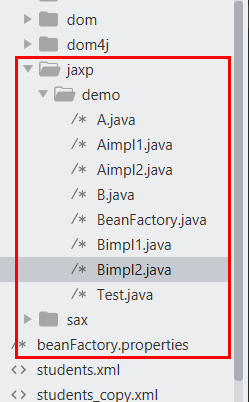
# XML

|  |
| --- |
| xml(eXtendsible markup language)：可扩展的标记语言  **XML 有什么用?**  1.可以用来保存数据  2.可以用来做配置文件  3.数据传输载体  **文档声明**  简单声明， version : 解析这个xml的时候，使用什么版本的解析器解析  <?xml version="1.0" ?>  encoding : 解析xml中的文字的时候，使用什么编码来翻译  <?xml version="1.0" encoding="gbk" ?>  standalone : no - 该文档会依赖关联其他文档 ， yes-- 这是一个独立的文档  <?xml version="1.0" encoding="gbk" standalone="no" ?>  **encoding详解**  在解析这个xml的时候，使用什么编码去解析。 ---解码。  文字， 而是存储这些文字对应的二进制 。 那么这些文字对应的二进制到底是多少呢？ 根据文件使用的编码 来得到。  -----------------------------------------------  **txt默认文件保存的时候，使用的是GBK的编码保存。**  所以要想让我们的xml能够正常的显示中文，有两种解决办法  1.让encoding也是GBK 或者 gb2312 .  2.如果encoding是 utf-8 ， 那么保存文件的时候也必须使用utf-8  3.保存的时候见到的ANSI 对应的其实是我们的本地编码 GBK。  4.为了通用，建议使用UTF-8编码保存，以及encoding 都是 utf-8  -----------------------------------------------  XML解析方式(面试常问)  有很多种，但是常用的有两种。  DOM、SAX  针对这两种解析方式的API  一些组织或者公司， 针对以上两种解析方式， 给出的解决方案有哪些？  jaxp sun公司。 比较繁琐  jdom  dom4j 使用比较广泛 |

# Jaxp



|  |
| --- |
| A.java  package cn.itcast.jaxp.demo;  public interface A {  *void* show();  }  B.java  package cn.itcast.jaxp.demo;  public interface B {  *void* show();  } |

|  |
| --- |
| Aimpl1.java  package cn.itcast.jaxp.demo;  public class Aimpl1 implements *A* {    @*Override*    public *void* show() {        System.out.println("这是第一个实现类!");    }  }  Aimpl2.java  package cn.itcast.jaxp.demo;  public class Aimpl2 implements *A* {    @*Override*    public *void* show() {        System.out.println("这是第二个实现类!");    }  } |

|  |
| --- |
| Bimpl1.java  package cn.itcast.jaxp.demo;  public class Bimpl1 implements *B* {    @*Override*    public *void* show() {        System.out.println("这是第一个实现类!");    }  }  Bimpl2.java  package cn.itcast.jaxp.demo;  public class Bimpl2 implements *B* {    @*Override*    public *void* show() {        System.out.println("这是第二个实现类!");    }  } |

|  |
| --- |
| BeanFactory.java  package cn.itcast.jaxp.demo;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.IOException;  import java.util.Properties;  public class BeanFactory {    private static  *Properties* prop = new Properties();    static{      //1.读取配置文件        try {          prop.load(new FileInputStream("src/beanFactory.properties"));        } catch (*FileNotFoundException* *e*) {          e.printStackTrace();        } catch (*IOException* *e*) {          e.printStackTrace();        }    }      public static <*T*> *T* getBean(*Class*<*T*> *c*){      //1.读取配置文件      try {        //2.取出用哪个实现类  *String* className  = prop.getProperty(c.getSimpleName());        //3.根据完整类名创建实例并返回  *Class*<*T*> clazz = (*Class*<T>) Class.forName(className);        return clazz.newInstance();      } catch (*Exception* *e*) {        // TODO Auto-generated catch block        e.printStackTrace();      }        return null;    }  } |

|  |
| --- |
| Test.java  package cn.itcast.jaxp.demo;  public class Test {    public static *void* main(*String*[] *args*) {      /\*A a = BeanFactory.getBean(A.class);      a.show();\*/  *B* b = BeanFactory.getBean(B.class);      b.show();    }  } |

# students.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <students>    <student number="itcast\_0001">      <name>tom</name>      <age>18</age>      <sex>male</sex>    </student>    <student number="itcast\_0002">      <name>jerry</name>      <age>16</age>      <sex>female</sex>    </student>  </students> |

