# Git学习

### Windows上安装git

运行exe文件，采取默认安装方式，安装完成后启用git bash设置如下：

git config --global user.name "sigoff"

git config --global user.email "1131270457@qq.com"

### 免密钥配置

1、生成公钥和密钥：ssh-keygen -t rsa -C "1131270457@qq.com"

2、在电脑账户的目录下生成.ssh文件夹，其中包含id\_rsa（密钥）和id\_ras.pub（公钥）文件，并将公钥添加在github账户上

3、查看是否配置成功：ssh -T git@github.com

### eclipse中git的使用

在eclipse中，通过git将项目提交到github，熟悉了eclipse中git的使用流程，并将jeeplus框架源码上传到GitHub的仓库中。

### 使用git命令提交文件

1、通过git clone在GitHub中仓库中的项目复制下来

2、通过git status查看修改或新增的文件

3、用git add将修改的文件提交到暂存区

4、用git commit -m "提交说明"将暂存区中的修改文件提交到分支

5、用git push origin master将本地仓库推送到远处仓库，也就是GitHub上。

### git常用命令

1、本地仓库的创建：git init

2、查看仓库状态：git status，查看修改的内容：git diff 文件名

3、将文件添加到暂存区：git add 文件名，可以以多个文件同时添加

4、将暂存区内容提交到仓库：git commit -m "提交说明"

5、版本回退：

git log/git log --pretty=oneline用于显示从最近到最远的提交日志，

git reset --hard commit\_id用于回退到之前的版本，

git relog可恢复到最新版本

6、推送到远程仓库：

git remote add origin git@github.com:sigoff-pd/notes.git用于关联远程仓库

git push origin master用于将本地仓库推送到远程仓库

7、获取资源并整合： git pull 远程主机名 远程分支名:本地分支名

# Linux常用命令学习

### cd命令

cd/cd ~：进入用户主目录

cd - 返回进入此目录之前所在的目录

cd .. 返回上级目录

cd ../.. 返回上两级目录

### mkdir命令：新建文件夹

mkdir dirname：直接跟文件夹名，可在当前目录下创建文件夹

mkdir /opt/lamp/dirname：可跟路径

mkdir -p /opt/lam/dirname：假如lam不存在，需要用-p才可以创建该文件夹

### touch命令：新建文件

touch dilename:直接跟文件名在当前目录下创建新的文件

### cat命令：文件常看

cat -n或-number filename：有1开始对所有输出的行数编号

cat -b或--number-nonblank filename：和-n相似，只不过对于空白行不编号

cat -s或--squeeze-blank filename：当遇到有连续两行以上的空白行，就代换为一行的空白行

cat -A filename：显示不可打印字符，行尾显示“$”

