**基础知识**

**1.**企业级应用特点：分布式，高速反应和响应，安全性，可扩展性，集成和持续集成

**2.**企业级程序体系结构：C/S两层模式，三层体系结构：显示层，服务器，DB数据库，多层体系结构

**3.javaEE：**是一种结构和一套标准，一种体系结构，有多种技术和框架，是为了简化开发，提高项目效率

编程思想：组件（有独立功能的类）+容器（组件运行的环境）——提供底层服务

分层模型（松耦合，轻量级）：对象层（POJO)，数据访问层(DAO)，业务逻辑层(Service)，控制层(Controller)，表现层(view)

主张思想：惯例优于配置，配置优于编码；

javaEE体系结构优点:1独立于硬件配置和操作系统

1. 坚持面向对象的设计原则
2. 灵活性，可移植性，互操作性
3. 轻松的企业信息系统集成

**4.servlet**:运行在服务端的小程序，本质是类（接口和实现类），源是个.java文件，是一种JavaEE的组件技术

动作：接受请求（request)，给出响应(response)

生存周期方法：init()初始化配置（只调用一次），service（）响应客户请求，destroy（）销毁servlet对象，回收资源（关闭服务器时调用）

非生命周期方法：getServletInfo(),getServletConfig()

运行阶段：每一个servlet只会创建一个对象，每一次浏览器的请求，服务器产生一个线程，调用service（），线程反复执行service（）

HttpServlet类的两个核心方法：doGet(request,response),doPost(request,response)

HttpServlet类的主要两大功能：1自定义处理方法，2调用service方法，处理HTTP请求和响应

每个servlet都有一个一个上下文实例——ServletContext

每个servlet都有一个配置——servletConfig

**5.URL:**绝对路径（action=”http://localhost:8080/j2017051181\_04\_login/Matching”）和相对路径（action=”Matching”)的使用，一般使用相对路径较为简介方便，相对路径是相对当前文件所在的路径，只能通过内部访问，两者都能成功访问对应servlet，因为服务器会自动分析截取url，然后与web.xml文件中的url匹配

**6.配置：**

1—xml配置：一个<servlet-mapping>元素用于映射一个servlet地址访问路径（虚拟路径）,一个servlet可被映射多个虚拟路径，一个<servlet-mapping>下可有多个<url-pattern>元素

<url-pattern>使用用例：<url-pattern>/myServlet</url-pattern>(前端：action=”myServlet”)

2—注解方式：

@webServlet(“/Matching”),

@webServlet(name=”myServlet”,urlPatterns=”/myServlet”)

@webServlet（name=”myServlet”,urlPatterns={”/myServlet”,”yourServlet”,”hisServlet”})

3——通配符：

路径映射：以”/”开头和以”/\*”结尾

扩展映射：以”\*.”开头

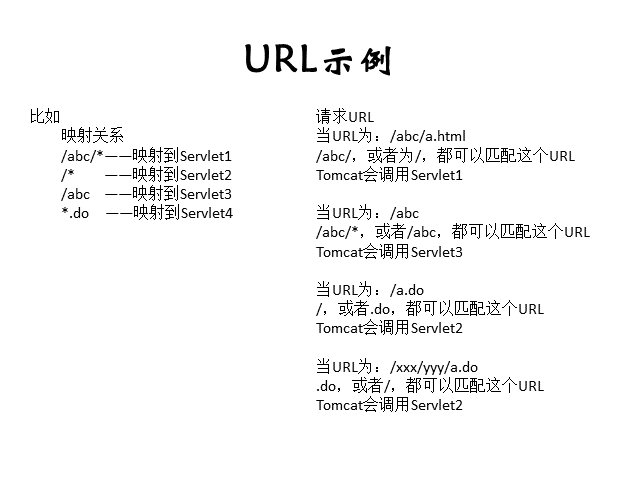
default servlet映射：”/”

详细映射：/aa/bb/cc.action

注意事项：通配符\*只能出现在开头或结尾；不使用通配符，必须以”/”开头

采取原则：最具体匹配原则

示例：



**7.转发和重定向：**

转发（RequestDispatcher对象）：forward（request，response)——请求转发，include（request，response）——包含

转发在请求时，交由其他资源处理，并在服务端完成

重定向（HttpServletResponse对象)：sendRedirect(url)

重定向在响应时，是在客户端重新请求并跳转到指定url,再次请求servlet

转发比重定向高效。

**8.Filter(过滤器）：**拦截请求响应对象

生命周期：1.web.xml实例化，2.init()初始化，3.doFilter()过滤，4.destroy（）销毁

DoFilter(request，response)方法中，chain.doFilter(request，response)用于继续转发请求，转向下一个过滤器或者servlet，若未调用这一方法，请求无法进行转发响应，因此可用来过滤和控制