# dom

|  |
| --- |
| package cn.shipeng.dom;  import java.io.File;  import javax.xml.parsers.DocumentBuilder;  import javax.xml.parsers.DocumentBuilderFactory;  import org.w3c.dom.Document;  import org.w3c.dom.Element;  import org.w3c.dom.Node;  import org.w3c.dom.NodeList;  public class Demo01 {    //获得所有学生信息    public static *void* main(*String*[] *args*) {      //1 获得jaxp工厂  *DocumentBuilderFactory* factory =  DocumentBuilderFactory.newInstance();      //2 通过工厂获得解析器实现类      try {  *DocumentBuilder* builder = factory.newDocumentBuilder();        //3 使用解析器加载xml文档 ==> document  *Document*  doc = builder.parse(new File("src/students.xml"));        //4 获得所有学生元素的集合  *NodeList* studentList = doc.getElementsByTagName("student");        //5 遍历集合          for(*int* i = 0 ; i < studentList.getLength() ; i++){  *Element* stuEle = (Element) studentList.item(i);            //获得学生元素的number属性  *String* number = stuEle.getAttribute("number");            System.out.println("学生的学号是:"+number);            //获得学生节点下的所有子节点(包括文本在内一共7个)  *NodeList* children = stuEle.getChildNodes();              //遍历集合并提取出我们想要的name age sex元素对象            for(*int* x = 0 ; x < children.getLength() ; x++){  *Node* node = children.item(x);              /\*方式1:               \* if(node instanceof Element){                }\*/              /\*               \* 方式2:               \*/              if(node.getNodeType() == Node.ELEMENT\_NODE){  *Element* child = (Element) node;                if(child.getNodeName().equals("name")){  *String* name = child.getTextContent();                  System.out.println("学生的姓名是:"+name);                }else if(child.getNodeName().equals("age")){  *String* age = child.getTextContent();                  System.out.println("学生的年龄是:"+age);                }else if(child.getNodeName().equals("sex")){  *String* sex = child.getTextContent();                  System.out.println("学生的性别是:"+sex);                }              }            }          }        } catch (*Exception* *e*) {        e.printStackTrace();      }    }  } |

# sax

|  |
| --- |
| package cn.itcast.sax;  import java.io.File;  import javax.xml.parsers.ParserConfigurationException;  import javax.xml.parsers.SAXParser;  import javax.xml.parsers.SAXParserFactory;  import org.xml.sax.SAXException;  public class Demo1 {    public static *void* main(*String*[] *args*) {      //1 获得解析器工厂类  *SAXParserFactory* factory = SAXParserFactory.newInstance();      //2 获得解析器      try {  *SAXParser* parser = factory.newSAXParser();        //3 解析文档        parser.parse(new File("src/students.xml"),new MyHandler2("itcast\_0002"));      } catch (*Exception* *e*) {        e.printStackTrace();      }    }  } |

|  |
| --- |
| package cn.itcast.sax;  import org.xml.sax.Attributes;  import org.xml.sax.SAXException;  import org.xml.sax.helpers.DefaultHandler;  //获得所有学生信息 ,打印  public class MyHandler extends *DefaultHandler* {    @*Override*    public *void* startDocument() throws *SAXException* {      System.out.println("文档开始解析啦!");    }    @*Override*    //sax 1.0版本中前两个参数没有用,永远是null.    //参数3 告知开发者当前触发的元素是哪个元素.    //参数4 将标签上的属性 提供给你    public *void* startElement(*String* *uri*, *String* *localName*, *String* *qName*,  *Attributes* *attributes*) throws *SAXException* {      System.out.println(qName+"元素开始啦!");      //判断当前遇到的元素是否是student元素.      if(qName.equals("student")){  *String* number = attributes.getValue("number");        System.out.println("学生的学号是:"+number);      }        //如果是==> 获得number属性    }      @*Override*    public *void* characters(*char*[] *ch*, *int* *start*, *int* *length*)        throws *SAXException* {  *String* str = new String(ch, start, length);      str = str.trim();//去掉两端的空白字符 回车 空格 制表符      if(str.length() >0){        System.out.println(str);      }    }    @*Override*    public *void* endElement(*String* *uri*, *String* *localName*, *String* *qName*)        throws *SAXException* {      System.out.println(qName+"元素结束啦!");    }    @*Override*    public *void* endDocument() throws *SAXException* {      System.out.println("文档结束解析啦!");    }  } |

|  |
| --- |
| package cn.itcast.sax;  import org.xml.sax.Attributes;  import org.xml.sax.SAXException;  import org.xml.sax.helpers.DefaultHandler;  //获得所有学生信息 ,打印  public class MyHandler2 extends *DefaultHandler* {    private *String* number;      public MyHandler2(*String* *number*) {      super();      this.number = number;    }    private *boolean* flag = false;    private *String* current = "";    @*Override*    public *void* startElement(*String* *uri*, *String* *localName*, *String* *qName*,  *Attributes* *attributes*) throws *SAXException* {      current = qName;      if(qName.equals("student")&&attributes.getValue("number").equals(number)){        flag = true;      }    }      @*Override*    public *void* characters(*char*[] *ch*, *int* *start*, *int* *length*)        throws *SAXException* {  *String* str = new String(ch,start,length);      str = str.trim();      if(flag && str.length()>0){        if(current.equals("name")){          System.out.println("学生的姓名是:"+str);        }else if(current.equals("age")){          System.out.println("学生的年龄是:"+str);        }else if(current.equals("sex")){          System.out.println("学生的性别是:"+str);        }      }    }    @*Override*    public *void* endElement(*String* *uri*, *String* *localName*, *String* *qName*)        throws *SAXException* {      if(qName.equals("student")){        flag = false;      }    }  } |