### du命令：查看文件大小的

du -sh \*：显示当前目录下所有文件的大小

du -sh filename：显示该文件大小

du -sh：显示当前目录所占空间大小

### mv命令：移动文件夹或文件

mv：类似于windows下的剪切

mv aaa /etc/udev：将aaa移动至/etc/udev目录下

mv /opt/lampp /etc/udev：将/opt目录下的lampp移动至/etc/udev目录下

mv -r aaa /etc/udev：将aaa文件夹递归移动至/etc/udev目录下，不加-r会出错

mv aaa bbb：将aaa改名为bbb

-b：当文件存在时，覆盖前，为其创建一个备份；

-f：若目标文件或目录与现有的文件或目录重复，则直接覆盖现有的文件或目录；

-i：交互式操作，覆盖前先行询问用户，如果源文件与目标文件或目标目录中的文件同名，则询问用户是否覆盖目标文件。

-u：当源文件比目标文件新或者目标文件不存在时，才执行移动操作。

### cp命令：复制文件夹或文件

cp /opt/lampp /etc/udev：用法与mv基本相似

-f：强行复制文件或目录，不论目标文件或目录是否已存在；

-i：覆盖既有文件之前先询问用户；

-s：对源文件建立符号连接，而非复制文件；

-R/r：递归处理，将指定目录下的所有文件与子目录一并处理

### rm命令：删除文件或文件夹

rm -rf filename/dirname：强制删除文件和文件夹

-r 递归删除

-f 强制删除，不询问

### >命令：重定向

cat aaa.txt > bbb.txt：将aaa的内容写入bbb中，覆盖写入

cat aaa.txt >> bbb.txt：将aaa的内容追加写入bbb中，不覆盖原来内容

> bbb.txt：将bbb清空

### chmod命令：权限管理

chmod -R：给文件夹下所有的文件赋权限，递归处理

chmod u+x,g+w filename：为文件设置自己可以执行，组员可以写入的权限

chmod u=rwx,g=rw,o=r fiename：给所属用户添加读写执行权限，给组添加读写权限，给其他用户添加读权限

chmod 764 filename：以数字的方式赋予所属用户/用户组/其他用户权限r=4，w=2，x=1

chmod a+x filename：对文件的u,g,o都设置可执行属性，a代表all

find命令：查找

find -name filename：查找filename所在目录

find -name name\*：查找开头为name的文件所在目录

find -name \*name：查找结尾为name的文件所在目录

find -name \*name\*：查找包含name字符串的文件所在目录

find -user username：查看用户的文件

### ****gzip/guzip tar命令：压缩和解压****

gzip test.txt：压缩文件

gzip -r test：压缩所有test下的子文件

gunzip test.gz：解压文件

tar -zcvf test.tar test：将test压缩为test.tar并输出详细信息

tar -zxvf test.tar：将test.tar解压缩，并输出详细信息

-c 创建新文档，就是代表压缩的意思

-x 解压文档

-f 使用归档文件

-z 使用gzip解压

-v 详细输出模式

### ****rpm/yum/源码安装：安装程序****

rpm -ivh mysql.rpm 安装mysql并显示安装进度和安装信息

rpm -e mysql.rpm 卸载mysql，只适用于rpm包安装的软件

rpm -e mysql.rpm --nodeps　　　　　 --nodeps可以忽略依赖关系强制卸载

rpm -ivh --nodeps mysql.rpm --nodeps可以忽略依赖关系强制安装

rpm -ivh --replacefiles mysql.rpm 当系统内已存在相同安装包时使用--replacefiles可替换已存在的安装包

-i 安装

-U 更新（很少用）

-e 卸载

-v 显示安装信息

-h 显示安装进度

yum install mysql 安装mysql

yum -y install mysql -y参数表示在安装过程中所有需要询问的选项都选yes

yum list|grep mysql 显示可安装的mysql列表

yum update mysqld 更新mysql

yum list installed|grep mysql 查询是否安装了mysql

yum remove mysql 卸载yum安装的mysql

1、系统环境变量文件在/etc下的profile文件

2、添加变量：export PATH=$PATH:/opt/lampp/bin /opt/lampp/bin为lampp的bin目录

3、source /etc/profile

### systemctl命令

查看全部服务命令：systemctl list-unit-files --type service

查看服务：systemctl status name.service

启动服务：systemctl start name.service

停止服务：systemctl stop name.service

重启服务：systemctl restart name.service

增加开机启动：systemctl enable name.service

删除开机启动：systemctl disable name.service

### tomcat部署

1、tomcat增加启动参数，需要增加一个pid文件

在tomca/bin 目录下面，增加 setenv.sh 配置，catalina.sh启动的时候会调用，同时配置Java内存参数。

#add tomcat pid

CATALINA\_PID="$CATALINA\_BASE/tomcat.pid"

#add java opts

JAVA\_OPTS="-server -XX:PermSize=256M -XX:MaxPermSize=1024m -Xms512M -Xmx1024M -XX:MaxNewSize=256m"

2、增加tomcat.service

在/usr/lib/systemd/system目录下增加tomcat.service，目录必须是绝对目录。

[Unit]

Description=Tomcat

After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]

