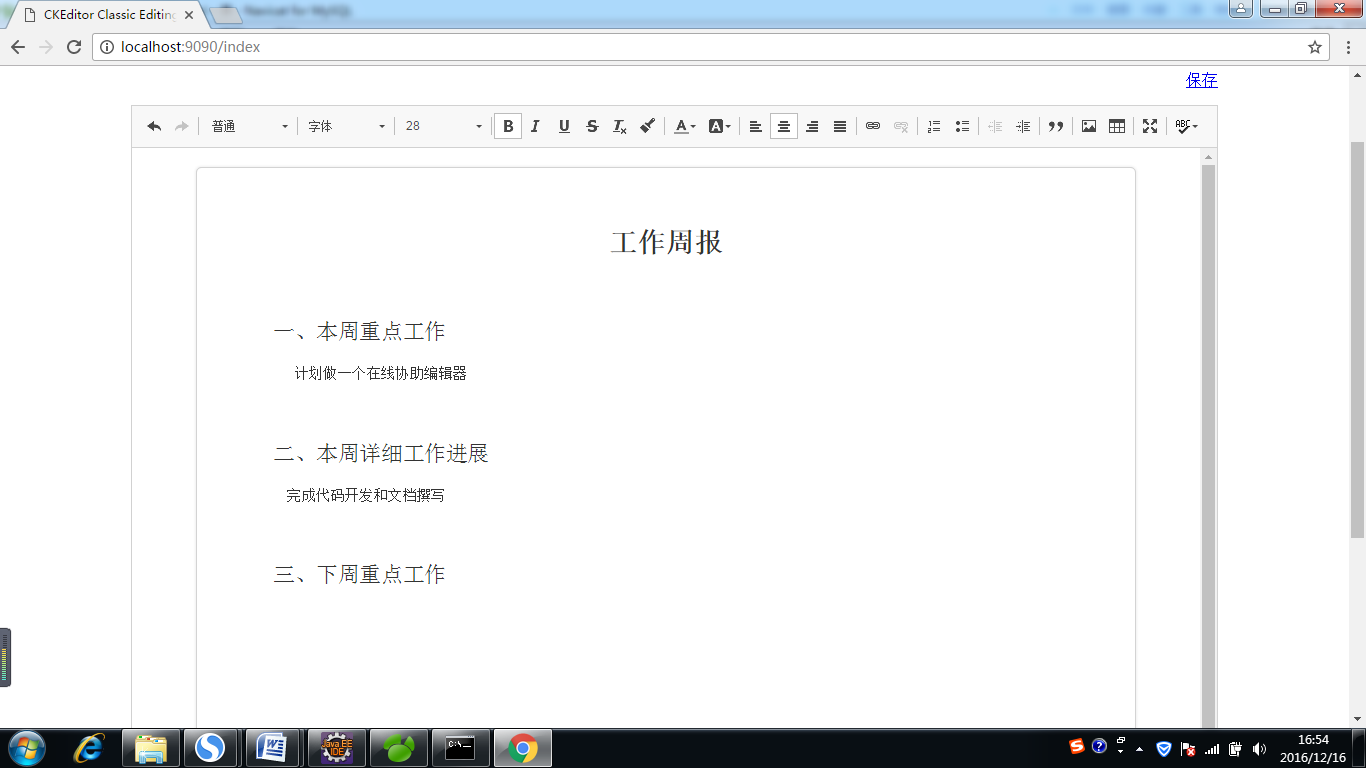
## 一、实验简介

一个多人在线协作的文档编辑器，例如可以用于工作中小团队或者小组内进行实时编写和收集周报。用到的库（框架）主要有：

* ckeditor （在线的编辑器）
* localStorage （HTML 5 Web 存储）
* springmvc (轻量级Web框架)
* spring jdbcTemplate
* mysql数据库

本节实验完成的效果图如下，供参考：



完成的功能：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **需求名称** | **需求详述** | **重要程度** | **功能** |
| 1 | 在线协作编辑 | 嵌入浏览器网页的在线编辑器，支持多人协作编辑，刷新后读取后台最新编辑保存的内容 | 很重要 | 前台 |
| 2 | 实时自动保存 | 通过在线编辑器，输入的内容，能够实时自动保存到客户端；刷新后数据不丢失；点击提交能够保存到数据库中 | 很重要 | 后台 |

