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

计划课程安排：

FUNDAMENTAL(7天)         该阶段已结束

OOP(11天)           该阶段已结束

JAVASE01(10天) 该阶段已结束

JAVASE02(12天) ←当前为该阶段第11天

WEBBASIC(7天)

JQUERY(2天)

DATABASE(6天)

JDBC(3天)

SERVLETJSP(12天)

SPRINGMYBATIS01(8天)

AJAX(2天)

SPRINGMYBATIS02(12天)

LINUX(2天)

NGINX(2天)

REDIS(2天)

T-STORE(5天)

预计合计：102天

预计剩余：64天

注：上述信息随时可能发生更改，仅供参考。

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

**2019年4月24日**

**Java就业班-第48天**

**8:30实训室2-晨讲**

今日晨讲主题——方法的重写与重载

我怎么记得这个内容有人讲过了呢……

**9:00实训室2-正课**

开始上课。

※什么是“池”：简单来说只要一提到“池”，一般指的就是控制数量和重用。

线程池概念代码：

package Thread;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

/\*\*

\* 线程池 ExecutorService

\*

\* 线程池主要解决两个问题：

\* 1.控制线程数量

\* 2.重用线程

\*

\* @author Administrator

\*

\*/

public class PoolDemo {

public static void main(String[] args) {

//创建线程池及设置线程数量

ExecutorService threadPool = Executors.newFixedThreadPool(5);

//布置任务

for (int i = 0; i < 50; i++) {

Runnable runn = new Runnable() {

public void run() {

try {

Thread t = Thread.currentThread();

System.out.println(t.getName() + "正在执行任务……");

Thread.sleep(2000);

System.out.println(t.getName() + "任务执行完毕！");

} catch (Exception e) {

System.out.println("线程被中断！！");

}

}

};

threadPool.execute(runn);//启动线程池中的所有线程

System.out.println("已指派一个任务给线程池处理——————→");

}

/\*

\* shutdown方法调用，线程池不再接收新的任务，

\* 并且会将当前线程池已缓存任务全部执行完毕后停止。

\*

\* shutdownNow方法调用后，线程池会立即调用

\* 内部线程的中断方法并停止线程池。

\*/

// threadPool.shutdown();

threadPool.shutdownNow();

System.out.println("线程池已关闭！");

}

}

**9:40实训室2-正课**

接下来还是拿课程项目练手，不过这次是最后一个版本了。

新建一个项目WebServer\_v17

README.txt--------------

本版本主要工作：收尾

1.使用线程池来管理处理客户端请求的线程，来执行任务ClientHandler

实现：

在WevServer类中添加一个属性，是线程池类型的。

在构造方法中初始化线程池。

在start方法中当接收到客户端连接后，创建ClientHandler

并交给线程池来处理。

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

功能实现：

package com.webserver.core;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

/\*\*

\* 服务端

\* @author Administrator

\*

\*/

//通过http://localhost:8088/可以实现本地访问

public class WebServer {

private ServerSocket server;

private ExecutorService threadPool ;//线程池

/\*\*

\* 构造方法，用于初始化

\*/

public WebServer() {

try {

System.out.println("正在启动服务器……");

server = new ServerSocket(8088);

//设置有50个线程的线程池

threadPool = Executors.newFixedThreadPool(50);

System.out.println("服务器启动完毕！");

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 服务端开始工作的方法

\*/

public void start() {

try {

while(true) {

System.out.println("等待客户端连接……");

Socket socket = server.accept();

System.out.println("一个客户端连接了！");

//启动一个线程处理该客户端请求

ClientHandler handler = new ClientHandler(socket);

threadPool.execute(handler);

}

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

WebServer server = new WebServer();

server.start();

}

}

**10:20实训室2-正课**

这里先不继续往下讲了，我们搞点课外的内容：如何生成验证码。

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.image.BufferedImage;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Random;

import javax.imageio.ImageIO;