# dom4j

查询

|  |
| --- |
| package cn.shipeng.dom;  import java.io.File;  import java.util.List;  import org.dom4j.Document;  import org.dom4j.Element;  import org.dom4j.io.SAXReader;  // 查询出所有学生的所有信息  public class Demo01 {    public static *void* main(*String*[] *args*) {      try {  *SAXReader* reader = new SAXReader();  *Document* document = reader.read(new File("src/students.xml"));        //1.获得根元素  *Element* root = document.getRootElement();        //2.迭代根元素下的所有名叫student的子元素  *List*<*Element*> list = root.elements("student");        for(*Element* student : list){          //3.获得student元素的number属性  *String* number = student.attributeValue("number");          //4.student子元素的内容(name age sex)  *String* name = student.elementText("name");  *String* age = student.elementText("age");  *String* sex = student.elementText("sex");            System.out.println("当前学生的学号是"+number+",姓名是:"+name+",年龄是:"+age+",性别是"+sex+"");        }          /\*        //2.迭代根元素下的所有名叫student的子元素        for(Iterator<Element> it = root.elementIterator("student");it.hasNext();){          Element student = it.next();          //3.获得student元素的number属性          String number = student.attributeValue("number");          //4.student子元素的内容(name age sex)          String name = student.elementText("name");          String age = student.elementText("age");          String sex = student.elementText("sex");            System.out.println("当前学生的学号是"+number+",姓名是:"+name+",年龄是:"+age+",性别是"+sex+"");        }  \*/      } catch (*Exception* *e*) {        e.printStackTrace();      }    }  } |

增加

|  |
| --- |
| package cn.shipeng.dom;  import java.io.File;  import java.io.FileOutputStream;  import java.io.PrintWriter;  import org.dom4j.Document;  import org.dom4j.Element;  import org.dom4j.io.OutputFormat;  import org.dom4j.io.SAXReader;  import org.dom4j.io.XMLWriter;  //增加一个学生元素. jack itcast\_0003 19 male  public class Demo01 {    public static *void* main(*String*[] *args*) {  *SAXReader* reader = new SAXReader();      try {  *Document* document = reader.read(new File("src/students.xml"));          //1 获得根元素  *Element* root = document.getRootElement();        //2 添加Element,添加number属性,  *Element* studentEle = root.addElement("student").addAttribute("number", "itcast\_0003");        //3 添加name age sex 子元素并添加子元素中的文本        studentEle.addElement("name").addText("张三e");        studentEle.addElement("age").addText("19");        studentEle.addElement("sex").addText("male");        //4 将document对象写到文件中.        //创建格式化器  *OutputFormat* format = OutputFormat.createPrettyPrint();        format.setEncoding("UTF-8");        //创建写入器        //XMLWriter writer = new XMLWriter(new FileWriter("src/students\_copy.xml"),format);        //XMLWriter writer = new XMLWriter(new PrintWriter("src/students\_copy.xml", "GBK"),format);  *XMLWriter* writer = new XMLWriter(new FileOutputStream("src/students\_copy.xml"),format);        //写入        writer.write(document);        //关闭资源        writer.close();      } catch (*Exception* *e*) {        // TODO Auto-generated catch block        e.printStackTrace();      }    }  } |

删除、修改

|  |
| --- |
| package cn.shipeng.dom;  import java.io.File;  import java.io.FileOutputStream;  import org.dom4j.Document;  import org.dom4j.Element;  import org.dom4j.io.OutputFormat;  import org.dom4j.io.SAXReader;  import org.dom4j.io.XMLWriter;  public class Demo01 {      public static *void* main(*String*[] *args*) {  *SAXReader* reader = new SAXReader();      try {  *Document* document = reader.read(new File("src/students.xml"));        //1.定义xpath表达式  *String* xpath ="//student[@number='itcast\_0001']";          //2.使用xpath查找  *Element* studentEle = (Element) document.selectSingleNode(xpath);          //3.修改student元素的子元素的name ,age .sex内容        studentEle.element("name").setText("rose");        studentEle.element("age").setText("16");        studentEle.element("sex").setText("female");          //删除        //System.out.println(studentEle.getParent().remove(student));          //回写  *XMLWriter* writer = new XMLWriter(new FileOutputStream("src/students\_copy.xml"), OutputFormat.createPrettyPrint());          writer.write(document);          writer.close();      } catch (*Exception* *e*) {        // TODO Auto-generated catch block        e.printStackTrace();      }    }  } |

