Spring：Spring是一个开源框架，是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架，为了解决企业应用开发的复杂性而创建的。Spring的一个最大的目的就是使JAVA EE开发更加容易，Spring致力于提供一个以统一的、高效的方式构造整个应用，将单层框架以最佳的组合揉和在一起建立一个连贯的体系，Spring是一个提供了更完善开发环境的一个框架，可以为POJO(Plain Ordinary Java Object)对象提供企业级的服务。

## 优点

## 1.JAVA EE 更加容易使用

1. Spring将使用接口的复杂度降低到零
2. Spring框架会使代码的测试更加简单
3. 在Java中，已检查异常被过度使用，框架不迫使你捕获不能恢复的异常。

轻量——从大小与开销两方面而言Spring都是轻量的。完整的Spring框架可以在一个大小只有1MB多的JAR文件里发布。并且Spring所需的处理开销也是微不足道的。此外，Spring是非侵入式的：典型地，Spring应用中的对象不依赖于Spring的特定类。

控制反转——Spring通过一种称作控制反转（IoC）的技术促进了松耦合。当应用了IoC时，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。可以认为IoC与JNDI相反，不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

面向切面——Spring提供了[面向切面编程](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%88%87%E9%9D%A2%E7%BC%96%E7%A8%8B" \t "https://baike.baidu.com/item/spring%E6%A1%86%E6%9E%B6/_blank)的丰富支持，允许通过分离应用的业务逻辑与系统级服务（如审计和事务管理）进行内聚性的开发。应用对象只实现它们应该做的完成业务逻辑仅此。并不负责甚至是意识其它的系统级关注点，（如日志或事务支持）。

容器——Spring包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，你可以配置你的每个bean如何被创建基于一个可配置原型，bean可以创建一个单独的实例或者每次需要时都生成一个新的实例，以及它们是如何相互关联的，Spring不被混同于传统的重量级的EJB容器，因为经常是庞大与笨重的，难以使用。

框架——Spring可以将简单的组件配置、组合成为复杂的应用。在Spring中，应用对象被声明式地组合，典型地是在一个XML文件里。Spring也提供了很多基础功能（事务管理、持久化框架集成等等），将应用逻辑的开发留给了开发者。

所有Spring的这些特征使开发者能够编写更干净、更可管理、并且更易于测试的代码。也为Spring中的各种模块提供了基础

Spring框架由七个定义明确的模块组成

1. 核心容器
2. 应用上下文（Context）模块
3. Spring的AOP模块
4. 对象/关系映射集成模块
5. Spring的Web模块
6. [MVC框架](https://baike.baidu.com/item/MVC%E6%A1%86%E6%9E%B6/9241230" \t "https://baike.baidu.com/item/spring%E6%A1%86%E6%9E%B6/_blank)
7. Spring框架Web页面乱码问题

Sprngmvc：Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring 框架提供了构建 [Web](https://baike.baidu.com/item/Web/150564" \t "https://baike.baidu.com/item/spring%20MVC/_blank) 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用Spring进行WEB开发时，可以选择使用Spring的Spring MVC框架或集成其他MVC开发框架，如Struts1(一般不用)，Struts 2(一般老项目使用)等解决了V-C的交互问题，即视图与控制器的交互问题。

在原生的Java EE技术中，使用Servlet作为项目中的控制器，用于接收用户的请求，并给予响应结果。这种做法最大的问题在于：在常规做法中，每个Servlet对应1个请求路径，如LoginServlet处理login.com的请求，而RegisterServlet处理register.com的请求，所以，会导致Servlet数量太多，不便于管理（无论从源文件的数量，还是从配置文件的内容），且对象太多进而占用大量内存空间的问题。

### SpringMVC核心组件

1.DispatcherServlet

前端控制器，主要职责是接收所有请求（根据配置文件来决定），并将请求转发给对应的控制器，接收控制器的处理结果，确定最终由哪个视图完成响应

1. HandlerMapping

处理请求路径与控制器的映射关系

1. Controller

实际处理请求的组件，例如接收请求参数，决定最终是转发或重定向的方式来响应

1. ModelAndView

控制器的处理结果，其中的Model表示转发的数据（如果是重定向，则Model没有意义），而View表示最终负责响应的视图组件的名称。

1. ViewResolver

根据视图组件的名称，确定具体使用的是哪个视图组件

在SpringMVC中，请求对应的可以是某个方法，而不一定每个请求对应一个控制器类，如果请求对应的是以上方法，则需要在方法的声明之前添加@RequestMapping("/hello.com")完成后，即：路径为/hello.com的请求将触发showHello()方法被执行。

Mybats：MyBatis 支持普通 SQL查询，[存储过程](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B" \t "https://baike.baidu.com/item/MyBatis/_blank)和高级映射的优秀[持久层](https://baike.baidu.com/item/%E6%8C%81%E4%B9%85%E5%B1%82" \t "https://baike.baidu.com/item/MyBatis/_blank)框架，它支持定制化 SQL、存储过程及高级映射。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Ordinary Java Object,普通的 Java对象)映射成数据库中的记录。