/\*\*

\* 验证码生成

\* @author Administrator

\*

\*/

public class RandomImage {

public static void main(String[] args) throws IOException {

char chars[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ".toCharArray();

//创建画布 xxx.jpg

BufferedImage image = new BufferedImage(65, 25, BufferedImage.TYPE\_INT\_RGB);

//获取对这张图的画笔，用于往这张图上画内容

Graphics g = image.getGraphics();

//创建一个颜色（R,G,B）

Color coloer = new Color(255,210,210);

//设置画笔为这个颜色

g.setColor(coloer);

//使用画笔填充一个矩形，颜色为当前画笔指定颜色

g.fillRect(0, 0, 65, 25);

//使用画笔画字符

g.setColor(new Color(100,180,160));

g.setFont(new Font("Default",Font.BOLD,20));

//随机生成4个字符

Random rd = new Random();

for (int i = 0; i < 4; i++) {

//随机字符颜色

g.setColor(new Color(rd.nextInt(100),rd.nextInt(180),rd.nextInt(160)));

//随机生成字符间距15

g.drawString(chars[rd.nextInt(chars.length)] + "", i\*15+5, 18);

}

//画干扰线

for (int i = 0; i < 6; i++) {

g.setColor(new Color(rd.nextInt(100),rd.nextInt(180),rd.nextInt(160)));

g.drawLine(rd.nextInt(25), rd.nextInt(65), rd.nextInt(25), rd.nextInt(65));

}

ImageIO.write(image, "jpg", new FileOutputStream("image.jpg"));

System.out.println("图片已生成");

}

}

**11:37实训室2-正课**

接下来试试看怎么将验证码实装到课程项目上。

**15:00实训室2-正课**

（部分涉及到其他类的源码修改，此处没有全部贴出，如有需要可以下载up主的源码）

实装成功：

package com.webserver.servlet;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Random;

import javax.imageio.ImageIO;

import com.webserver.http.HttpRequest;

import com.webserver.http.HttpResponse;

/\*\*

\* 随机验证码

\* @author Administrator

\*

\*/

public class RandomImageServlet extends HttpServlet{

private static char chars[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ".toCharArray();

private static int IMAGE\_WIDTH = 65 ;

private static int IMAGE\_HEIGHT = 25 ;

@Override

public void service(HttpRequest request, HttpResponse response) {

System.out.println("RandomImageServlet:生成随机验证码图片……");

//创建画布 xxx.jpg

BufferedImage image = new BufferedImage(IMAGE\_WIDTH, IMAGE\_HEIGHT, BufferedImage.TYPE\_INT\_RGB);

//获取对这张图的画笔，用于往这张图上画内容

Graphics g = image.getGraphics();

//创建一个颜色（R,G,B）

Color coloer = new Color(255,210,210);

//设置画笔为这个颜色

g.setColor(coloer);

//使用画笔填充一个矩形，颜色为当前画笔指定颜色

g.fillRect(0, 0, IMAGE\_WIDTH, IMAGE\_HEIGHT);

//使用画笔画字符

g.setColor(new Color(100,180,160));

g.setFont(new Font("Default",Font.BOLD,20));

//随机生成4个数

Random rd = new Random();

for (int i = 0; i < 4; i++) {

//随机字符颜色

g.setColor(new Color(rd.nextInt(100),rd.nextInt(180),rd.nextInt(160)));

//随机生成字符间距15

g.drawString(chars[rd.nextInt(chars.length)] + "", i\*15+5, 18);

}

//画干扰线

for (int i = 0; i < 6; i++) {

g.setColor(new Color(rd.nextInt(100),rd.nextInt(180),rd.nextInt(160)));

g.drawLine(rd.nextInt(IMAGE\_HEIGHT), rd.nextInt(IMAGE\_WIDTH), rd.nextInt(IMAGE\_HEIGHT), rd.nextInt(IMAGE\_WIDTH));

}

try {

/\*

\* ByteArrayOutputStream是一个低级流，通过这个流写出的字节会保存到它内部的一个字节数组中。

\*/

ByteArrayOutputStream out = new ByteArrayOutputStream();

ImageIO.write(image, "jpg", out);

//获取输出流里已经写出来的字节（图片的所有字节）

byte data[] = out.toByteArray();

//将图片数据作为正文设置到响应对象中

response.setContentData(data);

response.putHeader("Content-Type", "image/jpeg");

} catch (IOException e) {

e.printStackTrace();

}

System.out.println("图片已生成");

}

}

**15:23实训室2-正课**

虽然在WebServer\_v17已经收尾了，但在原基础上还可以再做一个附加功能。

新建一个项目WebServer\_v18.