Github源码下载：

<https://github.com/abshawn/WeeklyReport.git>

## 二、创建项目

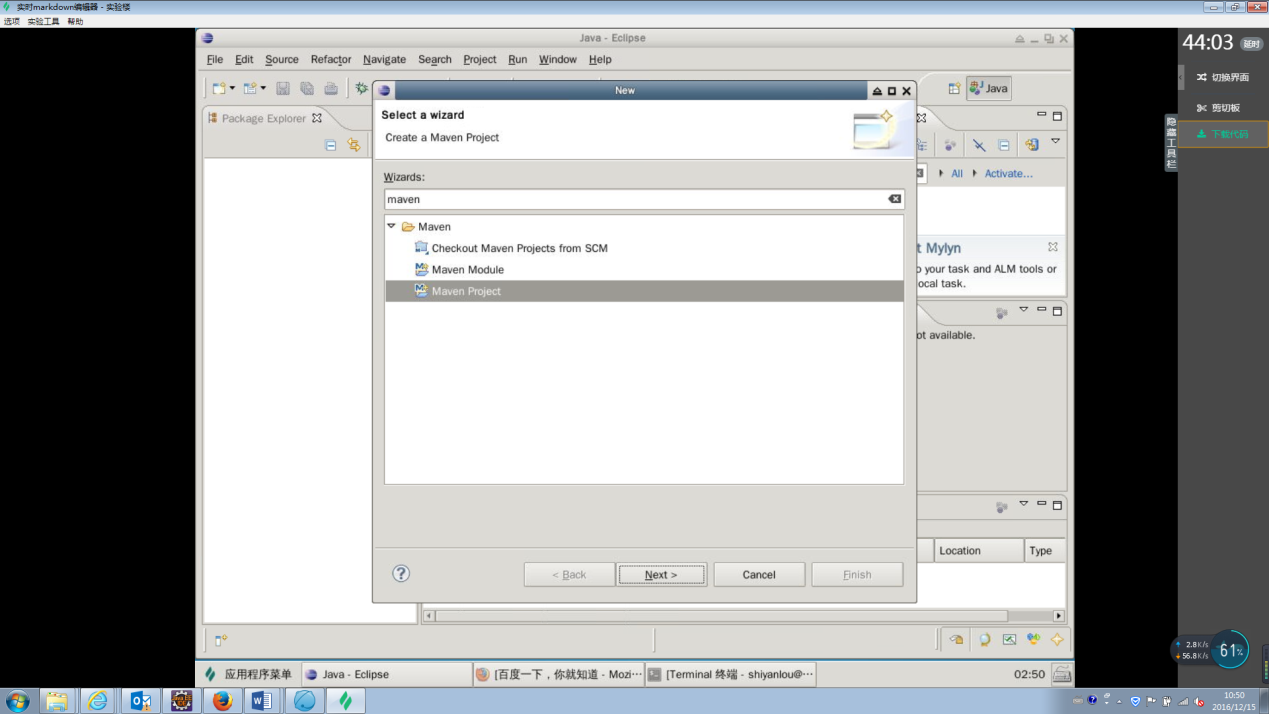
在虚拟机中创建Maven项目，搭建spring mvc的环境；

### 新建maven项目

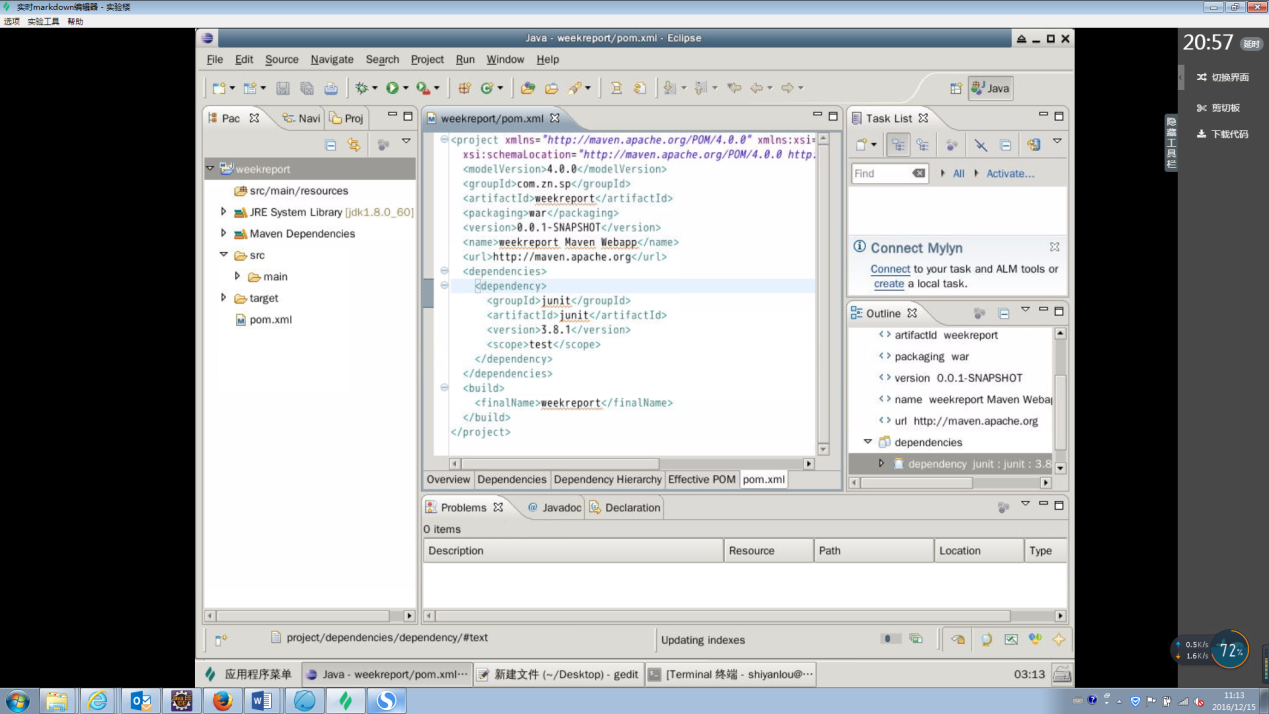
打开eclipse，新建maven项目，如下图所示，Filter输入web, 选择 maven-archetype-web项目，输入如下信息后，回车生成如下图程序目录