Mybatis的功能架构分为三层：（1）API接口层：提供给外部使用的接口API，开发人员通过这些本地API来操纵数据库。接口层接收到调用请求就会调用数据处理层来完成具体的数据处理。

(2)数据处理层：负责具体的SQL查找、SQL解析、SQL执行和执行结果映射处理等。主要的目的是根据调用的请求完成一次数据库操作。

(3)基础支撑层：负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为最基础的组件。为上层的数据处理层提供最基础的支撑。

特点：

简单易学：本身就很小且简单，没有任何第三方依赖，最简单安装只要两个jar文件+配置几个sql映射文件易于学习，易于使用，通过文档和源代码，可以比较完全的掌握它的设计思路和实现。

灵活：mybatis不会对应用程序或者数据库的现有设计强加任何影响，sql写在xml里，便于统一管理和优化，通过sql语句可以满足操作数据库的所有需求。

解除sql与程序代码的耦合：通过提供DAO层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元测试。sql和代码的分离，提高了可维护性。

提供映射标签，支持对象与数据库的orm字段关系映射，提供对象关系映射标签，支持对象关系组建维护。

提供xml标签，支持编写动态sql。

1. 集中式vs分布式

笔者之前使用过非常多的版本控制工具，主要有CVS（09年左右）、VSS（微软的Visual Source Safe）、TFS(微软的Team Foundation Server)、SVN（Subversion）。这些都是集中式的版本控制系统，而TFS功能是最强大的，可以讲它已经到了极致。

什么是集中式？

版本库是集中存放在中央服务器的，而干活的时候，用的都是自己的电脑，所以要先从中央服务器取得最新的版本，然后开始干活，干完活了，再把自己的活推送给中央服务器。集中式版本控制系统最大的毛病就是必须联网才能工作，如果在局域网内还好，带宽够大，速度够快，可如果在互联网上，遇到网速慢的话，可能提交一个10M的文件就需要5分钟，这还不得把人给憋死啊。

什么是分布式？

首先，分布式版本控制系统根本没有“中央服务器”，每个人的电脑上都是一个完整的版本库，这样，你工作的时候，就不需要联网了，因为版本库就在你自己的电脑上。分布式版本控制系统的安全性要高很多，因为每个人电脑里都有完整的版本库，某一个人的电脑坏掉了不要紧，随便从其他人那里复制一个就可以了。

那多个人如何协作呢？比方说你在自己电脑上改了文件A，你的同事也在他的电脑上改了文件A，这时，你们俩之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

如果你们俩不在一个局域网内，两台电脑互相访问不了，也可能今天你的同事病了，他的电脑压根没有开机？在实际使用分布式版本控制系统的时候，其实很少在两人之间的电脑上推送版本库的修改。因此，分布式版本控制系统通常也有一台充当“中央服务器”的电脑，但这个服务器的作用仅仅是用来方便“交换”大家的修改，没有它大家也一样干活，只是交换修改不方便而已。

分布式版本很多，Git（免费的）的优势不单是不必联网这么简单，后面我们还会看到Git极其强大的分支管理。

2. Git的诞生

Linus Torvalds是大牛中的大牛，它不但开创了Linux操作系统，而且他发明了Git版本控制系统。

在2002年以前，世界各地的志愿者把源代码文件通过diff的方式发给Linus，然后由Linus本人通过手工方式合并代码。2005年Linux 内核开发社区正面临严峻的挑战：他们不能继续使用 BitKeeper了（一个分布式版本控制系统），原因是当时Bitkeeper 著作权所有者决定收回授权，内核开发团队与其协商无果，而又没有其他的 SCM （Software Configuration Management）可满足他们的分布式系统的需求。

为什么Linus不把Linux代码放到版本控制系统里呢？不是有CVS、SVN这些免费的版本控制系统吗？因为Linus坚定地反对CVS和SVN，这些集中式的版本控制系统不但速度慢，而且必须联网才能使用。

Linus花了两周时间自己用C写了一个分布式版本控制系统，这就是Git！一个月之内，Linux系统的源码已经由Git管理了！牛是怎么定义的呢？大家可以体会一下。Git迅速成为最流行的分布式版本控制系统，尤其是2008年，GitHub网站上线了，它为开源项目免费提供Git存储，无数开源项目开始迁移至GitHub，包括jQuery，PHP，Ruby等等。

Atlassian为Git做了一个10年的纪录图。

3. git github gitlab区别

git 它是一种开源的版本控制系统，可以高效的管理项目版本。同时也是一个版本控制软件。在Git中并不存在主库这样的概念，每一份复制出的库都可以独立使用，任何两个库之间的不一致之处都可以进行合并。

github 它可以托管各种Git库，并提供web界面。GitHub的独特卖点在于从另外一个项目进行分支的简易性。为一个项目贡献代码非常简单：首先点击项目站点的“fork”的按钮，然后将代码检出并将修改加入到刚才分出的代码库中，最后通过内建的“pull request”机制向项目负责人申请代码合并。已经有人将GitHub称为代码玩家的MySpace。

gitlab 它是一个用于仓库管理系统的开源项目，使用git进行代码管理工具，可以自己搭建GitLab仓库服务器。

4. 安装

最早Git是在Linux上开发的，很长一段时间内，Git也只能在Linux和Unix系统上跑。不过，慢慢地有人把它移植到了Windows上。现在，Git可以在Linux、Unix、Mac和Windows这几大平台上正常运行了。安装比较简单，这里不解释了，看官方介绍吧