Type=forking

PIDFile=/data/tomcat/tomcat.pid

ExecStart=/data/tomcat/bin/startup.sh

ExecReload=/bin/kill -s HUP $MAINPID

ExecStop=/bin/kill -s QUIT $MAINPID

PrivateTmp=true

[Install]

WantedBy=multi-user.target

[unit]配置了服务的描述，规定了在network启动之后执行

[service]配置服务的pid，服务的启动，停止，重启

[install]配置了使用用户

3、使用tomcat.service

配置开机启动：systemctl enable tomcat

启动tomcat：systemctl start tomcat

停止tomcat：systemctl stop tomcat

重启tomcat：systemctl restart tomcat

因为配置pid，在启动的时候会再tomcat根目录生成tomcat.pid文件，停止之后删除。

同时tomcat在启动时候，执行start不会启动两个tomcat，保证始终只有一个tomcat服务在运行，多个tomcat可以配置在多个目录下，互不影响。

# bootstrap学习

## bootstrap需导入文件

<!-- 新 Bootstrap 核心 CSS 文件 -->

<link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">

<!-- 可选的Bootstrap主题文件（一般不使用） -->

<script src="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"></script>

<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->

<script src="https://cdn.bootcss.com/jquery/2.1.1/jquery.min.js"></script>

<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->

<script src="https://cdn.bootcss.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

## bootstrap css

### 栅格系统

1、栅格的基本结构：

<div class="container">

<div class="row">

<div class="col-\*-\*"></div>

<div class="col-\*-\*"></div>

</div>

<div class="row">...</div>

</div>

2、偏移列

.col-md-offset-\* 类会把一个列的左外边距（margin）增加 \* 列，其中 \* 范围是从 1 到 11

3、嵌套列

为了在内容中嵌套默认的网格，请添加一个新的 .row，并在一个已有的 .col-md-\* 列内添加一组 .col-md-\* 列。被嵌套的行应包含一组列，这组列个数不能超过12（其实，没有要求你必须占满12列）

4、列排序

.col-md-push-\* 和 .col-md-pull-\* 类的内置网格列的顺序，其中 \* 范围是从 1 到 11

### 排版

1、标题

h1-h6标签，或.h1-.h6

2、内联子标题

small标签，或.small

3、引导主体副本

为了给段落添加强调文本，则可以添加 class="lead"，这将得到更大更粗、行高更高的文本

### 表单

1、垂直或基本表单

向父 <form> 元素添加 role="form"。

把标签和控件放在一个带有 class .form-group 的 <div> 中。这是获取最佳间距所必需的。

向所有的文本元素 <input>、<textarea> 和 <select> 添加 class ="form-control" 。

2、内联表单

向 <form> 标签添加 class .form-inline

默认情况下，Bootstrap 中的 input、select 和 textarea 有 100% 宽度。在使用内联表单时，您需要在表单控件上设置一个宽度

使用 class .sr-only，您可以隐藏内联表单的标签

3、水平表单

向父 <form> 元素添加 class .form-horizontal。

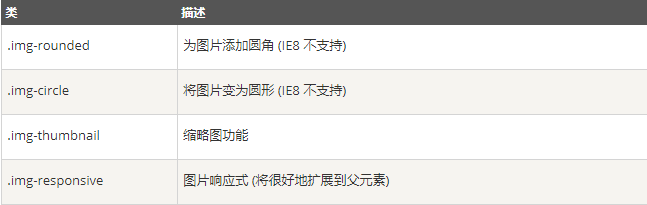
把标签和控件放在一个带有 class .form-group 的 <div> 中。

向标签添加 class .control-label。

### 按钮



### 图片



## bootstrap组件

### 字体图标

1、Bootstrap 图标：http://www.runoob.com/try/demo\_source/bootstrap3-glyph-icons.htm

2、定制图标：http://www.runoob.com/try/demo\_source/bootstrap-glyph-customization.htm

### 下拉菜单



### 按钮组

1、基本按钮组：class .btn-group

2、按钮工具栏：class .btn-toolbar

3、按钮大小：class .btn-group-\*

4、嵌套：在一个 .btn-group 内嵌套另一个 .btn-group

5、垂直按钮组：class .btn-group-vertical

### 按钮下拉菜单

<div class="btn-group">

<button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown"> 默认 <span class="caret"></span>