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

|  |
| --- |
| element.element("stu") : 返回该元素下的第一个stu元素  element.elements(); 返回该元素下的所有子元素。  1创建SaxReader对象  2.指定解析的xml  3.获取根元素。  4.根据根元素获取子元素或者下面的子孙元素  try {    //1. 创建sax读取对象  *SAXReader* reader = new SAXReader(); //jdbc -- classloader    //2. 指定解析的xml源  *Document*  document  = reader.read(new File("src/xml/stus.xml"));  //3. 得到元素、  //得到根元素  *Element* rootElement= document.getRootElement();  //获取根元素下面的子元素 age  //rootElement.element("age")    //System.out.println(rootElement.element("stu").element("age").getText());  //获取根元素下面的所有子元素 。 stu元素  List&lt;Element&gt; elements = rootElement.elements();  //遍历所有的stu元素  for (*Element* element : elements) {    //获取stu元素下面的name元素  *String* name = element.element("name").getText();  *String* age = element.element("age").getText();  *String* address = element.element("address").getText();    System.out.println("name="+name+"==age+"+age+"==address="+address);  }  } catch (*Exception* *e*) {    e.printStackTrace();  } |

# Homework

|  |
| --- |
| StudentDao.java  package cn.itcast.dom4j.homework;  import java.util.List;  //Dao ==> Data Access Object  public interface StudentDao {  //增  void saveStudent(Student stu);  //删  void deleteStudent(String number);  void deleteStudent(Student stu);  //改  void updateStudent(Student stu);  //查  Student getStudentByNumber(String number);  List<Student> getAllStudent();  } |

|  |
| --- |
| Student.java  package cn.itcast.dom4j.homework;  public class Student {  private String number;  private String name;  private String sex;  private int age;    public Student() {  super();  // TODO Auto-generated constructor stub  }    public Student(String number, String name, String sex, int age) {  super();  this.number = number;  this.name = name;  this.sex = sex;  this.age = age;  }  public String getNumber() {  return number;  }  public void setNumber(String number) {  this.number = number;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getSex() {  return sex;  }  public void setSex(String sex) {  this.sex = sex;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  } |

|  |
| --- |
| StudentDaoImpl.java  package cn.itcast.dom4j.homework;  import java.io.File;  import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.UnsupportedEncodingException;  import java.util.List;  import org.dom4j.Document;  import org.dom4j.DocumentException;  import org.dom4j.Element;  import org.dom4j.io.OutputFormat;  import org.dom4j.io.SAXReader;  import org.dom4j.io.XMLWriter;  public class StudentDaoImpl implements StudentDao {  @Override  public void saveStudent(Student stu) {  try {  Document doc = getDocument();  Element root = doc.getRootElement();  //4.在根元素上添加student子元素,添加属性  Element studentEle = root.addElement("student").addAttribute("number", stu.getNumber());  //5. 在student上添加name,age,sex子元素并添加内容  studentEle.addElement("name").setText(stu.getName());  studentEle.addElement("age").setText(stu.getAge()+"");  studentEle.addElement("sex").setText(stu.getSex());  save2File(doc);  } catch (Exception e) {  e.printStackTrace();  }      }  private void save2File(Document doc) throws UnsupportedEncodingException,  FileNotFoundException, IOException {  //6.回写  //1>创建OutputFormat  OutputFormat format = OutputFormat.createPrettyPrint();  format.setEncoding("utf-8");  //2>创建xmlwriter  XMLWriter writer = new XMLWriter(new FileOutputStream("src/students\_copy.xml"), format);  //3>写入  writer.write(doc);  //4>释放资源  writer.close();  }  private Document getDocument() throws DocumentException {  //1.创建解析器  SAXReader reader = new SAXReader();  //2.加载xml文件 ==> doc  Document doc = reader.read(new File("src/students\_copy.xml"));  //3.获得根元素  return doc;  }  @Override  public void deleteStudent(String number) {  }  @Override  public void deleteStudent(Student stu) {  }  @Override  public void updateStudent(Student stu) {  }  @Override  public Student getStudentByNumber(String number) {  return null;  }  @Override  public List<Student> getAllStudent() {  return null;  }  } |

# XML约束

|  |
| --- |
| 如下的文档， 属性的ID值是一样的。 这在生活中是不可能出现的。 并且第二个学生的姓名有好几个。 一般也很少。那么怎么规定ID的值唯一， 或者是元素只能出现一次，不能出现多次？ 甚至是规定里面只能出现具体的元素名字。  <stus>  <stu id="10086">  <name>张三</name>  <age>18</age>  <address>深圳</address>  </stu>  <stu id="10086">  <name>李四</name>  <name>李五</name>  <name>李六</name>  <age>28</age>  <address>北京</address>  </stu>  </stus> |

