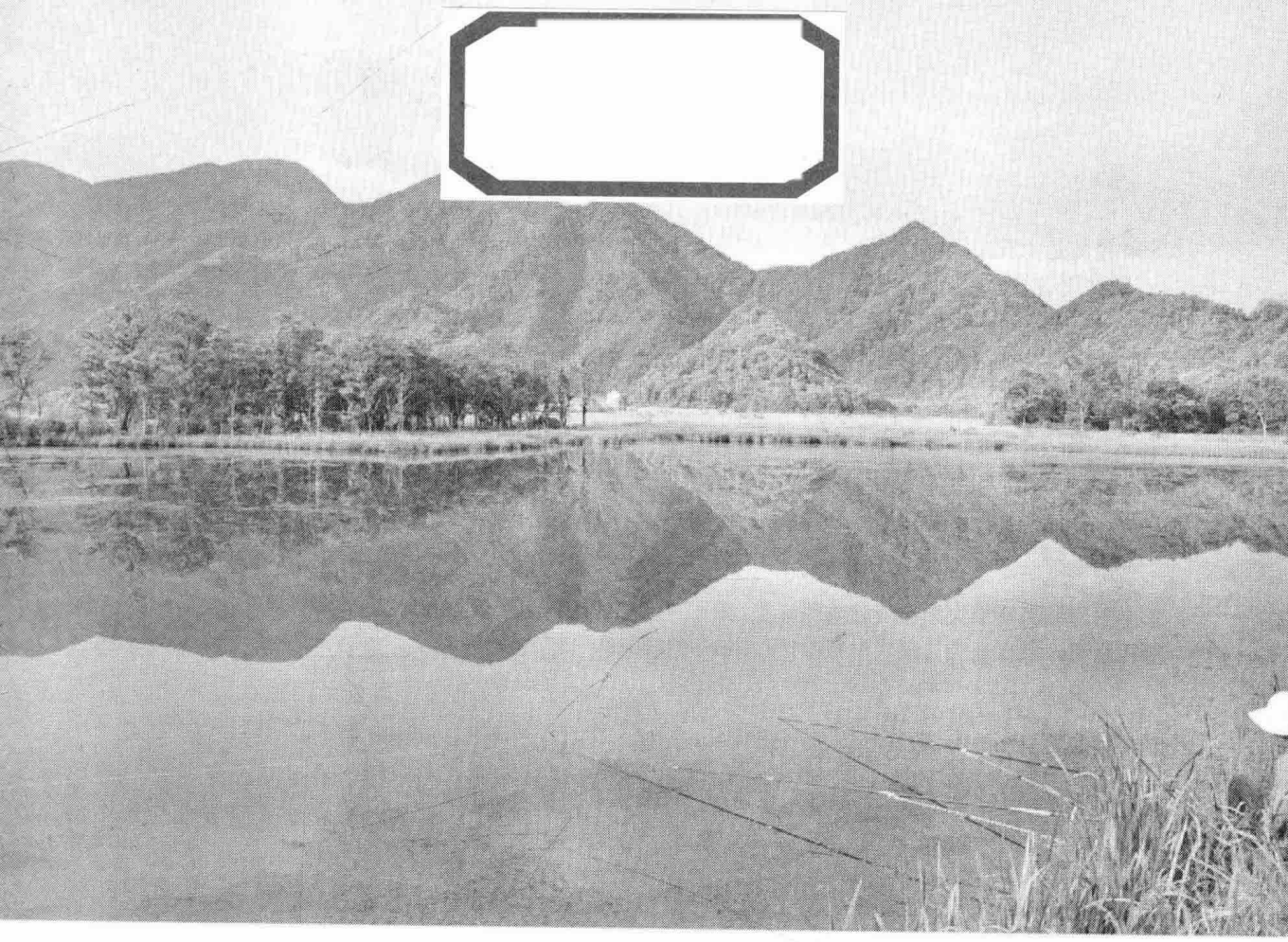
掌握Spring框架基础，快速形成Spring框架全局观

本书介绍Spring框架各个模块的使用，并集成Spring Boot、Spring MVC、MyBatis技术实现一个项目案例，让读者轻松快速上手Spring框架。

崔彦威　卢欣欣　王倩　著

本书示例源代码

清华大学出版社



Spring快速入门

崔彦威 卢欣欣 王倩 著

常州大学图书馆
藏书章

清華大學出版社
北京

内 容 简 介

SSM 目前是 J2EE 开发最常用、最流行的框架。本书将对 Spring、Spring MVC、MyBatis、Spring Boot、Docker 的使用进行介绍，每一章都有代码示例，便于理解每个章节的知识点，让读者掌握 SSM 框架，快速上手。