</button>

<ul class="dropdown-menu" role="menu">

<li> <a href="#">功能</a> </li>

<li> <a href="#">另一个功能</a> </li>

</ul>

</div>

分割的按钮下拉菜单

分割的按钮下拉菜单使用与下拉菜单按钮大致相同的样式，但是对下拉菜单添加了原始的功能。分割按钮的左边是原始的功能，右边是显示下拉菜单的切换。

按钮下拉菜单的大小

使用带有各种大小按钮的下拉菜单：.btn-large、.btn-sm 或 .btn-xs。

按钮上拉菜单

菜单也可以往上拉伸的，只需要简单地向父 .btn-group 容器添加 .dropup 即可。

### 输入框组

1、默认

把前缀或后缀元素放在一个带有 class .input-group 的 <div> 中。

接着，在相同的 <div> 内，在 class 为 .input-group-addon 的 <span> 内放置额外的内容。

把该 <span> 放置在 <input> 元素的前面或者后面。

2、输入框组的大小

通过向 .input-group 添加相对表单大小的 class（比如 .input-group-lg、input-group-sm）来改变输入框组的大小。输入框中的内容会自动调整大小。

3、复选框和单选插件

把复选框和单选插件作为输入框组的前缀或者后缀元素

4、按钮插件

把按钮作为输入框组的前缀或者后缀元素，这个时候不是添加 .input-group-addon class，需要使用 class .input-group-btn 来包裹按钮。

5、带有下拉菜单的按钮

在输入框组中添加带有下拉菜单的按钮，只需要简单地在一个 .input-group-btn class 中包裹按钮和下拉菜单即可

6、分割的下拉菜单按钮

在输入框组中添加带有下拉菜单的分割按钮，使用与下拉菜单按钮大致相同的样式，但是对下拉菜单添加了主要的功能

### 导航元素

1、表格导航或标签

创建一个标签式的导航菜单：

以一个带有 class .nav 的无序列表开始。

添加 class .nav-tabs。

2、胶囊式的导航菜单

基本的胶囊式导航菜单

如果需要把标签改成胶囊的样式，只需要使用 class .nav-pills 代替 .nav-tabs 即可

垂直的胶囊式导航菜单

使用 class .nav、.nav-pills 的同时使用 class .nav-stacked，让胶囊垂直堆叠

3、两端对齐的导航

您可以在屏幕宽度大于 768px 时，通过在分别使用 .nav、.nav-tabs 或 .nav、.nav-pills 的同时使用 class .nav-justified，让标签式或胶囊式导航菜单与父元素等宽。

4、禁用链接

对每个 .nav class，如果添加了 .disabled class，则会创建一个灰色的链接，同时禁用了该链接的 :hover 状态



### 导航栏

1、默认的导航栏

创建一个默认的导航栏的步骤如下：

向 <nav> 标签添加 class .navbar、.navbar-default。

向上面的元素添加 role="navigation"，有助于增加可访问性。

向 <div> 元素添加一个标题 class .navbar-header，内部包含了带有 class navbar-brand 的 <a> 元素。这会让文本看起来更大一号。

为了向导航栏添加链接，只需要简单地添加带有 class .nav、.navbar-nav 的无序列表即可。

### 面包屑导航

Bootstrap 中的面包屑导航（Breadcrumbs）是一个简单的带有 .breadcrumb class 的无序列表。

### 分页





### 标签

使用 class .label 来显示标签



### 徽章（Badges）

徽章（Badges）主要用于突出显示新的或未读的项。如需使用徽章，只需要把 <span class="badge"> 添加到链接、Bootstrap 导航等这些元素上即可。

