北京市朝阳区酒仙桥北路7号电通创意广场三号楼A区

Git：

取消在当前目录下进行本地初始化（不小心输入了git init）

执行rm -rf .git

添加远程

git remote rm origin

git remote add origin https://github.com/xnx123/note.git

删除本地文件，让云中也没有了这个文件

git rm git.txt

git commit -m "a"

git push -u origin master

在自己的项目目录中建仓库：git init

#创建本地test分支

$ git branch test

##查看本地分支有哪些

$ git branch

doc

\* master

##将本地test分支推送到远程服务器

$ git push origin test

##切换到test分支

$ git checkout test

Switched to branch 'test'

##提交信息

giscafer@LAOHOUBIN-PC /G/002\_project/Comments (test)

$ git commit -m '提交数据结构表设计文档到test分支上'

[test 867e877] 提交数据结构表设计文档到test分支上

查看运程分支：

git branch –r

查看全部分支：

git branch –a

本地分支推送到远程分支

git push -u origin master

SSH

（1） 首先在客户端生成一对密钥（ssh-keygen）；

　　（2） 并将客户端的公钥ssh-copy-id 拷贝到服务端；

　　（3） 当客户端再次发送一个连接请求，包括ip、用户名；

　　（4） 服务端得到客户端的请求后，会到authorized\_keys中查找，如果有响应的IP和用户，就会随机生成一个字符串，例如：qwer；

　　（5） 服务端将使用客户端拷贝过来的公钥进行加密，然后发送给客户端；

　　（6） 得到服务端发来的消息后，客户端会使用私钥进行解密，然后将解密后的字符串发送给服务端；

（7） 服务端接受到客户端发来的字符串后，跟之前的字符串进行对比，如果一致，就允许免密码登录。

FTP

public interface FtpServerContext extends FtpletContext {

ConnectionConfig getConnectionConfig();

MessageResource getMessageResource();

FtpletContainer getFtpletContainer();

Listener getListener(String name);

Map<String, Listener> getListeners();

CommandFactory getCommandFactory();

void dispose();

ThreadPoolExecutor getThreadPoolExecutor();

}

死锁

产生死锁的四个必要条件：

（1） 互斥条件：一个资源每次只能被一个进程使用。  
（2） 请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。  
（3） 不剥夺条件:进程已获得的资源，在末使用完之前，不能强行剥夺。  
（4） 循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

HTLML

<body>标签里面的java代码几乎与js里面的java代码同时执行，实际上很多时候两时间都一样。

<%xxx%>这是jsp中嵌入的java程序，是在服务器端运行的。  
<script>xxx</script>这是页面脚本，是在浏览器客户端运行的。

3.<body>里面的onload方法在页面结束加载之后触发。

**• <html>**

**• <head>**

**• <script>**

•     alert("script-head");

**• </script>**

**• </head>**

**• <body** onload="alert('html-tag');"**>**

**• <script>**

•     alert("script-body");

**• </script>**

**• </body>**

**• <script>**

•     alert("script-body-outer");

**• </script>**

**• </html>**

**• <script>**

•     alert("script-html-outer");

**• </script>**

script-head、script-body、script-body-outer、script-html-outer、html-tag为输出结果。

Spring

@RestController

@RequestMapping("/classPath")

**public** **class** **Application** {

@RequestMapping("/methodPath")

**public** String **method**() {

**return** "mapping url is /classPath/methodPath";

}

}

@RequestMapping("/users/{username}")

**public** String **userProfile**(@PathVariable("username") String username) {

**return** String.format("user %s", username);

}

@RequestMapping("/posts/{id}")

**public** String **post**(@PathVariable("id") **int** id) {

**return** String.format("post %d", id);

}

在之前所有的@RequestMapping注解的方法中，返回值字符串都被直接传送到浏览器端并显示给用户。但是为了能够呈现更加丰富、美观的页面，我们需要将HTML代码返回给浏览器，浏览器再进行页面的渲染、显示。