适用场合：对用户请求进行统一认证（如统一编码），日志审核和记录，图像转换，数据压缩，加密/解密，身份验证，资源访问触发事件过滤

**9.Listener(监听器）：**用于监听另一个java对象的变化——方法监听，属性监听，事件监听

常见三种监听：上下文监听(context)，会话监听(session)，请求监听（request)

**10.JSP:**含有java写成的脚本的动态页面

注释：1.<%--- 注释内容 --%>（客户端不能查看到） <!--注释内容--> （客户端能查看）

变量声明：<%! %> 如：<%! int num=110; %>

jsp表达式：<%= %> 如：<%=”Hello World” %> <%=name(已声明变量) %>

java代码块：<% %>

Jsp指令：<%@指令名属性=”值”%>

page指令：如<%@ page import=”java.util.\* ”%> 定义jsp页面全局属性

include指令：用于包含其他jsp,html，文本文件，被包含文件被加载在指令位置

taglib指令： 允许用户使用自定义标签

jsp动作：1.include动作 <jsp:include page=”路径” flush=”true”/>资源插入，引用

2.Forward动作 <jsp:forward page=”path”}/>请求转发

3.useBean动作 <jsp:useBean id=”name” class=”className” scope=”作用域对象”/>

4.getProperty动作：<jsp:getProperty>获取javaBean中的属性值

5.setProperty动作：<jsp:setProperty>设置javaBean中的属性值

6.param动作：<jsp:param name=”result” value=”110”/> 键值对形式

7.plugin动作

内置作用域对象：page作用范围：只在当前页面有效

request:在一个请求过程中有效

session:在一个会话中有效

application:在服务器运行过程中有效

jsp九大隐含对象：无需”new”获取实例，可直接使用的对象

1.out 2.request 3.response 4.session 5.application

6.pageContext 7.page 8.config 9.exception

[使用详情请按ctrl点击](https://blog.csdn.net/sx0809/article/details/79479206)

**11.服务器应用对象：**

1.ServletRequest: 向服务端传递参数

2.ServletResponse： 回应客户端请求

3.ServletConfig： 获取servlet的初始化参数

4.ServletContext: 存储整个web应用的相关信息，存储全局共享数据，在一个Web服务器中，是唯一的，所有servlet共享共享同一个ServletContext对象

获取方式：1.从ServletConfig对象中获取 2.从HttpServlet对象中获取

应用：1.实现数据共享（存取的数据在服务器运行过程中有效）

1. 获取Web应用的（上下文）初始化参数（web.xml）
2. 获取与服务器相关信息（名字，版本号等）
3. 读取资源文件（本地资源路径，指定参数的文件等）
4. 记录日志（向servlet的日志文件写入消息，写入错误日志和异常等）

5.HttpSession：由服务器创建，在服务端存放用户会话状态和信息

方法：1.setAttribute(String name, Obiect object)——将对象绑定到会话中，使之成为会话属性

2.getAttribute(String name)——返回会话属性

3.getAttributeNames()——返回所有会话属性名

4.removeAttribute(name)——删除会话属性

5.invalidate()——使会话中止，并删除绑定的所有会话属性（关闭浏览器时调用）

6.setMaxInactiveInterval（int interval）——设置会话的超时时间（负数表示永不过期，0表示立即失效）【也可在web.xml配置过期时间】

6.out: 向客户端输送信息

**12.json:** javascript对象表示方式，一种以文本方式表示对象的格式

语法：JSON对象：{属性：值，属性：值}（类似键值对形式）

如：{“stuId”:2017051181,”stuName”:”周鸽”}

JSON数组：{JSON对象,JSON对象,{属性：值}}

如：{stu1,stu2,{“stuId”:2017051181,”stuName”:”周鸽”}}

js中使用json：1. JSON对象：var jsonObj={stu1,{“stuId”:2017051181,”stuName”:”周鸽”}}

1. JSON文本：var jsonText={“stuId”:2017051181,”stuName”:”周鸽”}——文本字符串，是纯文本
2. Json对象和json文本只在js（javascript）中可用

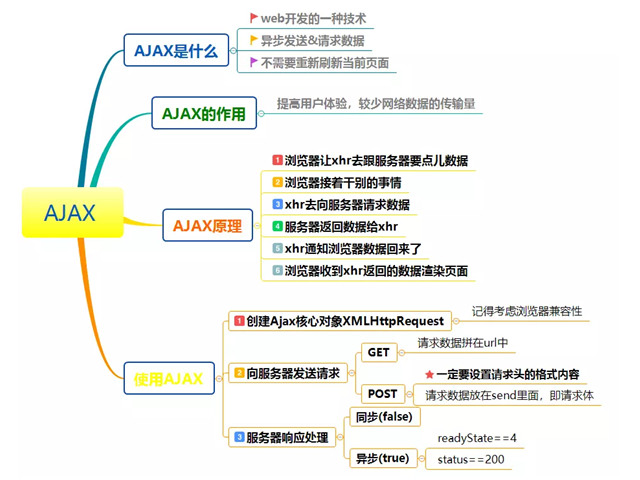
**13.Ajax（异步的JavaScript和XML）：**A和B执行功能，A的执行不会影响B的执行，A和B可同时执行。异步过程调用后，调用者无需得到结果便可继续执行后续操作，其调用返回也不受调用者控制。