全书分为 11 章，内容包括 Spring 框架用到的注解、反射，Spring 框架基础，核心容器，AOP，DAO，ORM 与 MyBatis，Spring MVC，Spring Boot 配置，Spring Boot 应用，Docker 入门知识，最后实现一个用户权限管理的项目案例。

本书既适合 Spring 框架初学者学习（需要有 Java 编程基础），也适合高等院校和培训学校相关专业的师生参考。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目（CIP）数据

Spring 快速入门/崔彦威，卢欣欣，王倩著. —北京：清华大学出版社，2019
ISBN 978-7-302-53082-4

Ⅰ. ①S… Ⅱ. ①崔… ②卢… ③王… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2019）第 102171 号

责任编辑： 夏毓彦
封面设计： 王　翔
责任校对： 闫秀华
责任印制： 沈　露

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者： 北京鑫丰华彩印有限公司
装 订 者： 三河市溧源装订厂
经　　销： 全国新华书店
开　　本： 190mm×260mm　　**印　　张：** 18　　**字　　数：** 461 千字
版　　次： 2019 年 7 月第 1 版　　**印　　次：** 2019 年 7 月第 1 次印刷
定　　价： 69.00 元

产品编号：081773-01

此为试读，需要完整PDF请访问：www.ertongbook.com

前　言

平时我有写博客的习惯，喜欢将学到的知识点放在博客上：一是当作自己的学习笔记，将学习的内容整理之后再输出，也能够加深印象，忘记知识点时还可以快速复习；二是分享给有需要的朋友，希望各位在学习时能少走些弯路，少跳些坑。作为经常从网上索取免费资料的一员，我也要有回报的思想。

SSM 框架集目前是 J2EE 开发最常用、最流行的框架。Spring Boot 是由 Pivotal 团队提供的全新框架，设计目的是简化新 Spring 应用的初始搭建以及开发过程。Docker 容器技术在现在流行的 Devops 流水线上也扮演着重要的角色。

在本书中，我们将对 Spring、Spring MVC、MyBatis、Spring Boot、Docker 的使用进行介绍，而且每个章节基本都有代码示例，基本都是与技术相关、业务相关的，例子接近生活，便于读者对每个章节的知识点加深理解，快速上手。

本书读者对象

- 熟悉面向对象编程、经验丰富又打算学习 SSM、Spring Boot 的其他语言从业者。
- 有意提升网站和 Web 应用程序开发能力的 Web 开发人员。
- 希望在学习完 Java 编程想进一步提高开发技能的初学者。

阅读本书需要掌握 Java 面向对象编程知识，了解面向对象思想。

本书内容

本书共包括 11 章。第 1 章介绍 Java 基础，主要介绍 Spring 框架中常用的反射和注解技术，了解反射、注解相关概念。第 2 章先对 Spring 框架进行简单介绍，讲解 Spring 框架重要的 IOC、AOP 思想。第 3 章讲解 Spring 核心容器，介绍 Bean 的配置、注入方式、作用域和生命周期。第 4 章对 AOP 进行详细介绍，了解 AspectJ 的使用。第 5 章介绍 Spring 的 DAO 模块，同时了解 JDBC 的使用。第 6 章学习 MyBatis 的使用，主要包括 XML 的配置和映射，动态 SQL、逆向工程和 Pagehelper 的使用。第 7 章介绍 SSM 框架中的 SpringMVC，了解 Spring MVC 的处理流程、View 与 Controller 之间的数据传递。第 8、9 章主要介绍 Spring Boot 的相关知识以及 Spring Boot 配置，使用 Spring Boot 引入 Thymeleaf、JSP、MyBatis、Redis、Druid 等工具。第 10 章讲述 Docker 基础知识以及 Docker 的三大核心概念，并在 Docker 中使用 Tomcat 部署 war 包。第 11 章给出了一个项目实例，对前面章节介绍的知识点进行巩固。

本书导读

学习编程步骤可以用“学、练、悟、通”4 个字概括。

（1）“学”指的是接收的过程，侧重理论。本书每个章节基本都是先介绍理论知识，让读者理解知识点为什么出现、要解决什么问题、有哪些优势。

（2）“练”指的是实践的过程。没有实践只有理论属于纸上谈兵，看的时候理解，操作时无从下手，动手能力差。本书每个章节都有实例，在学习理论的过程中可以参考实例操作一遍。