一种很直观的方法是在处理请求的方法中，直接返回HTML代码，但是这样做的问题在于——一个复杂的页面HTML代码往往也非常复杂，并且嵌入在Java代码中十分不利于维护。更好的做法是将页面的HTML代码写在模板文件中，渲染后再返回给用户。为了能够进行模板渲染，需要将@RestController改成@Controller：

Servlet（Server Applet）是Java Servlet的简称，称为小服务程序或服务连接器，用Java编写的服务器端程序，主要功能在于交互式地浏览和修改数据，生成动态Web内容。

该文本行的流可以这样处理：  
InputStream input =// get the InputStream from the client socket

1

BufferedReader reader = new BufferedReader(new InputStreamReader(input));

2

3

String nameLine   = reader.readLine();

4

String ageLine    = reader.readLine();

5

String emailLine  = reader.readLine();

6

String phoneLine  = reader.readLine();

本文介绍linux如何查看所有的用户和组信息的方法：

1、cat /etc/passwd；

2、cat /etc/group

3. userdel –r username

rmp软件包的安装可以使用程序rpm来完成。执行下面的命令  
rpm -i your-package.rpm

**执行可执行文件：**

**/etc/init.d/node restart（根目录）**

**./etc/init.d/node restart(当前目录)**

下面是linux系统约定不同类型文件默认的颜色：

• 白色：表示普通文件

• 蓝色：表示目录

• 绿色：表示可执行文件

• 红色：表示压缩文件

• 浅蓝色：链接文件

• 红色闪烁：表示链接的文件有问题

• 黄色：表示设备文件

• 灰色：表示其它文件

.git\refs\heads\master这个文件中保存的是本地库中最新的commit id;

remotes文件夹中的每一个文件夹代表一个远程库名称（git remote），其中的每个文件关联远程库的一个分支，其中保存该分支的最新commit id.

**private static String filePath = "d:\\科室测试.xlsx";**

**private static String titles = "医院名称,科室大类,科室小类,科室名称,科室图片,科室简介";**

**List<Map<String, Object>> list = new ArrayList<Map<String,Object>>();**

**FileInputStream fis = null;**

**XSSFWorkbook workbook = null;**

**XSSFSheet sheet = null;**

**int rowSize = 0;**

**try {**

**//get file**

**fis = new FileInputStream(new File(fpath));**

**//get excel file**

**workbook = new XSSFWorkbook(fis);**

**//get one sheet**

**sheet = workbook.getSheetAt(0);**

**//get total rowsize**

**rowSize = sheet.getPhysicalNumberOfRows();**

**String[] arr = titles.split(",");**

**for (int i = 0; i < arr.length; i++) {**

**if (!getCellValue(sheet,1,i).contains(arr[i])) {**

**throw new PHYYException("模板错误，请检查如下原因：1.请选择正确模板 2.模板的列顺序要正确 3.模板的列头要放在第2行");**

**}**

**}**

**// getCellValue(sheet,1,0).contains("医院名称") get value**

**//the first row is 1,the first cow is 0;**

**if (getCellValue(sheet,1,0).contains("医院名称")) {**

**for (int j = 2; j <= rowSize; j++) {**

**// 前6条数据有不为空的，才会处理**

**if (!getCellValue(sheet,j,0).equals("")**

**|| !getCellValue(sheet,j,1).equals("")**

**|| !getCellValue(sheet,j,2).equals("")**

**|| !getCellValue(sheet,j,3).equals("")**

**|| !getCellValue(sheet,j,4).equals("")**

**|| !getCellValue(sheet,j,5).equals("")**

**) {**

**Map<String, Object> map = handle(sheet,j);**

**list.add(map);**

**}**

**}**

**return list;**

**}else {**

**return null;**

**}**

**} catch (FileNotFoundException e) {**

**e.printStackTrace();**

**} catch (IOException e) {**

**e.printStackTrace();**

**} finally {**

**if (fis != null) {**

**try {**

**fis.close();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**return null;**

**}**

**}**

•

•

• //通过bean实例化对象

**• public** **class** SpringTest {

•

•     **public** **static** **void** main(String[] args) {

•         ApplicationContext ctx = **new** ClassPathXmlApplicationContext("bean.xml");