<groupId>com.zn.sp</groupId>

<artifactId>WeekReport</artifactId>



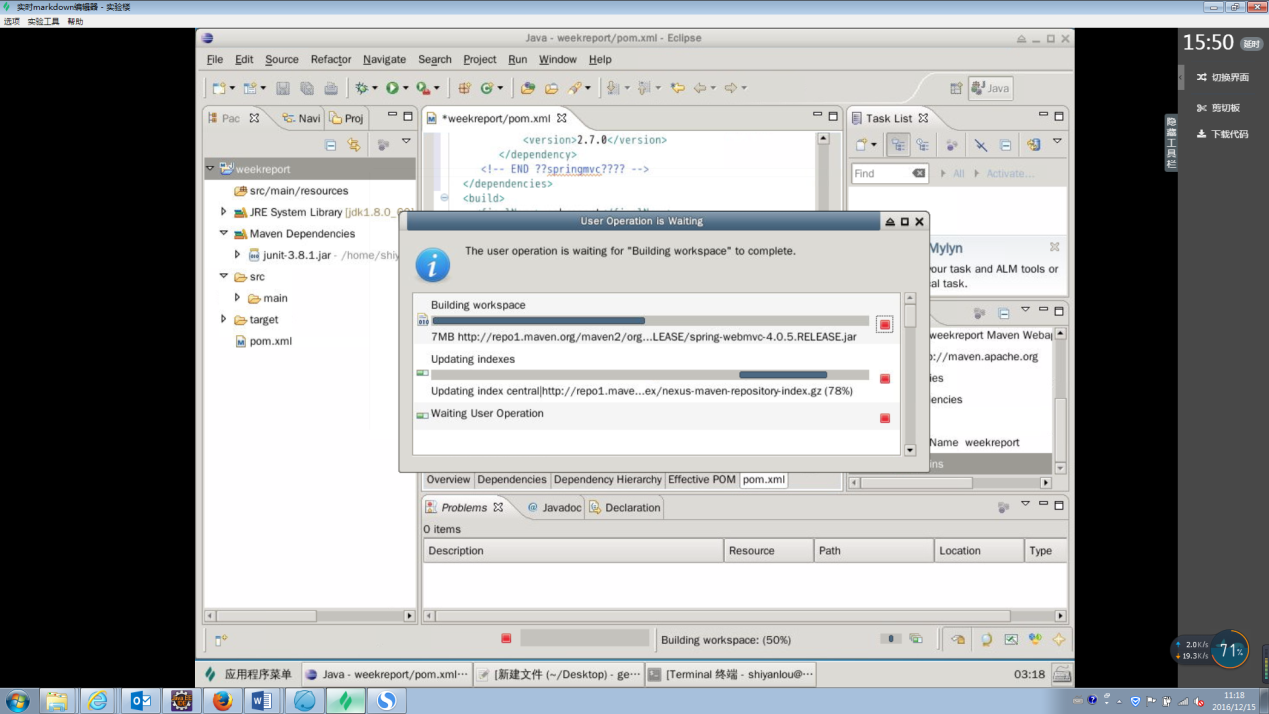
图一



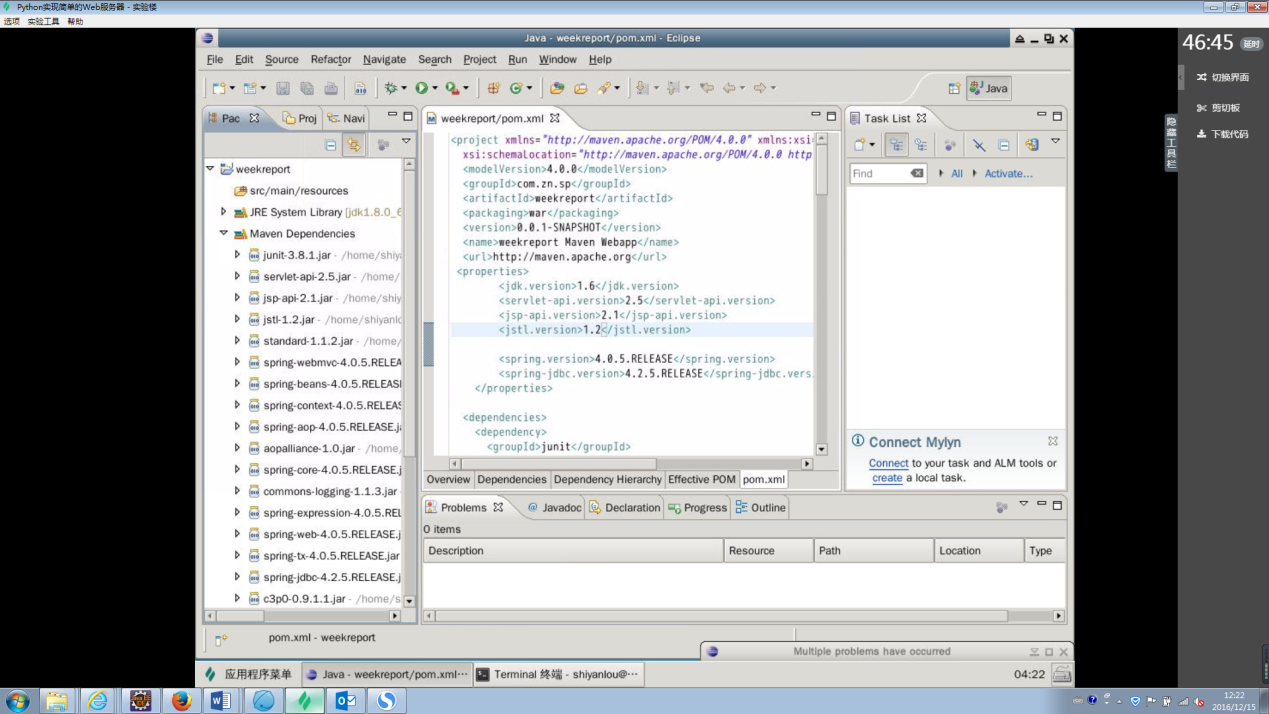
图二

### 引入spring相关包

打开编辑pom文件，引入spring、mysql等相关jar包；



图三



图四

### 配置spring和springmvc文件

在 src/main/resources下新建3个配置文件

applicationContext.xml： spring的配置文件

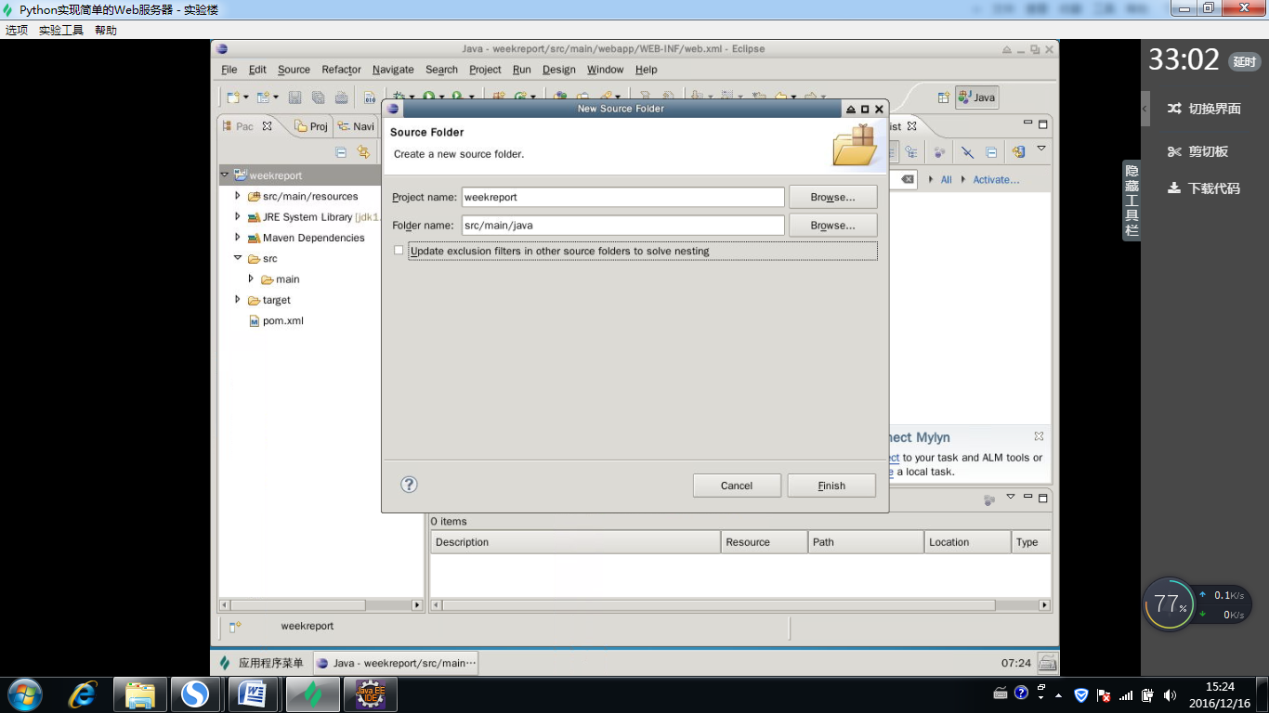
jdbc.properties：连接mysql数据库的配置文件