README.txt------------------

本版本主要工作：使用thymeleaf完成动态页面

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

**17:20实训室2-正课**

（部分涉及到其他类的源码修改，此处没有全部贴出，如有需要可以下载up主的源码）

功能实现：

package com.webserver.servlet;

import java.io.RandomAccessFile;

import java.io.UnsupportedEncodingException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import org.thymeleaf.TemplateEngine;

import org.thymeleaf.context.Context;

import org.thymeleaf.templateresolver.FileTemplateResolver;

import com.webserver.http.HttpRequest;

import com.webserver.http.HttpResponse;

/\*\*

\* 利用thymeleaf框架，将user.dat文件中用户信息

\* 展现在静态页面showAllUser.html中，并响应给客户端

\* @author Administrator

\*

\*/

public class ShowAllUserServlet extends HttpServlet{

@Override

public void service(HttpRequest request, HttpResponse response) {

System.out.println("ShowAllUserServlet:开始展示用户信息……");

/\*

\* 传建一个集合用于保存从user.dat文件中读取出来的所有用户信息，

\* 集合中的每个元素表示一个公户的信息。

\* 我们使用Map作为集合元素，其中key的用户的属性（用户名，密码等），

\* value是该属性对应的值（张三，123456），这意味着每个Map实例表示一个员工的相关信息

\*

\* OGNL3表达式的思想……

\*/

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

try (RandomAccessFile raf = new RandomAccessFile("user.dat","rw")){

for (int i = 0; i < raf.length()/100; i++) {//以100个字节码的长度作为数据块区分

//读取用户名

byte data[] = new byte[32];//取32个字节

raf.read(data);//读取数据

String userName = new String(data,"utf-8").trim();

raf.read(data);

String passWord = new String(data,"utf-8").trim();

int age = raf.readInt();

raf.read(data);

String nickName = new String(data,"utf-8").trim();

Map<String,String> user = new HashMap<>();

user.put("username", userName);

user.put("password", passWord);

user.put("nickname", nickName);

user.put("age", age+"");

list.add(user);

}

} catch (Exception e) {

}

/\*

\* thymeleaf提供的模板引擎

\* 就是利用这个类完成数据与静态页面的绑定

\*/

TemplateEngine eng = new TemplateEngine();

/\*

\* 解释器

\* 用于给引擎设置各种参数，以及从哪里寻找静态页面等。

\* FileTemplateResolver是基于文件系统寻找静态资源，加载静态文件资源使用的路径与File类一致

\*/

FileTemplateResolver resolver = new FileTemplateResolver();

//设置字符集，以便模板引擎可以按照正确的字符集加载页面

resolver.setCharacterEncoding("utf-8");

//将解释其设置到引擎中

eng.setTemplateResolver(resolver);

//以上操作后，引擎初始化完毕

/\*

\* Context是以树状结构组件所有需要在页面上呈现的数据

\*/

Context context = new Context();

context.setVariable("list", list);

/\*

\* 利用引擎将静态页面与数据进行绑定

\* 返回值是一个字符串，内容就是绑定好数据的html代码

\*/

String html = eng.process("./webapps/myweb/showAllUser.html", context);

//响应客户端

try {

response.setContentData(html.getBytes("utf-8"));

response.putHeader("Content-Type", "text/html");

