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

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(8天) ←当前为该阶段第7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：100天

预计剩余：62天

注：上述信息随时可能发生更改，仅供参考。

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

**2019年4月17日**

**Java就业班-第42天**

**8:37实训室2-早自习**

今日晨讲主题——对象的引用

下周四月考。

**9:07实训室2-正课**

远程直播的音频再次无声……无声直播教学，厉害了。

看画面上的操作似乎是在对昨天写的项目代码进行回顾讲解。

/\*\*

设置响应正文对应实体文件

设置正文额同时会自动向相应对象中添加说明

响应正文的响应头：Content-Type和Content-Length

**9:26实训室2-正课**

今日课程主体为——XML（可扩展标记语言（Extensible Markup Language））

Html早期其实也是XML的一个分支（但Html已经有自己的一套体系了，只是语法部分还是以XML为基准）。

**9:34实训室2-正课**

讲师开始聊天了。

提到外包工作，简单来说就是人力派遣。为什么能活下来，因为国人在给全世界写项目。

就因为一点，我们是廉价劳动力。

996其实就是明目张胆耍流氓，在这个僧多粥少的时代，有多少人敢随便跳槽、告老板，虽然一告一个准，但是公司肯定会找机会把你开了。

讲师说他有个同事，6点活就干完了，啥事也不干坐到9点再上传。

如果面试的时候hr问能不能接受996工作模式，回答不能就没戏了。回答能的话，对方就要开始耍流氓了，就会开始聊理想了，潜台词就是说996是正常上班时间，没有加班费。

国人占了太多太多的基础工作岗位，打个比方，国外招一个洗碗工，市场时薪20美元，国人表示5美元就干。比方可能比较偏激，但映射到现实是不违和的。

打住了。

**10:26实训室2-正课**

关于XML的基本语法。

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**list**>

<**emps** id=*"Num 9"* name=*"Diva"* age=*"20"* gender=*"female"* salary=*"10000"*/>

<**emps** id=*"Num 8"* name=*"Also"* age=*"32"* gender=*"male"* salary=*"50000"*></**emps**>

<**emps**>

<**id**>Num 1</**id**>

<**name**>Kamui</**name**>

<**age**>25</**age**>

<**gender**>male</**gender**>

<**salary**>5000</**salary**>

</**emps**>

<**emps**>

<**id**>Num 2</**id**>

<**name**>Subaru</**name**>

<**age**>28</**age**>

<**gender**>male</**gender**>

<**salary**>8000</**salary**>

</**emps**>

<**emps**>

<**id**>Num 3</**id**>

<**name**>Luna</**name**>

<**age**>17</**age**>

<**gender**>female</**gender**>

<**salary**>20000</**salary**>

</**emps**>

<**emps**></**emps**>

<**emps**></**emps**>

<**emps**></**emps**>

<**emps**></**emps**>

<**emps**></**emps**>

</**list**>

**11:20实训室2-正课**

关于如何配置jar包。

例：如果想要使用dom4j，则必须要导入.jar包：

（注：dom4j是java用于解析xml文件的一个API，能够快速地读写xml文件。）

<**project** xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<**modelVersion**>4.0.0</**modelVersion**>

<**groupId**>Acfun.cn</**groupId**>

<**artifactId**>Kamui\_XML\_CodeTraning</**artifactId**>

<**version**>0.0.1-SNAPSHOT</**version**>

<**dependencies**>

<**dependency**>

<**groupId**>dom4j</**groupId**>

<**artifactId**>dom4j</**artifactId**>

<**version**>1.6.1</**version**>

</**dependency**>

</**dependencies**>

</**project**>

**12:03实训室2-午休时间**

上午的课程闲聊比较多，主要介绍了一些XML的相关信息和如何导入dom4j的包，出了dom4j以外还有其他的包可以用来解析xml文档，这里只是以传统的dom4j举例。

下午再通过这个包的使用进行效果测试。

**14:02实训室2-正课**

开始上课。

首先建立一个用于保存信息用的类：

package xml;

/\*\*

\* 员工对象

\* 该类每实例用于表示一个员工信息

\* **@author** Administrator

\*

\*/

public class Emp {

private int id;

private String name;

private int age;

private String gender;

private int salary;

public Emp(int id, String name, int age, String gender, int salary) {

super();

this.id = id;

this.name = name;

this.age = age;

this.gender = gender;

this.salary = salary;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getGender() {

return gender;

}

public void setGender(String gender) {

this.gender = gender;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

*@Override*

public String toString() {

return "Emp [id=" + id + ", name=" + name + ", age=" + age + ", gender=" + gender + ", salary=" + salary + "]";