### 超大屏幕（Jumbotron）

使用超大屏幕（Jumbotron）的步骤如下：

创建一个带有 class .jumbotron. 的容器 <div>。

除了更大的 <h1>，字体粗细 font-weight 被减为 200。

### 页面标题（Page Header）

如需使用页面标题（Page Header），请把您的标题放置在带有 class .page-header 的 <div> 中

### 缩略图

使用 Bootstrap 创建缩略图的步骤如下：

在图像周围添加带有 class .thumbnail 的 <a> 标签。

这会添加四个像素的内边距（padding）和一个灰色的边框。

当鼠标悬停在图像上时，会动画显示出图像的轮廓。

### 警告（Alerts）

1、默认

通过创建一个 <div>，并向其添加一个 .alert class 和四个上下文 class（即 .alert-success、.alert-info、.alert-warning、.alert-danger）之一，来添加一个基本的警告框。

2、可取消的警告（Dismissal Alerts）

创建一个可取消的警告（Dismissal Alert）步骤如下：

通过创建一个 <div>，并向其添加一个 .alert class 和四个上下文 class（即 .alert-success、.alert-info、.alert-warning、.alert-danger）之一，来添加一个基本的警告框。

同时向上面的 <div> class 添加可选的 .alert-dismissable。

添加一个关闭按钮。确保使用带有 data-dismiss="alert" data 属性的 <button> 元素。

3、警告（Alerts）中的链接

在警告（Alerts）中创建链接的步骤如下：

通过创建一个 <div>，并向其添加一个 .alert class 和四个上下文 class（即 .alert-success、.alert-info、.alert-warning、.alert-danger）之一，来添加一个基本的警告框。

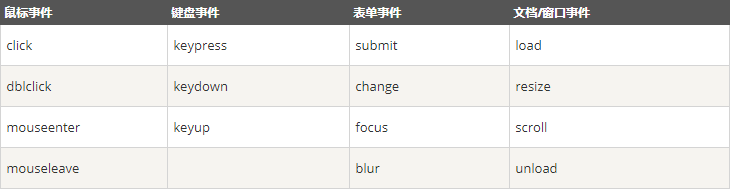
使用 .alert-link 实体类来快速提供带有匹配颜色的链接。

# jQuery学习

### 选择器



### 事件



## 效果

### 显示/隐藏

$(selector).hide(speed,callback);

$(selector).show(speed,callback);

$(selector).toggle(speed,callback);

### 淡入淡出

$(selector).fadeIn(speed,callback);

$(selector).fadeOut(speed,callback);

$(selector).fadeToggle(speed,callback);

$(selector).fadeTo(speed,opacity,callback); 给定透明度opacity

### 滑动

$(selector).slideDown(speed,callback);

$(selector).slideUp(speed,callback);

$(selector).slideToggle(speed,callback);

### 动画

$(selector).animate({params},speed,callback);

必需的 params 参数定义形成动画的 CSS 属性。

可选的 speed 参数规定效果的时长。它可以取以下值："slow"、"fast" 或毫秒。

可选的 callback 参数是动画完成后所执行的函数名称。

### 停止动画

$(selector).stop(stopAll,goToEnd);