•         //判断两次请求singleton作用域的Bean实例是否相等

•         System.out.println(ctx.getBean("bean1")==ctx.getBean("bean1"));

•         //判断两次请求prototype作用域的Bean实例是否相等

•         System.out.println(ctx.getBean("bean2")==ctx.getBean("bean2"));

•     }

•

• }

1、value  / method 示例

默认RequestMapping("....str...")即为value的值；

 @RequestMapping(value="toupdatebusiness")

    public String toupdatebusiness(Model model, String id){

        Map<String, Object> business;

try {

business = businessServiceImpl.findOne(id);

model.addAttribute("business",business);

} catch (Exception e) {

e.printStackTrace();

}

        return "main/business/updatebusiness";

    }

<td align="left">

<input name="name" id="name" value="${business.name}" class="inputstyle" maxlength="200" style="margin: 0px 10px 0px 0px;width: 100%;" placeholder="企业名称长度最大为200"/>

</td>

**ajax 发送的消息需要@responsebody**

**点击update不需要--> return "main/business/addbusiness";**

**；**

request.getContextPath()，得到工程名：/xjjcz;

 request.getRequestURL()，返回IE地址栏地址：http://localhost:8080/dmsd-itoo-exam-log-web/course/index.jsp；

        request.getRequestURI() ，返回包含工程名的当前页面全路径：/dmsd-itoo-exam-log-web/course/index.jsp

如果在测试之前有些工作我们只想做一次，用不着每个函数之前都做一次。比如读一个很大的文件。那就用下面两个来标注：

@BeforeClass  
@AfterClass

@Autowired

private DiseaseServiceImpl diseaseService;

    在域变量上加上标签@Autowired,并且去掉 相应的get 和set方法

@RunWith就是一个运行器

@RunWith(JUnit4.class)就是指用JUnit4来运行

@RunWith(SpringJUnit4ClassRunner.class),让测试运行于Spring测试环境

@RunWith(Suite.class)的话就是一套测试集合

b b1 = **new** b();  
System.***out***.println(b.**class**.getResource(**"/"**));  
*//file:/Users/xienanxiang/Downloads/xie/out/production/xie/  
//return* 当前类所在的文件夹

• 1、#将传入的数据当成一个字符串，会对自动传入的数据加一个双引号。例如

order by #id#,如果传入的值是111，那么解析成sql时的值变为order by "111",如果传入的值是id,在解析成sql为order by "id"

其实原sql语句通常写成 order by #{id} 与order by #id#的效果一样

• 2

2、$将传入的数据直接显示在sql语句中。例如 order by ${id},如果传入的值是9则解析成sql语句为order by 9

如此，我们就不再需要在 XML 中显式使用 <bean/> 进行Bean 的配置。Spring 在容器初始化时将自动扫描 base-package 指定的包及其子包下的所有 class文件，所有标注了 @Repository 的类都将被注册为 Spring Bean。

@Resource可以作用于字段和函数上。当作用于字段上的时候，如果我们只是简单的这样写

@Resource

PersonDao p;

这时候spring注入p的过程是 1：先查找xml中是否有id为p的元素

2：如果没有找到，则看是否有name属性（@Resource name=“”），有则查找name

3:否则查找PersonDao类型的元素

@Resource可作用于set函数上。

例如：

@Resource

public void setP(PersonDao p) {

this.p = p;

}

@Autowired注解是根据类型进行查找，比如PersonDao p，他会去xml文件里查找类型为PersonDao的元素

作者：wuxinliulei  
链接：https://www.zhihu.com/question/39356740/answer/80926247  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