（3）“悟”指的是思考的过程。练更多的是模仿，照葫芦画瓢。在练的过程中也要多思考，多问几个为什么，多归纳总结，在做项目之前可以先把整个思路在脑子里过一遍。

（4）“通”指的是举一反三的过程。实现本书的例子不难，难的是将学到的知识举一反三，灵活地运用到其他项目中。可以找一些开源项目来研究，以达到融会贯通的境界。

示例源码下载

本书基本每个章节都有示例，完整源码可以扫描右边二维码，如果下载有问题，请联系技术支持邮箱 cuiyw525@163.com，邮件主题为“Spring 快速入门”。要运行本书中的示例，需要安装 Eclipse、Maven，并配置相关环境。

勘误与技术支持邮箱

作者已尽最大努力确保正文和代码没有问题。可是，金无足赤，疏漏在所难免。如果书中有错误，希望您能及时反馈给我们。我们将诚挚接受广大读者的批评指正，交流邮箱为 cuiyw525@163.com。勘误将发布在作者博客上：https://www.cnblogs.com/5ishare/。

致谢

出书、创业卖胡辣汤、做 IT 讲师是我大学时的梦想。这本书的出版也算是实现了我的一个梦想，当然实现梦想离不开大家的帮助。首先要感谢夏毓彦编辑，是他发现了我，给了我机会，并给予我自始至终的指导，还为我介绍同行朋友。感谢我的两个兄弟，陈岩亮、袁伟，大学时候的“铁三角”，我们一起度过了美好的大学生活。感谢卢欣欣、王倩、崔春英、陈立勇等教过我的大学老师，正是他们的教导让我对计算机产生了兴趣。感谢身边的同事和卢艳霞同学，他们给了我坚持写下去的动力。感谢生我养我的父母，长大后才慢慢理解了他们的不易，理解了生活的不易。最后感谢所有帮助过我的人。

本书封面照片由蜂鸟网的摄影家 ptwkzj 先生友情提供，在此表示衷心感谢。

崔彦威

2019 年 5 月于深圳

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第 1 章

◂ Java基础 ▸

在学习 Spring 之前我们需要对 Java 基础语法有一定的了解，Java 中最重要的两个知识点是注解和反射。注解和反射在 Spring 框架中应用的最广泛。掌握注解和反射，有助于后面 Spring 的学习。

本章主要涉及的知识点：

- 注解基础：什么是注解?怎么理解注解?什么是元注解?
- 注解应用：自定义注解、注解的应用场景。
- 反射：反射的定义、反射的应用。

不管学习什么框架都需要先把 Java 基础夯实，基础打好之后才能厚积薄发。用到的时候不能只会用，不知道为什么这样用。学习编程还有最重要的一点就是需要勤动手，不能眼高手低，看着会做，真要动手时无从下手。

1.1 注解

本节首先介绍注解的基本概念，理解什么是注解、注解的作用是什么。在此基础上通过示例动手操作加深理解。

1.1.1 什么是注解

我们先看官方解释：它提供了一种安全的类似注释的机制，用来将任何的信息或元数据（metadata）与程序元素（类、方法、成员变量等）进行关联。为程序的元素（类、方法、成员变量）加上更直观、更明了的说明，这些说明信息与程序的业务逻辑无关，并且供指定的工具或框架使用。Annontation 像一种修饰符一样，应用于包、类型、构造方法、方法、成员变量、参数及本地变量的声明语句中。Java 注解是附加在代码中的一些元信息，便于一些工具在编译、运行时进行解析和使用，起到说明、配置的功能。注解不会也不能影响代码的实际逻辑，仅仅起到辅助性的作用，包含在 java.lang.annotation 包中。

看着上面的解释是不是还是一头雾水？其实我们可以更通俗地理解一下。最近几年出现一个词“斜杠青年”，还有黄某某拍摄的广告语：给人贴标签、下定义，总是很容易，而我却不

会因为一件事被定性。这里的斜杠青年、贴标签都是把某些属性附加给对象，和注解功能差不多，它提供了一种安全的类似注释的机制，用来将任何信息或元数据（metadata）与程序元素（类、方法、成员变量等）进行关联。我们可以再来理解一下这句话，这里的程序元素可以理解为人，信息或元数据理解为标签，把标签属性（信息或元数据）赋给人（程序元素）。

上面两段基本把什么注解解释出来了，如果还是不知道注解是什么，那也没关系。其实我们在编程中已经用到或者看到过了，比如@Override、@Deprecated。是不是很熟悉？其实它们就是注解。