spring-mvc.xml： springmvc的配置文件

另外：修改web.xml文件，配置utf-8中文编码，避免中文乱码的问题出现。

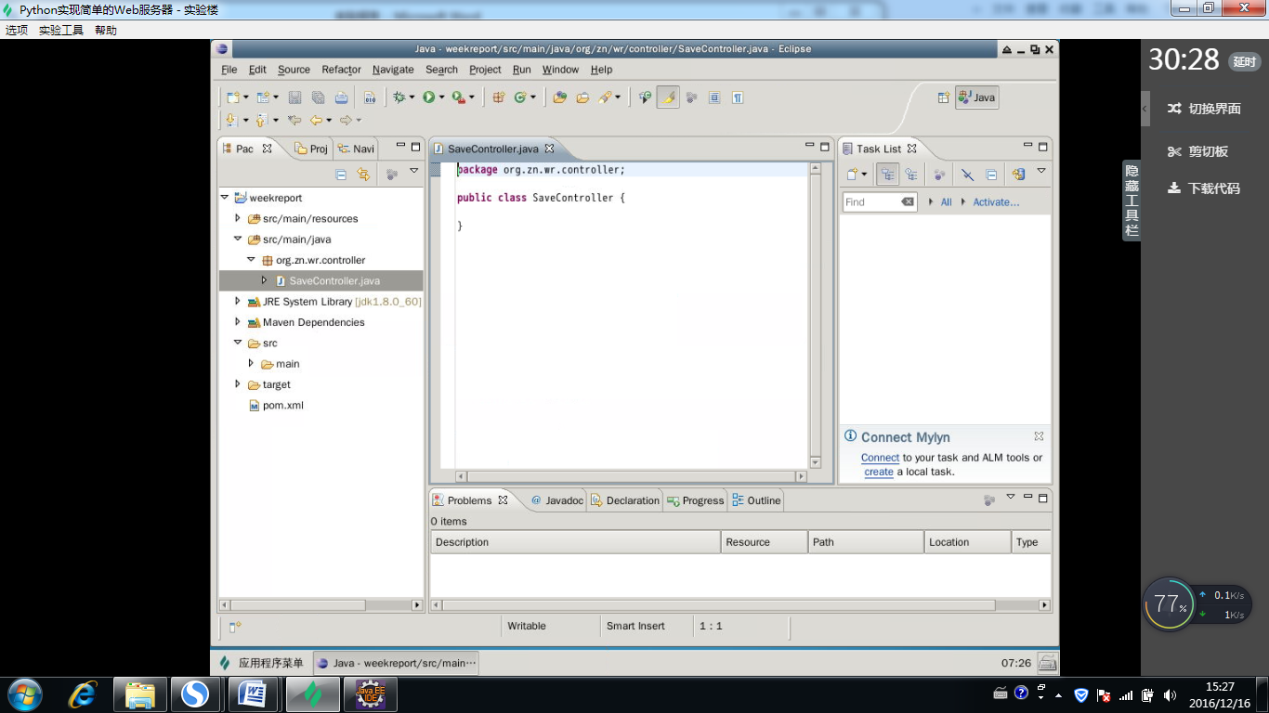
### 通过action跳转到index.jsp页面

新建source folder ：src/main/java

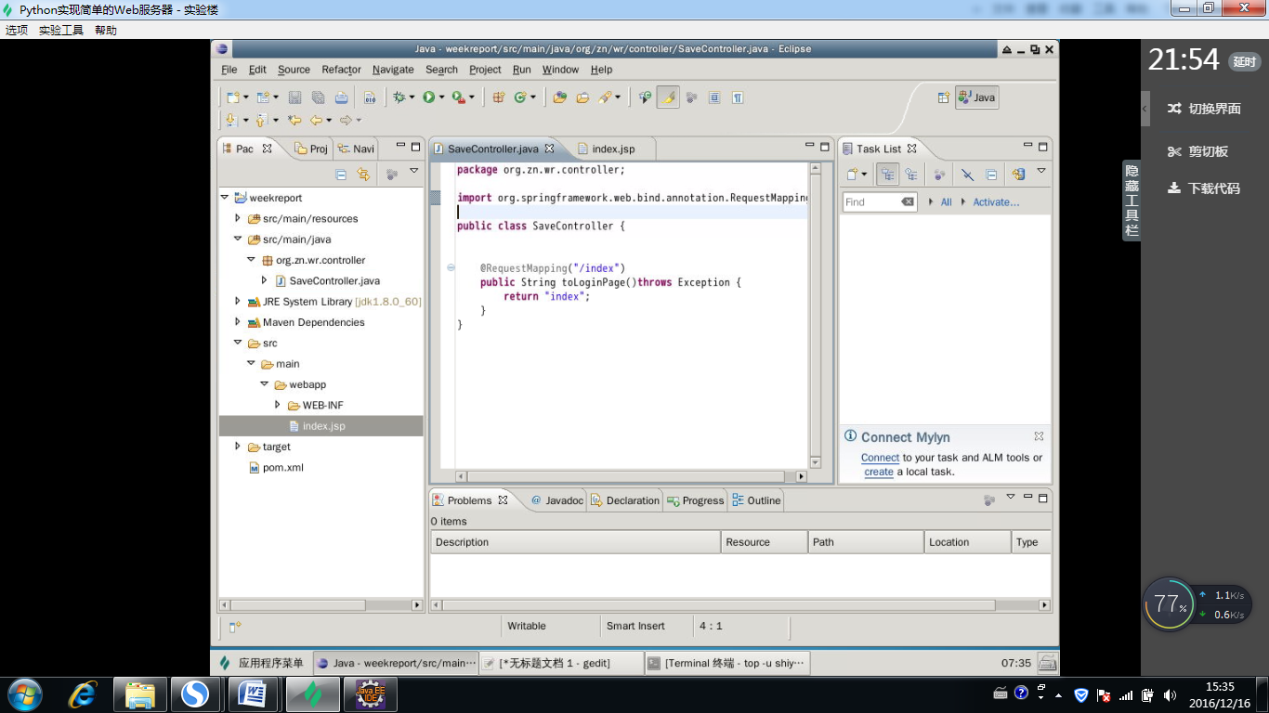


图五

在src/main/java下，新建package org.zn.wr.controller，并新增SaveController.java文件



图六



图七

通过设置@RequestMapping /index ，来进行请求跳转。

## 三、实现“在线协作编辑”功能

在线编辑器，使用ckeditor插件,官网 <http://ckeditor.com/> 有一些在线的demo可以参考，调用也很简单，只需要引入对应的js文件，即可：

<script src="https://cdn.ckeditor.com/4.6.0-441b33b/full-all/ckeditor/ckeditor.js"></script>

<script src="ckeditor/plugin-ajax.js"></script>

/\*\*

\* 初始化编辑框

\*/

Function initedior(){

var editor= CKEDITOR.replace( 'editor1' , {

toolbar: [

{ name: 'clipboard', items: [ 'Undo', 'Redo' ] },

{ name: 'styles', items: [ 'Format', 'Font', 'FontSize' ] },

{ name: 'basicstyles', items: [ 'Bold', 'Italic', 'Underline', 'Strike', 'RemoveFormat', 'CopyFormatting' ] },

{ name: 'colors', items: [ 'TextColor', 'BGColor' ] },

{ name: 'align', items: [ 'JustifyLeft', 'JustifyCenter', 'JustifyRight', 'JustifyBlock' ] },

{ name: 'links', items: [ 'Link', 'Unlink' ] },

{ name: 'paragraph', items: [ 'NumberedList', 'BulletedList', '-', 'Outdent', 'Indent', '-', 'Blockquote' ] },