}

}

随后开始测试dom4j中的各种方法：

emplist.xml

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**list**>

<!-- <emps id="Num 9" name="Diva" age="20" gender="female" salary="10000"/>

<emps id="Num 8" name="Also" age="32" gender="male" salary="50000"></emps> -->

<**emps** id=*'1'*>

<**name**>Kamui</**name**>

<**age**>25</**age**>

<**gender**>male</**gender**>

<**salary**>5000</**salary**>

</**emps**>

<**emps** id=*'2'*>

<**name**>Subaru</**name**>

<**age**>28</**age**>

<**gender**>male</**gender**>

<**salary**>8000</**salary**>

</**emps**>

<**emps** id=*'3'*>

<**name**>Luna</**name**>

<**age**>17</**age**>

<**gender**>female</**gender**>

<**salary**>20000</**salary**>

</**emps**>

</**list**>

ParseXmlDemo.java

package xml;

import java.io.File;

import java.util.ArrayList;

import java.util.List;

import org.dom4j.Document;

import org.dom4j.Element;

import org.dom4j.io.SAXReader;

/\*\*

\* 使用dom4j解析xml文档

\* **@author** Administrator

\*

\*/

public class ParseXmlDemo {

public static void main(String[] args) {

/\*

\* 使用DOM4J解析XML的大致步骤：

\* 1.创建SAXReader

\* 2.使用SAXReader读取XML文档，并生成Document对象

\* 3.通过Document对象获取根元素

\* 4.从根元素开始逐级获取子元素以达到便利XML文档数据的目的

\*/

/\*

\* 将emplist.xml文档中的所有员工信息读取

\*/

List<Emp> empList = new ArrayList<>();

try {

//1

SAXReader reader = new SAXReader();

//2

Document doc = reader.read(new File("emplist.xml"));

//3

/\*

\* Document提供了获取根元素的方法：

\* Element getRootElement()

\*

\* Element的每一个实例用于表示XML文档中的一个元素（标签）

\*

\* 其提供了获取标签相关信息的一组方法：

\* String getName()

\* 获取标签的名字

\*

\* String getText()

\* 获取标签文本(开始与结束标签中间的文本)

\*

\* Element element(String name)

\* 获取当前标签下指定名字的子标签

\*

\* List elements()

\* 获取当前标签下所有子标签

\*

\* List elements(String name)

\* 获取当前标签下所有同名字标签(指定的名字)

\*/

Element root = doc.getRootElement();

// String name = root.getName();

// System.out.println(name);

//获取根标签下所有的<emp>标签

List<Element> list = root.elements("emps");

System.***out***.println(list.size());

/\*

\* 遍历每个<emp>标签，用于获取该员工信息

\*/

for(Element empEle : list) {

//获取员工名字

//1 获取<name>标签

Element nameEle = empEle.element("name");

//2获取<name>标签中间的文本

String name = nameEle.getText().trim();

//获取年龄

Element ageEle = empEle.element("age");

int age = Integer.*parseInt*(ageEle.getText());

//获取性别

String gender = empEle.elementText("gender");

//获取工资

int salary = Integer.*parseInt*(empEle.elementText("salary"));

//获取ID--属性

int id = Integer.*parseInt*(empEle.attributeValue("id"));

Emp emp = new Emp(id,name,age,gender,salary);

empList.add(emp);

}//遍历完毕

System.***out***.println("解析完毕！");

for (Emp emp : empList) {

System.***out***.println(emp);

}

}catch (Exception e) {

e.printStackTrace();

}

}

}

**15:45实训室2-正课**

关于如何写出xml文档：

package xml;

import java.io.FileOutputStream;

import java.util.ArrayList;

import java.util.List;

import org.dom4j.Document;

import org.dom4j.DocumentHelper;

import org.dom4j.Element;

import org.dom4j.io.OutputFormat;

import org.dom4j.io.XMLWriter;

/\*\*

\* 写出xml文档

\* **@author** Administrator

\*

\*/

public class WriteXmlDemo {

public static void main(String[] args) {

List<Emp> empList = new ArrayList<>();

empList.add(new Emp(1,"Kamui",25,"男",5000));

empList.add(new Emp(2,"Subaru",26,"男",6000));

empList.add(new Emp(1,"Luna",17,"femal",7000));

empList.add(new Emp(1,"Diva",20,"femal",8000));

empList.add(new Emp(1,"Also",28,"male",9000));

/\*

\* 大致步骤：

\* 1.创建Document对象表示一个空白文档

\* 2.向Document中添加根元素

\* 3.从根元素开始逐级添加子元素以形成XML文档的树结构

\* 4.创建XmlWriter

\* 5.通过XmlWriter将Document写出以生成XML

\*/

//1

Document doc = DocumentHelper.*createDocument*();

//2

/\*

\* Document提供的方法：

\* Element addElement(String name)

\* 向当前文档中添加给定名字的根元素，并以Element实例形式将其返回，以便继续对根元素追加操作。

\* 注意，Document的这个方法只能调用一次，

\* 因为一个文档只能有一个根元素。

\*/

Element root = doc.addElement("list");

for (Emp emp : empList) {

//像根元素中添加<emp>

Element empEle = root.addElement("emp");

//添加名字

Element nameEle = empEle.addElement("name");

nameEle.addText(emp.getName());

//添加年龄

empEle.addElement("age").addText(emp.getAge()+"");

//添加性别

empEle.addElement("grender").addText(emp.getGender());

//添加工资

empEle.addElement("salary").addText(emp.getGender());

