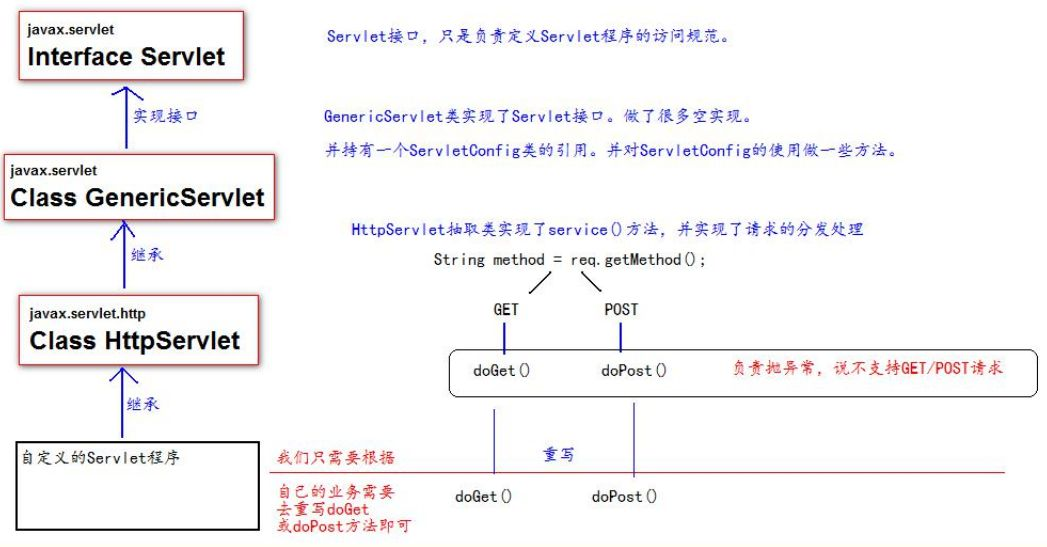
**Servlet基础**

* Servlet
  + 什么是Servlet
    - Java Servlet 是运行在 Web 服务器或应用服务器上的程序，它是作为来自 Web 浏览器或其他 HTTP 客户端的请求和 HTTP 服务器上的数据库或应用程序之间的中间层。
    - 下图显示了 Servlet 在 Web 应用程序中的位置。
  + 为什么使用Servlet
    - Servlet是持久的。Servlet只需Web服务器加载一次，而且可以在不同请求之间保持服务。
    - Servlet是快速的。
    - Servlet是与平台无关的。
    - Servlet是可扩展的。
    - Servlet是安全的。
  + Servlet运行流程
    - 客户端（很可能是Web浏览器）通过HTTP提出请求。
    - Web服务器接收该请求并将其转发给对应的Servlet。
    - Servlet将接受该HTTP请求并执行相应业务处理。
    - Servlet向Web服务器返回响应。
    - Web服务器将从Servlet收到的响应发送给客户端。
    - 客户端将结果显示出来。
  + Servlet体系结构
* 配置Servlet
  + 部署描述文件的位置
    - web.xml文件总是放置在Web应用的WEB-INF目录中。这是各个服务器之间唯一通用的位置。
  + 基本格式的定义
    - xml规则，必须有且只有一个根节点，大小写敏感，标签不嵌套，必须配对。
  + Servlet的命名与URL的指定
  + 部署描述文件的读取
    - 热部署是指在你修改项目BUG的时候对JSP或JAVA类进行了修改在不重启WEB服务器前提下能让修改生效。但是对配置文件的修改除外!