1.1.2 内置注解

上面的@Override、@Deprecated 都是 Java 中内置的注解，除了这两个还有其他的内置注解。这里列举了几个常用的内置注解以及它们的作用。

- @Deprecated：编译器在编译阶段遇到这个注解时会发出提醒警告，告诉开发者正在调用一个过时的元素，比如过时的方法、过时的类、过时的成员变量。
- @Override：提示子类要复写父类中被@Override 修饰的方法。
- @SuppressWarnings：阻止警告的意思。调用被@Deprecated 注解的方法后，编译器会警告提醒，而有时候开发者会忽略这种警告，他们可以在调用的地方通过@SuppressWarnings：达到目的。
- @SafeVarargs：参数安全类型注解。它的目的是提醒开发者不要用参数做一些不安全的操作，它的存在会阻止编译器产生 unchecked 这样的警告。它是在 Java 1.7 的版本中加入的。
- @FunctionalInterface：函数式接口注解，这个是 Java 1.8 版本引入的新特性。函数式编程很火，所以 Java 8 也及时添加了这个特性。函数式接口（Functional Interface）就是一个具有一个方法的普通接口。

1.1.3 元注解

通过前面的两小节，我们应该对注解有了一定的认识，下面进一步地了解一下注解。我们在自定义注解时会出现图 1-1 所示的一些选项。

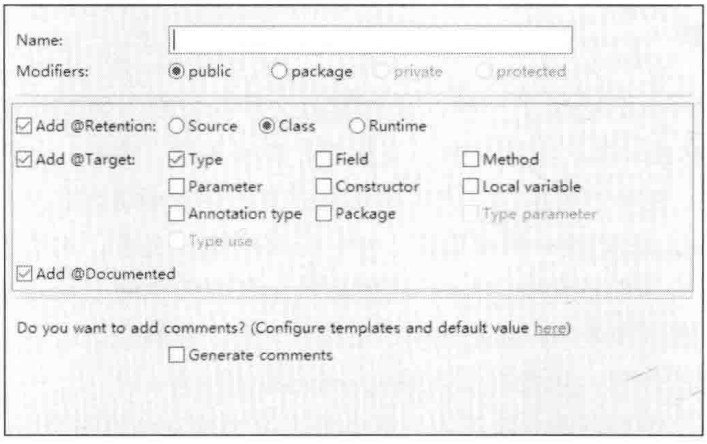


图 1-1

这些选项@Retention、@Target、@Documented 其实就是元注解。在创建时配置这些元注解，我们也可以推断出元注解的作用是什么。元注解负责注解自定义注解。java.lang.annotation 提供了 5 种元注解，专门注解其他的注解：

- @Retention：什么时候使用该注解。
- @Target：注解用于什么地方。
- @Documented：注解是否将包含在 JavaDoc 中。
- @Inherited：是否允许子类继承该注解。
- @Repeatable：指定注解可重复使用。

1. @Retention 定义注解的生命周期

- RetentionPolicy.SOURCE：在编译阶段丢弃。这些注解在编译结束之后不再有任何意义，所以它们不会写入字节码。@Override 和@SuppressWarnings 都属于这类注解。
- RetentionPolicy.CLASS：在类加载的时候丢弃。在字节码文件的处理中有用。注解默认使用这种方式。
- RetentionPolicy.RUNTIME：始终不会丢弃，运行期也保留该注解，因此可以使用反射机制读取该注解的信息。我们自定义的注解通常使用这种方式。

2. @Target 表示注解用于什么地方

默认值为任何元素，表示该注解用于什么地方。可用的 ElementType 参数包括：

- ElementType.CONSTRUCTOR：用于描述构造器。
- ElementType.FIELD：成员变量、对象、属性（包括 enum 实例）。
- ElementType.LOCAL_VARIABLE：用于描述局部变量。
- ElementType.METHOD：用于描述方法。
- ElementType.PACKAGE：用于描述包。
- ElementType.PARAMETER：用于描述参数。
- ElementType.TYPE：用于描述类、接口（包括注解类型）或 enum 声明。

3. @Documented 是一个简单的 Annotations 标记注解

表示是否将注解信息添加在 Java 文档中。

4. @Inherited 定义注解和子类的关系

@Inherited 元注解是一个标记注解，阐述了某个被标注的类型是被继承的。如果一个使用了@Inherited 修饰的 annotation 类型被用于一个 class，那么这个 annotation 将被用于该 class 的子类。