5. 命令操作

5.1 远程仓库操作

检出仓库 $git clone git://github.com/jquery/jquery.git

查看远程仓库

$git remote –v

$git remote show origin

添加远程仓库 $git remote add [name] [url]

删除远程仓库

$git remote rm [name]

修改远程仓库

$git remote set-url --push[name][newUrl]

拉取远程仓库

$git pull [remoteName] [localBranchName]

推送远程仓库

$git push [remoteName] [localBranchName]

$git push origin [name] 创建远程分支(本地分支push到远程)

5.2 分支操作

git branch #列出本地分支

git branch -a #列出local and remote branch

git branch -r #列出remote branch

git branch feture1 #新建一个分支feture1

git branch -d feture1 #删除分支feture1，d会检查feture1是否已经merge到其父分支，如果没有，不予删除。D直接删除

git checkout feture1 #切换到分支feture1

git checkout -b feture1 #创建新分支并立即切换到新分支

git merge feture1 #合并并提交feture1的所有版本到当前分支，如果当前分支自建立feture1后无更改，则启动fastforward，可以添加参数–no-ff关闭该效果

git status #查看目前分支下文件控制情况

git commit -v #可以查看diff内容

git log -1 #查看log信息,最近1条

git log --pretty=oneline #展示提交历史(用oneline 将每个提交放在一行显示)

git log --pretty=format:"%h - %an, %ar : %s" #可以定制要显示的记录格式

5.3 版本操作

查看版本 $git tag

创建版本

$git tag [name]

删除版本

$git tag -d [name]

查看远程版本 $git tag -r

创建远程版本(本地版本push到远程) $git push origin [name]

删除远程版本 $git push origin :refs/tags/[name]

5.4 submodule操作

git submodule add [url] [path]

添加子模块，如：

$git submodule add git://github.com/soberh/ui-libs.git src/main/webapp/ui-libs

git submodule init

初始化子模块，只在首次检出仓库时运行一次就行

git submodule update

更新子模块，每次更新或切换分支后都需要运行一下

git rm

删除子模块，

1)$ git rm --cached [path]

2) 编辑“.gitmodules”文件，将子模块的相关配置节点删除掉

3) 编辑“.git/config”文件，将子模块的相关配置节点删除掉

4) 手动删除子模块残留的目录

5.5 忽略

在仓库根目录下创建名称为“.gitignore”的文件，写入不需要的文件夹名或文件，每个元素占一行即可。可以忽略自身。.gitignore文件本身要放到版本库里，并且可以对.gitignore做版本管理！

.a #忽略.a的文件

!lib.a #不要忽略lib.a

/TODO #忽略此目录下的TODO文件，子目录下的TODO文件不忽略

build/ #忽略所有build/目录下的文件

doc/.txt #忽略文件入doc/notes.txt，但是子目录下的txt不被忽略

5.5 git config

因为Git是分布式版本控制系统，所以需要填写用户名和邮箱作为一个标识

$git config --global user.name "alex2chen"

$git config --global user.email "471502979@qq.com"

注意：git config –global 参数，有了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置，当然你也可以对某个仓库指定的不同的用户名和邮箱。

5.6 创建版本库

$cd D:

$mkdir git-test

$cd git-test

$pwd

$git init #把这个目录变成git可以管理的仓库,会多了一个.git的目录，是Git来跟踪管理版本的，没事千万不要手动乱改这个目录里面的文件，否则，会把git仓库给破坏了

$git add readme.txt #添加到暂存区

$git commit -m "readme.txt" #把文件提交到仓库

$git status #查看是否还有文件未提交

$git diff readme.txt #readme.txt文件已被修改，但是未被提交的修改，可以查看差异性

$git log #查看修改历史

$git log --pretty=oneline #查看简单的修改历史

$git reset --hard HEAD^ #回退到上一个版本的命令

我们前面说过使用Git提交文件到版本库有两步：

是使用 git add 把文件添加进去，实际上就是把文件添加到暂存区

使用git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支上

6. TortoiseGit 可视化

TortoiseGit是一个开放的git版本控制系统的源客户端，支持Winxp/vista/win7/Win10，下载地址。

7. github演示

or create a new repository on the command line

echo "# osgi\_javaModel\_t1" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/alex2chen/osgi\_javaModel\_t1.git

git push -u origin master

…or push an existing repository from the command line

git remote add origin https://github.com/alex2chen/osgi\_javaModel\_t1.git

git push -u origin master

…or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

版本库(Repository)：工作区有一个隐藏目录.git,这个不属于工作区，这是版本库。其中版本库里面存了很多东西，其中最重要的就是stage(暂存区)，还有Git为我们自动创建了第一个分支master,以及指向master的一个指针HEAD。