{ name: 'insert', items: [ 'Image', 'Table' ] },

{ name: 'tools', items: [ 'Maximize' ] },

{ name: 'editing', items: [ 'Scayt' ] }

],

height: 800 , //外面长方形 高多少

contentsCss: [ 'https://cdn.ckeditor.com/4.6.0-441b33b/full-all/ckeditor/contents.css', 'mystyles.css' ] , //有无里面的方格

allowedContent : true ,

disallowedContent: 'img{width,height,float}' ,

extraAllowedContent: 'img[width,height,align]' ,

extraPlugins: 'tableresize,uploadimage,uploadfile' ,

bodyClass: 'document-editor' ,

format\_tags: 'p;h1;h2;h3;pre' ,

removeDialogTabs: 'image:advanced;link:advanced'

});

return editor;

}

<body>

<textarea name="editor1" id="editor1"> </textarea>

</body>

这里我们创建了一个初始化编辑器的函数，并且设置了编辑器的主题样式以及模式等。这样就能够看到嵌入网页中的编辑器了，并且可以输入内容。

## 四、实现“实时自动保存”功能

### 前台

#### 增加实时缓存功能

在线编辑器，输入内容后为了保证不丢失文件，采用sessionStorage的方式进行本地缓存。

/\*\*

\* 自动实时保存

\*

\*/

$(document).ready(function () {

// 当编辑器内容改变时，触发自动保存功能

editor.on('change', function(evt) {

window.sessionStorage.setItem("comment\_top", evt.editor.getData());

});}

#### 增加手动保存功能

添加手动保存功能，点击保存后，将编辑好的数据存到mysql数据库中。

/\*\*

\* 手动保存到mysql数据库中

\*

\*/

function ManualSave(){

var aj = $.ajax( {

url:'save', // 跳转到 action

type:'POST',

dataType:'json',

contentType: 'application/json;chartset=UTF-8',

data:JSON.stringify(window.sessionStorage.getItem("comment\_top")), // 传批量的参数 list

success:function(data) {

if(data.state){

alert("保存成功！");

window.sessionStorage.removeItem("comment\_top");

window.location.reload();

}else{

alert("保存失败！");

}

},

error : function() {

alert("网络异常！");

}

});

};

<body>

<p><a href="javascript:void(0);" onclick="ManualSave();" >保存</a></p>

</body>

#### 刷新最新数据功能

/\*\*

\* 从后台初始化最新数据

\*

\*/

function initdata(editor,initcontent){

//显示进度条代码结束

var aj =$.ajax({

type : "POST",

url : "initdata",

async: false,

dataType : "json",

success : function(data) {

if (data == null) {

alert("没查到最新的信息");

} else if (!initcontent && typeof initcontent != "undefined" && initcontent != 0) {

editor.setData(data.scontent);

}else{

editor.setData(initcontent);

}

}

});

};

### 后台

#### ACTION方法

手动保存到mysql数据库；

后台初始化最新数据；

/\*\*

\* 手动保存到mysql数据库中

\*

\*/

import java.util.HashMap;

import java.util.Map;

import javax.annotation.Resource;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.ResponseBody;

import com.jd.wr.dao.contentDAO;

import com.jd.wr.model.Content;

@Controller

public class SaveController {

private Content content;

@Resource

private contentDAO contentdao;

@RequestMapping("/index")

public String toLoginPage()throws Exception {

return "index";

}

@ResponseBody

@RequestMapping(value = "initdata", method = RequestMethod.POST)

public Content toInitData(){

this.content = new Content();

this.content.setScontent(contentdao.search());

System.out.println("content:"+contentdao.search());

return content;

}

@RequestMapping(value="save", method = {RequestMethod.POST} ,produces = "text/html;charset=UTF-8")

@ResponseBody

public Map<String, Object> doSave(@RequestBody String initcontent) {

// System.out.println("initcontext:"+initcontext);

boolean state = contentdao.insert(initcontent);

Map<String, Object> modelMap = new HashMap<String, Object>();

modelMap.put("state", state);

return modelMap;

}

}

#### Dao方法

/\*\*

\* 实现action相应的后台操作

\*

\*/

import java.util.HashMap;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Repository;

@Repository

public class contentDAO {

@Autowired

private JdbcTemplate jdbcTemplate;

public boolean insert(String content)

{

try {

String sql = " INSERT INTO w\_content(content,createtime,updatetime )VALUES (?,now(),now())";

Object[] params = new Object[] { content };

jdbcTemplate.update(sql, params);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

public String search()