# DTD

|  |
| --- |
| ###*DTD*    语法自成一派， 早起就出现的。 可读性比较差。  1. 引入网络上的DTD      <!-- 引入dtd 来约束这个xml -->      <!--    文档类型  根标签名字 网络上的dtd   dtd的名称   dtd的路径  <!*DOCTYPE* stus PUBLIC "//UNKNOWN/" "unknown.dtd"> -->     2. 引入本地的DTD        <!-- 引入本地的DTD  ： 根标签名字 引入本地的DTD  dtd的位置 -->        <!-- <!*DOCTYPE* stus SYSTEM "stus.dtd"> -->  2. 直接在XML里面嵌入DTD的约束规则      <!-- xml文档里面直接嵌入DTD的约束法则 -->      <!*DOCTYPE* stus [        <!*ELEMENT* stus (stu)>        <!*ELEMENT* stu (name,age)>        <!*ELEMENT* name (#PCDATA)>        <!*ELEMENT* age (#PCDATA)>      ]>        <stus>        <stu>          <name>张三</name>          <age>18</age>        </stu>      </stus>      <!*ELEMENT* stus (stu)>  : stus 下面有一个元素 stu  ， 但是只有一个      <!*ELEMENT* stu (name , age)>  stu下面有两个元素 name  ,age  顺序必须name-age      <!*ELEMENT* name (#PCDATA)>      <!*ELEMENT* age (#PCDATA)>  <!*ATTLIST* stu id CDATA #IMPLIED> stu有一个属性 文本类型， 该属性可有可无      元素的个数：        ＋　一个或多个        \*  零个或多个        ? 零个或一个        属性的类型定义          CDATA : 属性是普通文字        ID : 属性的值必须唯一      <!*ELEMENT* stu (name , age)>   按照顺序来      <!*ELEMENT* stu (name | age)>   两个中只能包含一个子元素  ###*Schema*    其实就是一个xml ， 使用xml的语法规则， xml解析器解析起来比较方便 ， 是为了替代DTD 。    但是Schema 约束文本内容比DTD的内容还要多。 所以目前也没有真正意义上的替代DTD    约束文档：      <!-- xmlns  :  xml namespace : 名称空间 /  命名空间      targetNamespace :  目标名称空间 。 下面定义的那些元素都与这个名称空间绑定上。      elementFormDefault ： 元素的格式化情况。  -->      <schema xmlns="http://www.w3.org/2001/XMLSchema"        targetNamespace="http://www.itheima.com/teacher"        elementFormDefault="qualified">          <element name="teachers">          <complexType>            <sequence maxOccurs="unbounded">              <!-- 这是一个复杂元素 -->              <element name="teacher">                <complexType>                  <sequence>                    <!-- 以下两个是简单元素 -->                    <element name="name" type="string"></element>                    <element name="age" type="int"></element>                  </sequence>                </complexType>              </element>            </sequence>          </complexType>        </element>      </schema>      实例文档：      <?xml version="1.0" encoding="UTF-8"?>      <!-- xmlns:xsi : 这里必须是这样的写法，也就是这个值已经固定了。      xmlns : 这里是名称空间，也固定了，写的是schema里面的顶部目标名称空间      xsi:schemaLocation : 有两段： 前半段是名称空间，也是目标空间的值 ， 后面是约束文档的路径。       -->      <teachers        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"        xmlns="http://www.itheima.com/teacher"        xsi:schemaLocation="http://www.itheima.com/teacher teacher.xsd"      >        <teacher>          <name>zhangsan</name>          <age>19</age>        </teacher>        <teacher>          <name>lisi</name>          <age>29</age>        </teacher>        <teacher>          <name>lisi</name>          <age>29</age>        </teacher>  </teachers>  ##名称空间的作用  一个xml如果想指定它的约束规则， 假设使用的是DTD ，那么这个xml只能指定一个DTD  ，  不能指定多个DTD 。 但是如果一个xml的约束是定义在schema里面，并且是多个schema，那么是可以的。简单的说： 一个xml 可以引用多个schema约束。 但是只能引用一个DTD约束。  名称空间的作用就是在 写元素的时候，可以指定该元素使用的是哪一套约束规则。  默认情况下 ，如果只有一套规则，那么都可以这么写    <name>张三</name>    <aa:name></aa:name>    <bb:name></bb:name> |

# 03\_ServletContext 获取全局参数\_\_rec

|  |
| --- |
| package com.itheima.servlet;  import java.io.IOException;  import javax.servlet.ServletContext;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \* Servlet implementation class ServletContext02  \*/  public class ServletContext02 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {    //1. 获取对象  ServletContext context = getServletContext();  String address = context.getInitParameter("address");  System.out.println("这是02获取的数据：：：address="+address);  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  } |

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" id="WebApp\_ID" version="2.5">  <display-name>SerlvetContextDemo</display-name>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  <welcome-file>index.htm</welcome-file>  <welcome-file>index.jsp</welcome-file>  <welcome-file>default.html</welcome-file>  <welcome-file>default.htm</welcome-file>  <welcome-file>default.jsp</welcome-file>  </welcome-file-list>    <!-- 全局参数 ： 哪个servlet都可以拿 ， ServletContext -->  <context-param>  <param-name>address</param-name>  <param-value>深圳宝安</param-value>  </context-param>  <servlet>  <description></description>  <display-name>ServletContext01</display-name>  <servlet-name>ServletContext01</servlet-name>  <servlet-class>com.itheima.servlet.ServletContext01</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>ServletContext01</servlet-name>  <url-pattern>/ServletContext01</url-pattern>  </servlet-mapping>  <servlet>  <description></description>  <display-name>ServletContext02</display-name>  <servlet-name>ServletContext02</servlet-name>  <servlet-class>com.itheima.servlet.ServletContext02</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>ServletContext02</servlet-name>  <url-pattern>/ServletContext02</url-pattern>  </servlet-mapping>  </web-app> |