5. @Repeatable 指定注解可重复使用

使用@Repeatable 修饰表示该注解可以为重复使用。

1.1.4　自定义注解

元注解是负责注解自定义注解的。自定义注解时是有一些规则限制的，具体如下：

- Annotation 型定义为@interface，所有的 Annotation 会自动继承 java.lang.Annotation 这一接口，并且不能再去继承别的类或是接口。
- 参数成员只能用 public 或默认（default）这两个访问权修饰。
- 参数成员只能用基本类型 byte、short、char、int、long、float、double、boolean 八种基本数据类型和 String、Enum、Class、annotations 等数据类型，以及这一些类型的数组。
- 要获取类方法和字段的注解信息，必须通过 Java 的反射技术来获取 Annotation 对象，因为除此之外没有其他获取注解对象的方法。
- 注解也可以没有定义成员。

我们这里自定义一个注解来练习一下，主要用来演示自定义注解以及注解的继承。

1. 定义 CustomDescription 注解

CustomDescription 注解相当于标签。为了能多贴标签，又定义了注解容器 CustomDescriptions。其中，@Retention(RUNTIME)表示在运行时环境也可以获取注解，@Inherited 表示可继承，@Repeatable(CustomDescriptions.class)表示该注解可多次使用。

```
package CusAnnontation;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Documented;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import java.lang.annotation.Repeatable;

@Documented
@Retention(RUNTIME)
@Target(TYPE)
@Inherited
@Repeatable(CustomDescriptions.class)
public @interface CustomDescription {
   String description() default "";
}
```

CustomDescriptions 容器：

```
package CusAnnontation;

import static java.lang.annotation.ElementType.TYPE;
import static java.lang.annotation.RetentionPolicy.RUNTIME;
import java.lang.annotation.Documented;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
```

```
@Documented
@Retention(RUNTIME)
@Target(TYPE)
@Inherited
public @interface CustomDescriptions {
   CustomDescription[] value();
}
```

2. 实现继承关系

这里为了演示，我们创建了两个类：一个基类 Person，一个子类 Student。在 Person 类加两个自定义注解，在 Student 中加一个自定义注解。

Person：

```
package CusAnnontation;

@CustomDescription(description="基类")
@CustomDescription(description="人")
public class Person {

   private String Name;

   public String getName() {
       return Name;
   }

   public void setName(String name) {
       Name = name;
   }
}
```

Student：

```
package CusAnnontation;

@CustomDescription(description="学生")
public class Student extends Person {
   private String StudentId;

   public String getStudentId() {
       return StudentId;
   }

   public void setStudentId(String studentId) {
       StudentId = studentId;
   }
}
```

3. 通过反射获取注解属性值

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    CustomDescriptions  customDescriptions =new Student().getClass().
getAnnotation(CustomDescriptions.class);
    for(CustomDescription h: customDescriptions.value()){
        System.out.println("description:" + h.description());
    }
}
```

这里我们想通过反射（可以先不要理解）获取 Student 类的注解值，那么问题来了，它是输出什么的呢？会输出“description:学生”吗？并不是，而是输出父类 Person 的注解。

输出：

```
description:基类
description:人
```

如果想输出子类 Student 的注解该怎么设置呢?很简单，只需在子类 Student 上覆盖父类的注解就好。

```
@CustomDescription(description="学生")
@CustomDescription(description="人")
public class Student extends Person
```

输出：

```
description:学生
description:人
```

此时输出的就是子类的注解值了。这里我们还可以验证@Retention 生命周期的作用，只需要把@Retention(RUNTIME)改成 CLASS，再运行就会报错，因为 main 方法中的 custormDescriptions 对象是一个 null 空值。不过自定义注解一般来说都是使用@Retention(RUNTIME)。

1.1.5 注解使用场景介绍

在上一小节通过实例学习了自定义注解的使用，之后就该解决怎么用的问题了。其实，注解应用的场景还是挺多的。

（1）使用注解做 bean 的属性值校验，例如在开发 Java 服务器端代码时，会要求对外部传来的参数合法性进行验证。hibernate-validator 提供了一些常用的参数校验注解。

（2）使用注解做权限控制。例如，shiro 框架中有 5 个权限注解，我们也可以自定义注解进行权限控制。

（3）代替配置文件功能，像 Spring 基于注解的配置，减少了 xml 的配置。

（4）可以生成文档，像 Java 代码注释中的@see、@param 等。

此为试读，需要完整PDF请访问：www.ertongbook.com