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

}

}

showAllUser.xml------

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>用户列表</title>

</head>

<body>

<center>

<h1>用户列表</h1>

<table border="1">

<tr>

<td>用户名</td>

<td>密码</td>

<td>昵称</td>

<td>年龄</td>

</tr>

<tr th:each="user : ${list}">

<td th:text="${user.username}">Kamui</td>

<td th:text="${user.password}">abc123</td>

<td th:text="${user.nickname}">KT</td>

<td th:text="${user.age}">25</td>

</tr>

<tr>

<td>Subaru</td>

<td>abc121</td>

<td>ST</td>

<td>22</td>

</tr>

</table>

</center>

</body>

</html>

**18:58实训室2-晚自习**

本来今天的课程听得感觉还行的，谁知道到了下午讲师突然开始挂挡提速讲了一堆thymeleaf模板引擎的用法并实装到课程项目中去，中途我算是彻底断片了，但又不敢上百度查只好跟着他的代码写，而我根本就不知道我自己在写什么代码，哪怕是执行过后看到效果、出bug、异常报错，我TM根本就不知道是怎么回事、什么问题、怎么解决。听得我一脸懵逼。

完了完了，开始慌了……

晚自习就研究这个thymeleaf了。

**19:22实训室2-晚自习**

首先我了解到的是thymeleaf是一种模板引擎，而什么又是模板引擎？查了一下百度，简单的来说模板引擎就是：

将指定模板内容（字符串）中的特定标记（子字符串）替换一下便生成了最终需要的业务数据（比如网页）。

在了解模板引擎之前，貌似首先得搞清楚静态页面和动态页面的区别：

静态是相对于动态网页而言，是指没有后台数据库、不含程序和不可交互的网页。静态网页相对更新起来比较麻烦，适用于一般更新较少的展示型网站。容易误解的是静态页面都是htm这类页面，实际上静态也不是完全静态，他也可以出现各种动态的效果，如GIF格式的动画、FLASH、滚动字幕等。

※静态页面一般的页面是以html、htm为后缀的，在地址栏上可以看到。

所谓的动态网页，是指跟静态网页相对的一种网页编程技术。静态网页，随着html代码的生成，页面的内容和显示效果就基本上不会发生变化了--除非你修改页面代码。而动态网页则不然，页面代码虽然没有变，但是显示的内容却是可以随着时间、环境或者数据库操作的结果而发生改变的。

两者区别：

静态页面：是不能随时改动的，静态是一次性写好放在服务器上进行浏览的，如果想改动，必须在页面上修改，然后再上传服务器覆盖原来的页面，这样才能更新信息，比较麻烦，使用者不能随时修改。

动态页面：是可以随时改变内容的，有前后台之分，管理员可以在后台随时更新网站的内容，前台页面的内容也会随之更新，比较简单易学。

**19:41实训室2-晚自习**

首先确定下来称呼：Thymeleaf是一种模板引擎

再就是要使用Thymeleaf需要有依赖的jar包，即需要在项目目录下的配置文件（pom.xml）中加入以下代码：

<dependency>

<groupId>org.thymeleaf</groupId>

<artifactId>thymeleaf</artifactId>

<version>3.0.11.RELEASE</version>

</dependency>

随后了解到，thymeleaf的存在可以完全替代JSP，而JSP则是一种简化的Servlet。（Servlet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，具有独立于平台和协议的特性，主要功能在于交互式地浏览和生成数据，生成动态页面内容。）

Thymeleaf的主要目标在于提供一种可被浏览器正确显示的、格式良好的模板创建方式，因此也可以用作静态建模。你可以使用它创建经过验证的XML与HTML模板。**相对于编写逻辑或代码，开发者只需将标签属性添加到模板中即可。接下来，这些标签属性就会在DOM（文档对象模型）上执行预先制定好的逻辑。**