# 04\_ServletContext 普通方式读取工程文件\_\_rec

|  |
| --- |
| package com.itheima.servlet;  import java.io.FileInputStream;  import java.io.IOException;  import java.io.InputStream;  import java.util.Properties;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \* Servlet implementation class Demo02  \*/  public class Demo02 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {    //1. 创建属性对象  Properties properties = new Properties();    //2. 指定载入的数据源  /\*  \* 此处，如果想获取web工程下的资源，用普通的FileInputStream 写法是不OK 的。  \* 因为路径不对了。 这里相对的路径，其实是根据jre来确定的。 但是我们这是一个web工程，  \* jre 后面会由tomcat管理，所以这里真正相对的路径是 tomcat里面的bin目录  \*/  InputStream is = new FileInputStream("classes/config.properties");  properties.load(is);    //3. 获取name属性的值  String name = properties.getProperty("name");    System.out.println("name="+name);  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  } |

|  |
| --- |
| config.properties文件  name=zhangsan |

# 05\_ServletContext 获取资源文件（一）\_\_rec

|  |
| --- |
| package com.itheima.servlet;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.IOException;  import java.io.InputStream;  import java.util.Properties;  import javax.servlet.ServletContext;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \* Servlet implementation class Demo03  \*/  public class Demo03 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  /\*test01();\*/  //test02();  test03();    }  /\*\*  \* 根据classloader去获取工程下的资源 类加载器（JDBC）  \*/  private void test03() {  try {  // 1. 创建属性对象  Properties properties = new Properties();    //获取该java文件的class ，然后获取到加载这个class到虚拟机中的那个类加载器对象。    /\*  \* ServletContext  \* a路径--工程在tomcat里面的目录  \* D:\tomcat\apache-tomcat-7.0.52\apache-tomcat-7.0.52\wtpwebapps\Demo03  \*  \* ClassLoader  \*  \* a路径： D:\tomcat\apache-tomcat-7.0.52\apache-tomcat-7.0.52\wtpwebapps\Demo03\WEB-INF\classes  \*  \* 默认的lcassloader 的路径是上面这个路径，我们必须得回到Demo03这个目录下，才能进入file目录。如何回到上一级目录呢？  \* ../../ --- D:\tomcat\apache-tomcat-7.0.52\apache-tomcat-7.0.52\wtpwebapps\Demo03  \* ../../file/config.properties --- D:\tomcat\apache-tomcat-7.0.52\apache-tomcat-7.0.52\wtpwebapps\Demo03\file\config.properties  \* b路径： D:\tomcat\apache-tomcat-7.0.52\apache-tomcat-7.0.52\wtpwebapps\Demo03\file\config.properties  \*/  // System.out.println(getClass().getClassLoader());  InputStream is = this.getClass().getClassLoader().getResourceAsStream("../../file/config.properties");  properties.load(is);  // 3. 获取name属性的值  String name = properties.getProperty("name");  System.out.println("name333333=" + name);  is.close();  } catch (Exception e) {  e.printStackTrace();  }  }  /\*\*  \* 根据相对路径，直接获取流对象  \*/  private void test02() {  try {  // 获取ServletContext对象  ServletContext context = getServletContext();    // 1. 创建属性对象  Properties properties = new Properties();    //获取web工程下的资源，转化成流对象。 前面隐藏当前工程的根目录。  /\*  \* 相对路径 （有参照物） 相对谁?  \*  \* 工程在tomcat里面的根目录。  \*  \* a路径--工程在tomcat里面的目录  \* D:\tomcat\apache-tomcat-7.0.52\apache-tomcat-7.0.52\wtpwebapps\Demo03  \* b路径---  \* file\config.properties  \*  \* D:\tomcat\apache-tomcat-7.0.52\apache-tomcat-7.0.52\wtpwebapps\Demo03\file\config.properties  \*  \*  \* 绝对路径 (没有参照物)  \*  \* D:\tomcat\apache-tomcat-7.0.52\apache-tomcat-7.0.52\wtpwebapps\Demo03\file\config.properties  \*  \*/  InputStream is = context.getResourceAsStream("file/config.properties");  properties.load(is);  // 3. 获取name属性的值  String name = properties.getProperty("name");  System.out.println("name22=" + name);  is.close();  } catch (Exception e) {  e.printStackTrace();  }  }  /\*\*  \* 先获取路径，在获取流对象  \* @throws FileNotFoundException  \* @throws IOException  \*/  private void test01() throws FileNotFoundException, IOException {  // 获取ServletContext对象  ServletContext context = getServletContext();  //获取给定的文件在服务器上面的绝对路径。  String path = context.getRealPath("file/config.properties");  System.out.println("path="+path);    // 1. 创建属性对象  Properties properties = new Properties();  InputStream is = new FileInputStream(path);  properties.load(is);  // 3. 获取name属性的值  String name = properties.getProperty("name");  System.out.println("name=" + name);  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  } |