可选的 stopAll 参数规定是否应该清除动画队列。默认是 false，即仅停止活动的动画，允许任何排入队列的动画向后执行。

可选的 goToEnd 参数规定是否立即完成当前动画。默认是 false。

因此，默认地，stop() 会清除在被选元素上指定的当前动画。

## HTML

### 设置、获取内容和属性

text() - 设置或返回所选元素的文本内容

html() - 设置或返回所选元素的内容（包括 HTML 标记）

val() - 设置或返回表单字段的值

attr() 方法用于获取属性值。

### 添加元素

append() - 在被选元素的结尾插入内容

prepend() - 在被选元素的开头插入内容

after() - 在被选元素之后插入内容

before() - 在被选元素之前插入内容

### 删除元素

remove() - 删除被选元素（及其子元素）

empty() - 从被选元素中删除子元素

### 获取并设置css类

addClass() - 向被选元素添加一个或多个类

removeClass() - 从被选元素删除一个或多个类

toggleClass() - 对被选元素进行添加/删除类的切换操作

css() - 设置或返回样式属性

### 尺寸

width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。

height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。

innerWidth() 方法返回元素的宽度（包括内边距）。

innerHeight() 方法返回元素的高度（包括内边距）。

outerWidth() 方法返回元素的宽度（包括内边距和边框）。

outerHeight() 方法返回元素的高度（包括内边距和边框）。

### 遍历（dom树）

### 祖先

parent() 方法返回被选元素的直接父元素。

parents() 方法返回被选元素的所有祖先元素，它一路向上直到文档的根元素 (<html>)。

parentsUntil() 方法返回介于两个给定元素之间的所有祖先元素。

### 同胞(siblings)

siblings() 方法返回被选元素的所有同胞元素。

### 过滤

first() 方法返回被选元素的首个元素。

last() 方法返回被选元素的最后一个元素。

eq() 方法返回被选元素中带有指定索引号的元素。

filter() 方法允许您规定一个标准。不匹配这个标准的元素会被从集合中删除，匹配的元素会被返回。

not() 方法返回不匹配标准的所有元素。

## ajax

### AJAX load() 方法

$(selector).load(URL,data,callback);

必需的 URL 参数规定您希望加载的 URL。

可选的 data 参数规定与请求一同发送的查询字符串键/值对集合。

可选的 callback 参数是 load() 方法完成后所执行的函数名称。

可选的 callback 参数规定当 load() 方法完成后所要允许的回调函数。回调函数可以设置不同的参数：

responseTxt - 包含调用成功时的结果内容

statusTXT - 包含调用的状态

xhr - 包含 XMLHttpRequest 对象

### AJAX get() 和 post() 方法

$.get(URL,callback);

必需的 URL 参数规定您希望请求的 URL。

可选的 callback 参数是请求成功后所执行的函数名。

callback参数

第一个回调参数data存有被请求页面的内容，第二个回调参数status存有请求的状态。

$.post(URL,data,callback);

必需的 URL 参数规定您希望请求的 URL。

可选的 data 参数规定连同请求发送的数据。

可选的 callback 参数是请求成功后所执行的函数名。

callback参数

第一个回调参数data存有被请求页面的内容，第二个回调参数status存有请求的状态。

# Apache Shiro学习

什么是Shiro?

Apache Shiro是一个功能强大且易于使用的Java安全框架，提供了认证，授权，加密，和会话管理  
Shiro组件

1、Subject ：当前用户的操作

2、SecurityManager：用于管理所有的Subject

3、Realms：用于进行权限信息的验证

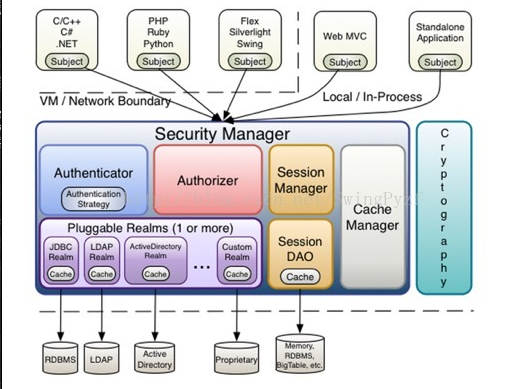
4、Authentication：是验证用户身份的过程。

5、Authorization：是授权访问控制，用于对用户进行的操作进行人证授权，证明该用户是否允许进行当前操作

6、SessionManager ：Shiro为任何应用提供了一个会话编程范式。

7、CacheManager :对Shiro的其他组件提供缓存支持。

Shiro 完整架构图



# spring boot

### 热部署

1、springloaded

<plugin>

                     <groupId>org.springframework.boot</groupId>

                     <artifactId>spring-boot-maven-plugin </artifactId>

                     <dependencies>

                       <!--springloaded  hot deploy -->

                       <dependency>

                           <groupId>org.springframework</groupId>

                           <artifactId>springloaded</artifactId>

                           <version>1.2.4.RELEASE</version>

                       </dependency>

                    </dependencies>

                    <executions>

                       <execution>

                           <goals>

                               <goal>repackage</goal>

                           </goals>

                           <configuration>

                               <classifier>exec</classifier>

                           </configuration>

                       </execution>

                     </executions>

</plugin>

如果是使用spring-boot:run的话，那么到此配置结束

如果使用的run as – java application的话，那么还需要做一些处理：

把spring-loader-1.2.4.RELEASE.jar下载下来，放到项目的lib目录中，然后把IDEA的run参数里VM参数设置为：-javaagent:.\lib\springloaded-1.2.4.RELEASE.jar -noverify