Thymeleaf的特点：

Thymeleaf在有网络和无网络的环境下皆可运行，即它可以让美工在浏览器查看页面的静态效果，也可以让程序员在服务器查看带数据的动态页面效果。这是由于它支持 html 原型，然后在 html 标签里增加额外的属性来达到模板+数据的展示方式。**浏览器解释 html 时会忽略未定义的标签属性，所以 thymeleaf 的模板可以静态地运行；当有数据返回到页面时，Thymeleaf 标签会动态地替换掉静态内容，使页面动态显示。这样非常有利于前后端的分离。**

而在今天出现过的Thymeleaf代码中，正好分为两种。

一种是在后端编写的逻辑代码：

/\*

\* thymeleaf提供的模板引擎

\* 就是利用这个类完成数据与静态页面的绑定

\*/

TemplateEngine eng = new TemplateEngine();

/\*

\* 解释器

\* 用于给引擎设置各种参数，以及从哪里寻找静态页面等。

\* FileTemplateResolver是基于文件系统寻找静态资源，加载静态文件资源使用的路径与File类一致

\*/

FileTemplateResolver resolver = new FileTemplateResolver();

//设置字符集，以便模板引擎可以按照正确的字符集加载页面

resolver.setCharacterEncoding("utf-8");

//将解释其设置到引擎中

eng.setTemplateResolver(resolver);

//以上操作后，引擎初始化完毕

/\*

\* Context是以树状结构组件所有需要在页面上呈现的数据

\*/

Context context = new Context();

context.setVariable("list", list);

/\*

\* 利用引擎将静态页面与数据进行绑定

\* 返回值是一个字符串，内容就是绑定好数据的html代码

\*/

String html = eng.process("./webapps/myweb/showAllUser.html", context);

然而这段代码也只是单纯的方法调用，如果要深究其底层原理需要下载Thymeleaf的源码jar包，暂时先不管吧。

另一种就是对应写在showAllUser.html中的前端页面代码：

<tr th:each="user : ${list}">

<td th:text="${user.username}">Kamui</td>

<td th:text="${user.password}">abc123</td>

<td th:text="${user.nickname}">KT</td>

<td th:text="${user.age}">25</td>

</tr>

目前我只了解到这种写法使用的是一种名叫“OGNL表达式”的一种具有明确规则体系的写法。在今天的课程项目代码中起到的作用是获取标签中的数据交付于后端处理……

**20:17实训室2-晚自习**

项目经理察觉到今天讲的Thymeleaf几乎没有人听懂，于是开始讲解了。

首先Thymeleaf是一种框架。（？）

其中很多功能、方法都是框架提供的功能，我们只需要按照流程设置就可以，因为框架的规则是已经制定好了的。

具体怎么使用，就像查字典一样，了解各种方法的功能与使用方法即可。

今天写的验证码功能，是一种通过程序动态生成的文件，而通过Thymeleaf生成的动态页面转为字符数组。最后输出到客户端展示。

**20:36实训室2-晚自习**

项目经理看出了我的疑惑，过来找上我问有什么不懂的。我表示原理不懂，我不会用。

随后项目经理的发言让我震惊，但仔细想了想也是这么个道理：

这个框架（任何框架）是别人写好的，写出来就是给你用的，使用方法也是别人制定好了的，正确的使用现有的稳定框架，远比你从头开始写框架要更有效率。当然也不是说你就不需要研究源代码了，只是现阶段你没必要去研究。

**…………**

**…………**

说的也是哦……

**20:45实训室2-晚自习**

明天下午18:30月考，这次月考不行的人可以直接留级了。不过虽说是留级，近期的3月班和4月班学员已经接近满员了，哪怕是留级也不见得能够进得了这两个班，所以只能各自好自为之了。

天杀的，热死爹了！这室内温度谁TM受得了啊！不行了不行了，得赶紧回去洗澡吹空调。明天晚自习要还是这种温度我就直接早退回宿舍自习得了。

今日感想总结：所以说……到底要不要学着造轮子？嗯………

今日评分：2/10分

今天就到这里，明天继续投稿。

感想支持！

================================================================

**Acer程序员群：790482850**

**up主的java学习日记word文件及全部代码（随日记同步更新）：**

Adress：github.com/wt62635