|  |
| --- |
| config.properties文件  name=zhangsan |

# 07\_ServletContext 使用ClassLoader获取资源文件\_\_rec

|  |
| --- |
| 同上 |

# 08\_ServletContext 获取登录成功总数（分析）\_\_rec

|  |
| --- |
| login.html  <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>Insert title here</title>  </head>  <body>  <h2>请输入以下内容，完成登录</h2>  <!--  A路径： Servlet的路径  http://localhost:8080/Demo4/login    B路径： 当前这个html的路径：  http://localhost:8080/Demo4/login.html -->      <form action="login" method="get">  账号:<input type="text" name="username"/><br>  密码:<input type="text" name="password"/><br>  <input type="submit" value="登录"/>  </form>  </body>  </html> |

|  |
| --- |
| login\_success.html  <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>Insert title here</title>  </head>  <body>  <h2>登录成功了</h2>  <a href="CountSrevlet">获取网站登录成功总数 </a>  </body>  </html> |

|  |
| --- |
| CountSrevlet.java  package com.itheima.servlet;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \* Servlet implementation class CountSrevlet  \*/  public class CountSrevlet extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {    //1. 取值  int count = (int) getServletContext().getAttribute("count");    //2. 输出到界面    response.getWriter().write("login success count ===："+count);    }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  } |

|  |
| --- |
| LoginServlet.java  package com.itheima.servlet;  import java.io.IOException;  import java.io.PrintWriter;  import java.util.ArrayList;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  public class LoginServlet extends HttpServlet {      /\*\*  \* request : 包含请求的信息  \*  \* response ： 响应数据给浏览器， 就靠这个对象  \*/  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {      //1. 获取数据  String userName = request.getParameter("username");  String password = request.getParameter("password");    System.out.println("userName="+userName+"==password="+password);  //2. 校验数据    //向客户端输出内容  PrintWriter pw = response.getWriter();    if("admin".equals(userName) && "123".equals(password)){  //System.out.println("登录成功");  // pw.write("login success..");  //成功就跳转到login\_success.html    //1. 成功的次数累加    //获取以前存的值 ， 然后在旧的值基础上 + 1  Object obj = getServletContext().getAttribute("count") ;    //默认就是0次  int totalCount = 0 ;  if(obj != null){  totalCount = (int) obj;  }    System.out.println("已知登录成功的次数是："+totalCount);    //给这个count赋新的值  getServletContext().setAttribute("count", totalCount+1);      //2. 跳转到成功的界面  //设置状态码? 重新定位 状态码  response.setStatus(302);  //定位跳转的位置是哪一个页面。  response.setHeader("Location", "login\_success.html");  }else{  pw.write("login failed..");  }    }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // TODO Auto-generated method stub  doGet(request, response);  }  } |

# 16\_HttpServletRequest 获取头信息\_\_rec

|  |
| --- |
| package com.itheima.servlet;  import java.io.IOException;  import java.util.Enumeration;  import java.util.Iterator;  import java.util.Map;  import java.util.Set;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  public class Demo01 extends HttpServlet {    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {    //1. 取出请求里面的所有头信息 ---- 得到一个枚举集合  Enumeration<String> headerNames = request.getHeaderNames();  while (headerNames.hasMoreElements()) {  String name = (String) headerNames.nextElement();  String value = request.getHeader(name);  System.out.println(name+"="+value);    }    System.out.println("-----------------------");    //2. 获取到的是客户端提交上来的数据  String name = request.getParameter("name");  String address = request.getParameter("address");  System.out.println("name="+name);  System.out.println("address="+address);    System.out.println("-----------------------");    // 获取所有的参数，得到一个枚举集合  // Enumeration<String> parameterNames = request.getParameterNames();    // name=zhangsan&address=beijing    //name=zhangsan&name=lisi&name=wangwu    Map<String, String[]> map = request.getParameterMap();    Set<String> keySet = map.keySet();  Iterator<String> iterator = keySet.iterator();  while (iterator.hasNext()) {  String key = (String) iterator.next();  System.out.println("key="+key + "--的值总数有："+map.get(key).length);  String value = map.get(key)[0];  String value1 = map.get(key)[1];  String value2 = map.get(key)[2];    System.out.println(key+" ======= "+ value + "=" + value1 + "="+ value2);  }    }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  } |