//添加id--属性

empEle.addAttribute("id", emp.getId()+"");

}

XMLWriter writer = null;

try {

writer = new XMLWriter(new FileOutputStream("myemp.xml"),OutputFormat.*createPrettyPrint*());

writer.write(doc);

System.***out***.println("写出完毕！");

} catch (Exception e) {

e.printStackTrace();

}finally {

if(writer!=null) {

try {

writer.close();

} catch (Exception e2) {

e2.printStackTrace();

}

}

}

}

}

**16:57实训室2-正课**

那么在了解了XML相关的一些功能与知识之后，我们再重新回到WebServer\_v9的课程项目中来，做这么一个事情：

/\*

\* 使用DOM4J解析conf/web.xml文件

\* 将根标签下所有名为<mime-mapping>的子标签

\* 获取回来，并将其中的子标签：

\* <extension>中间的文本作为key

\* <mime-type>中间的文本作为value

\* 来初始化mimeMapping这个Map

\*

\* 初始化完毕后，这个Map应当有1000多个元素。

\*/

这个代码自行完成。

**17:28实训室2-正课**

我的代码：

package com.webserver.http;

import java.io.File;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import org.dom4j.Document;

import org.dom4j.DocumentException;

import org.dom4j.Element;

import org.dom4j.io.SAXReader;

/\*\*

\* HTTP协议规定的相关内容

\* **@author** Administrator

\*

\*/

public class HttpContext {

/\*

\* 资源后缀名与Content-Type响应头对应值的映射关系

\* key:资源后缀名

\* value:Content-Type对应的值

\*/

private static Map<String,String> *mimeMapping* = new HashMap<>();

//初始化所有静态资源

static {

*initMimeMapping*();

}

private static void initMimeMapping() {

// mimeMapping.put("html", "text/html");

// mimeMapping.put("css", "text/css");

// mimeMapping.put("js", "application/javascript");

// mimeMapping.put("png", "image/png");

// mimeMapping.put("gif", "image/gif");

// mimeMapping.put("jpg", "image/jpeg");

/\*

\* 使用DOM4J解析conf/web.xml文件

\* 将根标签下所有名为<mime-mapping>的子标签

\* 获取回来，并将其中的子标签：

\* <extension>中间的文本作为key

\* <mime-type>中间的文本作为value

\* 来初始化mimeMapping这个Map

\*

\* 初始化完毕后，这个Map应当有1000多个元素。

\*/

try {

SAXReader reader = new SAXReader(); //创建SAXReader对象

Document doc = reader.read(new File("./conf/web.xml"));//读取Document的结果付给doc对象

Element root = doc.getRootElement();//获取根元素

List<Element> list = root.elements("mime-mapping");//获取所有子标签

for (Element element : list) {//将取得的值放入map集合

String key = element.elementText("extension");

String value = element.elementText("mime-type");

*mimeMapping*.put(key, value);

}

} catch (DocumentException e) {

e.printStackTrace();

}

System.***out***.println(*mimeMapping*.size());

}

/\*\*

\* 根据给定的资源后缀名获取对应的Content-Type值

\* **@param** ext

\* **@return**

\*/

public static String getMimeType(String ext) {

return *mimeMapping*.get(ext);

}

public static void main(String[] args) {

String type = HttpContext.*getMimeType*("css");

System.***out***.println(type);// text/css

}

}

**17:35实训室2-正课**

今天的课程就到这里了，休息一下，吃饭去吧。

**20:10实训室2-晚自习**

项目经理表示，这个月24号月考，提前做好准备。

然后开讲解了一大堆基础练习题，并强调太多的人基础太差，然而我尝试着做了一下练习，结果我也有好多题目做错了……

结果今天的晚自习没有写什么代码，尽是在看群里的大佬们商业互吹了。

一方面我的同桌已经开始放弃治疗了，他可能真的要留级了。

**20:30实训室2-晚自习**

今天可能真的是我的惰性逆袭了，教室环境也很燥热，只想赶紧回去洗澡睡觉。

空气不流通，真是难受，汗止不住的流。

**20:45实训室2-晚自习**

今天的这个XML的内容还挺麻烦的，但不是很复杂，理解起来也还算是能够接受。就是课程进度能够明显的感觉到快了不少……或者说是我的理解能力下降了不少？总感觉自己的反应慢半拍。

关于考证，结果我还是一门都没有选，其实也不想考。不想像当初自学日语的时候一样，无端的给自己太多考试的压力。也并不想通过强迫自己的方式来达到努力学习的效果，而且往往强迫自己学也无法达到理想的效果。嗯……这听上去可真像是个借口。

我想玩游戏了……

**21:10实训室2-晚自习**

最近让我分心的事情越来越多了，终究还是逃离不了来自熟人、亲戚投过来的偏见。我这才消失2个月就已经开始肆无忌惮的嘴臭我了。只在自己有事需要找我帮忙的时候才会想起我……我这的是活的有多可恶啊，合着我就是一个five石锤了。

今日感想总结：我想专心写写代码，怎么就这么难……

今日评分：6/10分

今天就到这里，明天继续投稿。

感谢支持！

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

**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