2、devtools

### 全局异常捕捉

1、新建一个类GlobalDefaultExceptionHandler，

2、在class注解上@ControllerAdvice,

3、在方法上注解上@ExceptionHandler(value = Exception.class)

### Spring Boot datasource - mysql

(1)在application.properties中加入datasouce的配置

########################################################

###datasource

########################################################

spring.datasource.url = jdbc:mysql://localhost:3306/test

spring.datasource.username = root

spring.datasource.password = root

spring.datasource.driverClassName = com.mysql.jdbc.Driver

spring.datasource.max-active=20

spring.datasource.max-idle=8

spring.datasource.min-idle=8

spring.datasource.initial-size=10

(2)在pom.xml加入mysql的依赖。

<dependency>

       <groupId>mysql</groupId>

       <artifactId>mysql-connector-java</artifactId>

</dependency>

(3)获取DataSouce的Connection进行测试。

### spring boot jpa-hibernate

1、在pom.xml添加mysql,spring-data-jpa依赖

<dependency>

       <groupId>mysql</groupId>

       <artifactId>mysql-connector-java</artifactId>

 </dependency>

<dependency>

           <groupId>org.springframework.boot</groupId>

           <artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

2、在application.properties文件中配置mysql连接配置文件

########################################################

###datasource

########################################################

spring.datasource.url = jdbc:mysql://localhost:3306/test

spring.datasource.username = root

spring.datasource.password = root

spring.datasource.driverClassName = com.mysql.jdbc.Driver

spring.datasource.max-active=20

spring.datasource.max-idle=8

spring.datasource.min-idle=8

spring.datasource.initial-size=10

3、在application.properties文件中配置jpa配置信息

########################################################

### Java Persistence Api

########################################################

# Specify the DBMS

spring.jpa.database = MYSQL

# Show or not log for each sql query

spring.jpa.show-sql = true

# Hibernate ddl auto (create, create-drop, update)

spring.jpa.hibernate.ddl-auto = update

# Naming strategy

spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.ImprovedNamingStrategy

# stripped before adding them to the entity manager)

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

4、编写测试例子

实体类加@Entity，主键加 @Id 和 @GeneratedValue

### 使用jpa-hibernate保存数据

(1) 创建实体类Demo,如果已经存在，可以忽略。

(2) 创建jpa repository类操作持久化。

(3) 创建service类。

(4) 创建restful请求类。

(5) 测试

### 使用JdbcTemplate

1、在pom.xml加入jdbcTemplate的依赖；

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

2、编写DemoDao类，声明为：@Repository，引入JdbcTemplate

@Resource

private JdbcTemplate jdbcTemplate;

String sql = "insert into Demo(name,age) values(?,?)";

jdbcTemplate.update(sql, new Object[]{demo.getName(),demo.getAge()});

3、编写DemoService类，引入DemoDao进行使用

4、编写Demo2Controller进行简单测试。

### 修改端口号

Spring boot 默认端口是8080，如果想要进行更改的话，只需要修改applicatoin.properties文件，在配置文件中加入：

server.port=9090

### 配置ContextPath

Spring boot默认是/ ，这样直接通过http://ip:port/就可以访问到index页面，如果要修改为http://ip:port/path/ 访问的话，那么需要在Application.properties文件中加入server.context-path = /你的path

### 改变JDK编译版本

需要修改pom.xml文件的<build> -- <plugins>加入一个plugin即可

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>