----

|  |
| --- |
| package com.itheima.servlet;  import java.io.IOException;  import java.io.UnsupportedEncodingException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \* Servlet implementation class LoginServlet  \*/  public class LoginServlet extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {    //这个说的是设置请求体里面的文字编码。 get方式，用这行，有用吗？ ---> get  request.setCharacterEncoding("UTF-8");      String username = request.getParameter("username");  String password = request.getParameter("password");    //test01(username, password);  //post过来的数据乱码处理：  System.out.println("post : userName="+username+"==password="+password);    }    /\*\*  \* 处理get请求过来的数据乱码  \* @param username  \* @param password  \* @throws UnsupportedEncodingException  \*/  private void test01(String username, String password) throws UnsupportedEncodingException {  System.out.println("userName="+username+"==password="+password);    //get请求过来的数据，在url地址栏上就已经经过编码了，所以我们取到的就是乱码，  //tomcat收到了这批数据，getParameter 默认使用ISO-8859-1去解码    //先让文字回到ISO-8859-1对应的字节数组 ， 然后再按utf-8组拼字符串  username = new String(username.getBytes("ISO-8859-1") , "UTF-8");  System.out.println("userName="+username+"==password="+password);  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  System.out.println("来了一个post请求...");  doGet(request, response);  }  } |

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>Insert title here</title>  </head>  <body>  执行登录：<br>  <form action="login" method="post">  账号:<input type="text" name="username"/><br>  密码:<input type="text" name="password"/><br>  <input type="submit" value="登录"/>  </form>  </body>  </html> |

# 20\_HttpServletResponse 简单使用（一）\_\_rec

|  |
| --- |
| package com.itheima.servlet;  import java.io.IOException;  import java.nio.charset.Charset;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  public class Demo01 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {    // test01(response);    // test03(response);    //以字节流输出    /\*  \* 如果想让服务器端出去的中文，在客户端能够正常显示。只要确保一点。  \*  \* 出去的时候用的编码 ， 和 客户端看这份数据用的编码 是一样的。  \*  \* 默认情况下getOutputStream 输出使用的是UTF-8的码表 。 如果想指定具体的编码，可以在获取byte数组的时候，指定。  \*  \*  \*  \*/    //设置响应的数据类型是html文本，并且告知浏览器，使用UTF-8 来编码。  response.setContentType("text/html;charset=UTF-8");      //String这个类里面， getBytes()方法使用的码表，是UTF-8， 跟tomcat的默认码表没关系。 tomcat iso-8859-1  String csn = Charset.defaultCharset().name();    System.out.println("默认的String里面的getBytes方法使用的码表是： "+ csn);    //1. 指定浏览器看这份数据使用的码表  // response.setHeader("Content-Type", "text/html;charset=UTF-8");    //2. 指定输出的中文用的码表  response.getOutputStream().write("我爱深圳黑马训练营..".getBytes("UTF-8"));    }    void test03(HttpServletResponse response){  //响应的数据中包含中文---> 乱码。 以字符流输出    try {  //这里写出去的文字，默认使用的是ISO-8859-1 ，我们可以指定写出去的时候，使用什么编码写  //1. 指定输出到客户端的时候，这些文字使用UTF-8编码  response.setCharacterEncoding("UTF-8");  //2. 直接规定浏览器看这份数据的时候，使用什么编码来看。  response.setHeader("Content-Type", "text/html; charset=UTF-8");  response.getWriter().write("我爱黑马训练营...");  } catch (Exception e) {  }  }    void test01(HttpServletResponse response){  //以字符流的方式写数据  //response.getWriter().write("<h1>hello response...</h1>");    try {  //以字节流的方式写数据  response.getOutputStream().write("hello response2222...".getBytes());  } catch (Exception e) {  // TODO: handle exception  }      //设置当前这个请求的处理状态码  //response.setStatus("");    //设置一个头  //response.setHeader(name, value);    //设置响应的内容类型，以及编码。  //response.setContentType(type);  }      protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  } |

# 24\_HttpServletResponse 下载文件（Tomcat提供下载）\_\_rec

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>Insert title here</title>  </head>  <body>  让tomcat的默认servlet去提供下载：<br>  <a href="download/aa.jpg">aa.jpg</a><br>  <a href="download/bb.txt">bb.txt</a><br>  <a href="download/cc.rar">cc.rar</a><br>  <br>手动编码提供下载。：<br>  <a href="Demo01?filename=aa.jpg">aa.jpg</a><br>  <a href="Demo01?filename=bb.txt">bb.txt</a><br>  <a href="Demo01?filename=cc.rar">cc.rar</a><br>    </body>  </html> |
| package com.itheima.servlet;  import java.io.FileInputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \* Servlet implementation class Demo01  \*/  public class Demo01 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {    //1. 获取要下载的文件名字 aa.jpg --- inputStream  String fileName = request.getParameter("filename");    //2. 获取这个文件在tomcat里面的绝对路径地址  String path = getServletContext().getRealPath("download/"+fileName);    //让浏览器收到这份资源的时候， 以下载的方式提醒用户，而不是直接展示。  response.setHeader("Content-Disposition", "attachment; filename="+fileName);    // response.setStatus(302);  // response.setHeader(Location, "login\_success.html");    //  response.sendRedirect("login\_success.html");    //3. 转化成输入流  InputStream is = new FileInputStream(path);  OutputStream os = response.getOutputStream();    int len = 0 ;  byte[]buffer = new byte[1024];  while( (len = is.read(buffer)) != -1){  os.write(buffer, 0, len);  }    os.close();  is.close();    }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  doGet(request, response);  }  } |