进程控制一般是由 OS 的内核中的原语来实现的。

由于 S 被初始化为 0，这样，若 P2 先执行必定阻塞，

只有在进程 P1 执行完 S1;signal(S);操作后使 S 增为 1 时， P2 进程方能执行语句 S2 成功。同样，我们可以利用信号 量，按照语句间的前趋关系(见图 2-12)，写出一个更为复 杂的可并发执行的程序。

如果A、B两个项目直接有依赖关系，如A项目依赖B项目，B为Library module（可以在项目的Facet中设置B为Library module），并且B中含有第三方的jar包（都放在B项目的libs文件夹中），A项目也用到了这些jar包，需要在B项目的Dependencies中勾选上Export。如下图：

**一定要学会下载别人的项目。**

**发布项目后，项目用web-inf下的jar**

//-未指定欢迎页时，缺省等于如下配置。这个应该不同的Web服务器可以设置，但大多数都如此-

<welcome-file-list>  
<welcome-file>index.html</welcome-file>  
<welcome-file>index.htm</welcome-file>  
<welcome-file>index.jsp</welcome-file>  
</welcome-file-list>

**Spring框架：**

**[2017-10-15 05:02:29,185] Artifact untitled:Web exploded: Artifact is being deployed, please wait...**

**15-Oct-2017 17:02:29.529 严重 [RMI TCP Connection(2)-127.0.0.1] org.apache.catalina.core.StandardContext.startInternal One or more listeners failed to start. Full details will be found in the appropriate container log file**

**15-Oct-2017 17:02:29.534 严重 [RMI TCP Connection(2)-127.0.0.1] org.apache.catalina.core.StandardContext.startInternal Context [/unnamed] startup failed due to previous errors**

**[2017-10-15 05:02:29,548] Artifact untitled:Web exploded: Error during artifact deployment. See server log for details.**

**15-Oct-2017 17:02:39.124 信息 [ContainerBackgroundProcessor[StandardEngine[Catalina]]] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [/Users/xienanxiang/Downloads/apache-tomcat-9.0.0.M26/webapps/manager]**

**在搭建spring项目后发布的时候这些的jar需要放到web-inf下的lib下，不然就不能成功发布。**

**直接建立maven**

**maven----maven-web:**

**Maven-maven-web-++++spring**

**+spring会自动添加applicationContext，dispatcher-servlet.xml**

**java工程**

**java-web-structs**

**spring工程（多了点包）**

**添加jar：首先在src／webapp／web-inf/lib下的jar肯定能发布到目的地，其次通过maven下载的jar**

**project structure**

**artifact：加如图**

读文件：

String encoding="GBK";

File file=new File(filePath);

InputStreamReader read = new InputStreamReader(

new FileInputStream(file),encoding);

BufferedReader bufferedReader =new BufferedReader(read);

  String lineTxt = null;

 while((lineTxt = bufferedReader.readLine()) != null){

                        System.out.println(lineTxt);

InputStream.read(byte[] b)

这个方法是先规定一个数组长度，将这个流中的字节缓冲到数组b中，返回的这个数组中的字节个数，这个缓冲区没有满的话，则返回真实的字节个数，到未尾时都返回-1

read file／write file

http://www.jb51.net/article/47062.htm              }

**bufferreader中能存8192个字符，io级别；**

bufferedReader 在字符char层面

inputstreamreader在字符

inputstream在字节层面

可以的首先，java中的一个char的确是2个字节。java采用unicode，2个字节来表示一个字符。一个数字或英文或汉字都是一个字符，只不过数字和英文时，存储的2个字节的第一个字节都为0，就是浪费了点空间。存汉字就占满了2个字节。

http://blog.csdn.net/u013063153/article/details/70237241

方法区中有：静态变量，常量，类信息，运行时常量池。

栈：基本类型变量，对象的引用变量。