{

String content = null;

try {

String sql = "SELECT content from w\_content ORDER BY updatetime DESC LIMIT 1";

content = (String) jdbcTemplate.queryForObject(sql, java.lang.String.class);

} catch (Exception e) {

e.printStackTrace();

}

return content;

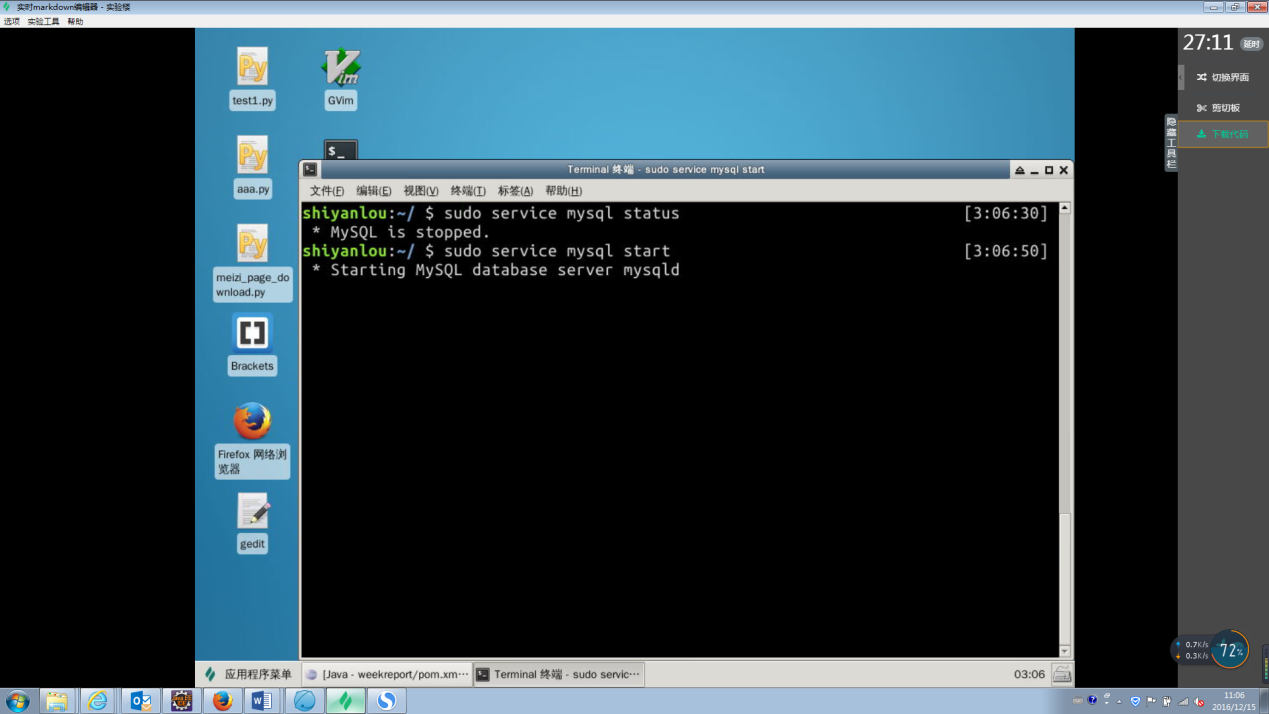
}

}

## 五、数据库

### 启动mysql数据库

1）启动mysql，如有问题的课参考实验楼中的【SQL的介绍及MySQL的安装】这个课程 <https://www.shiyanlou.com/courses/9/labs/48/document> 。



### 创建数据库、表

创建数据库：

create database weekreport;

创建表：

Navicat MySQL Data Transfer

Source Server : localhost\_3306

Source Server Version : 50715

Source Host : localhost:3306

Source Database : weekreport

Target Server Type : MYSQL

Target Server Version : 50715

File Encoding : 65001

Date: 2016-12-21 10:47:58

\*/

SET FOREIGN\_KEY\_CHECKS=0;

-- ----------------------------

-- Table structure for w\_content

-- ----------------------------

DROP TABLE IF EXISTS `w\_content`;

CREATE TABLE `w\_content` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT,

`content` text,

`createtime` datetime DEFAULT NULL,

`updatetime` datetime DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=40 DEFAULT CHARSET=utf8;

SET FOREIGN\_KEY\_CHECKS=1;

到此，可以启动 <http://localhost:9090/index>就可以正常访问和使用了。

## 六、待改进问题

* 本次项目课的内容比较多，建议多动手操作几遍，并且仔细回顾和思考，才能真正理解。
* 目前还不支持导出功能；可以考虑添加另存为word、pdf等格式的文件；
* 另外，还可以添加工作流、还可以考虑添加邮件、短信通知功能。