网页的异步更新，局部刷新：

1.向服务器发请求的js对象：XMLHttpRequest(**XHR**),使用Ajax，就是使用XHR对象的属性，方法和事件

2.谁发请求，谁接响应：js脚本中对象向服务器发送请求，对象接受响应，通过js脚本刷新局部页面

3.注意事项：html,js的编写修改不需重启服务器，servlet，配置，JavaBean的编写修改，需要重启服务器



**14.EL表达式：**存取数据的语言，计算和输出java对象

特点：1.直接访问JavaBean中的属性：${user.userName} 或者 ${user[userName]}

2.可执行运算：逻辑运算，算术运算，条件运算，关系运算等

3.可获得命名空间

4.除0无异常（返回无穷大Infinity）

5.可访问作用域对象(page,request,session,application)

6.输出文本：${“不要温和地走进那良夜”}

**查看[详情请按ctrl点击](https://blog.csdn.net/pan_junbiao/article/details/88567466" \o "查看详情请点击)**

**15.JSTL：**取代在jsp中嵌入java代码的做法，提高可维护性

核心标签库： 表达式操作：<c:out> <c:set> <c:remove> <c:catch>

流程控制：<c:if> <c:choose> <c:when> <c:otherwise>

迭代操作：<c:forEach> <c:forTokens>

[查看JSTL标签使用详情请按ctrl点击]("https://blog.csdn.net/weixin_42808295/article/details/81460609)

**16.JDBC:**规避数据库的不同，未开发人员访问数据库提供统一的编程接口（API）

JDBC API做三件事：**创建连接，创建执行语句，执行语句返回结果**

DriverManager:依赖数据库不同，管理加载驱动

Connection：连接数据库和传送数据

Statement:由Connection产生，负责执行sql语句

ResultSet：保存Statement执行后产生的处理结果

JDBC编程经典6步：**1.加载驱动：**使用Class类下的静态方法forName来加载驱动

**Class.*forName*(*jdbcDriver*)**

加载MySql驱动

——Class.forName(“com.mysql.jdbc.Driver“）

加载SQL Server驱动

——Class.forName(“com.microsoft.sqlserver.jdbc.SQLServerDriver“）

**2.创建连接:**通过DriverManager获取数据库连接

**Connection conn=DriverManager.getConnection(url,username,password)**

1.MySql—— url="jdbc:mysql://localhost:3306/database"

2.SQL Server——url="jdbc:sqlserver://localhost:1433;databaseName=database"

（datbase为要连接的数据库名）

**3.准备sql语句：**通过Connection对象创建Statemenbt对象

**1.String sql="....."**

**2.PreparedStatement ps = conn.prepareStatement(sql)——**预编译对象

或**Statement ps=conn.createStatement(sql)——**基本对象

**4.执行sql语句:**使用Statement对象执行SQL语句

**ResultSet rs=ps.executeQuery()——执行查询语句**

**int num=ps.executeUpDate()——执行增删改语句**

**ResultSet rs=ps.execute()——通用，执行任何SQL语句**

**5.处理结果:**返回的ResultSet对象来取出查询结果

while(**rs**.next()){

String xx=**rs**.getString("xx");

.......

}

**6.回收资源:**关闭ResultSet，关闭Statement,关闭Connection

**rs.close();**

**Ps.close();**

**Conn.close();**

1. **MVC分层：**Model（模型）——用来处理业务

View(视图）——用于显示

Controller(控制器）——用来控制分派

项目具体实现功能：

1. 用jsp做界面2.用servlet处理请求 3.用service处理业务（主要是接口和实现）

4.用dao处理数据库访问和操作 5.用pojo(entity)封装数据 6用junit做单元测试

7.DBUtils封装数据库访问工具和方法

项目分层结构：表示层→控制层→业务层→DAO层→数据访问层（依次依赖）

数据传递：jsp发出请求，传递参数→servlet接受请求，调用service→service调用dao层→dao访问数据库，处理业务，调用pojo封装数据,返回结果→service得到数据→servlet响应，发送数据→jsp显示数据

1. **控制反转/依赖注入（IOC）**——[点击了解]("https://blog.csdn.net/weixin_43499644/article/details/89385143)
2. **javaBean的构造器（构造方法）访问器（访问方法）和设置器（设置方法）——<https://www.cnblogs.com/gynbk/p/6556257.html>**