* Servlet常用接口和类
  + Javax.servlet包
    - Servlet接口，这个接口是所有Servlet必须直接或间接实现的接口。
      * init(ServletConfig config): 用于初始化Servlet。
      * destroy() ：销毁Servlet。
      * getServletInfo()：获得Servlet的信息。
      * getServletConfig()：获得Servlet配置相关信息。
      * service(ServletRequest req, ServletResponse res)：运行应用程序的入口，他接受两个参数，ServletRequest表示客户端请求的信息，ServletResponse表示对客户端的响应。
    - GenericServlet类
      * GenericServlet类是一种与协议无关的Servlet，是一种不对请求提供服务的Servlet，而是简单地通过init（）方法启动后台线程并在destory（）中销毁。它可以用于模拟操作系统的端口监控进程。GenericServlet类提供除了service（）方法外所有接口中方法的缺省实现。这意味着通过简单的扩展GenericServlet可以编写一个基本的servlet。他是一个抽象类，他的service（）方法是一个抽象方法，GenericServlet的派生类必须直接或者间接实现这个方法。
      * ServletConfig，在Servlet的初始化中，初始化方法使用ServletConfig对象作为参数，通常这个方法保存这个对象，以便能够有方法getServletConfig（）返回。在初始化的方法中，应当保存ServletConfig对象，并且重新编写getServletConfig（）方法，以便它能够从新的位置得到对象。
        + getInitParameter(java.lang.String name)：返回特定名字的初始化参数。
        + getInitParameterNames()：返回所有的初始化参数的名字。
        + getServletContext()：返回Servlet的上下文对象的引用。
      * ServletException
        + getRootCause()：返回造成这个ServletException的原因。
      * UnavailableException
        + 当Servlet或者Filter暂时或者永久不能使用时，就会抛出这个异常。
      * ServletRequest
        + 代表了Servlet的请求，他是一个高层的接口，HttpServletRequest是它的子接口。
      * ServletResponse
        + 代表了Servlet的响应，他是一个高层的接口，HttpServletResponse是它的子接口。
      * ServletInputStream
        + Servlet的输入流。
      * ServletOutputStream
        + Servlet的输出流。
      * ServletContext
        + 在服务器上使用Session对象来维持通单个客户相关的状态，而当为多个用户的Web应用维持一个状态时，则应使用Servlet环境（context）。
      * RequestDispatcher
        + forward(ServletRequest request, ServletResponse response)：把请求转发到服务器上的另一个资源（Servlet、JSP、HTML）
        + include(ServletRequest request, ServletResponse response)：把服务器上的另一个资源（Servlet、JSP、HTML）包含到响应中。
  + Javax.servlet.http包
    - HttpServlet
      * HttpServlet类扩展了GenericServlet，提供Servlet接口的HTTP协议的特定实现。如果Servlet必须遵循HTTP协议，则只能扩展HttpServlet类并覆盖所需的方法。
      * HttpServlet类调用指定到HTTP请求方法实现service（），即对DELETE、HEAD、GET、OPTIONS、POST、PUT、和TRACE，分别调用doDelete（）、doGet（）、doOptions（）、doPost（）、doPut（）和doTrace（）方法，将请求和响应对象置入其HTTP指定子类。
      * HttpServlet类是针对使用Http协议的Web服务器的Servlet类。HttpServlet类通过执行Servlet接口，能够提供Http协议的功能。
      * HttpServlet的子类必须实现以下方法中的一个：doGet、doPost、doPut、doDelete、init和destory、getServletInfo。
    - HttpServletRequest
      * HttpServletRequest接口可以获取由客户端传送的参数名称，也可以获取客户端正在使用的通信协议，可以获取产生请求并且接受请求的服务器远端主机名和其IP地址等信息。
    - HttpServletResponse
      * 代表了对客户端的Http的响应。它允许Servlet设置内容长度和回应的MIME类型，并且提供输出流ServletOutputStream。
    - HttpSession
      * 这个接口被Servlet引擎用来实现Http客户端和Http会话之间的联系。这种关联可能在多处连接和请求中持续一段给定的时间。Session用来在无状态的Http协议下越过多个请求页面来维持状态和识别用户。
* Servlet通信方式
  + doGet
    - doGet（）用来处理HTTP的GET、HEAD请求。
  + doPost
* Servlet Get和Post的区别
  + get和post这是http协议的两种方法，另外还有head, delete等 这两种方法有本质的区别，get只有一个流，参数附加在url后，大小个数有严格限制且只能是字符串。post的参数是通过另外的流传递的，不通过url，所以可以很大，也可以传递二进制数据，如文件的上传。 在servlet开发中，以doGet()和doPost()分别处理get和post方法。 另外还有一个doService(), 它是一个调度方法，当一个请求发生时，首先执行doService(),不管是get还是post。在HttpServlet这个基类中实现了一个角度，首先判断是请求时get还是post,如果是get就调用doGet(), 如果是post就调用doPost()。你也可以直接过载doService()方法，这样你可以不管是get还是post。都会执行这个方法。
* 使一个Servlet上同时支持Get和Post
  + 在Servlet的编写中常常遇到在同一个Servlet上需要同时支持GET和POST方法。可通过如下方法来实现：把所有的操作都在doGet里实现，然后再doPost里调用doGet方法。
* Java Servlet与JSP（JavaServer pages）的关系
  + （1）Servlet和JSP的确本质都是Servlet类（2）tomcat服务器说白了就是Servlet容器，并不能直接运行jsp页面，只是将其翻译为可执行的类了（3）jsp能实现的，用Servlet都可以实现，两者功能相同。
* Servlet的生命周期
  + Servlet 生命周期可被定义为从创建直到毁灭的整个过程。以下是 Servlet 遵循的过程：Servlet 初始化后调用 init () 方法。Servlet 调用 service() 方法来处理客户端的请求。Servlet 销毁前调用 destroy() 方法。最后，Servlet 是由 JVM 的垃圾回收器进行垃圾回收的。

  
下图显示了 Servlet 在 Web 应用程序中的位置。

