目录

[JavaEE概述 3](#_Toc59442508)

[三层体系结构应用程序 3](#_Toc59442509)

[定义 3](#_Toc59442510)

[优点 3](#_Toc59442511)

[Java EE技术体系结构 4](#_Toc59442512)

[Servlet 4](#_Toc59442513)

[Servlet定义 4](#_Toc59442514)

[Servlet和Applet对比 5](#_Toc59442515)

[Servlet工作流程 5](#_Toc59442516)

[Servlet装载和初始化 7](#_Toc59442517)

[调用Servlet 7](#_Toc59442518)

[处理请求 7](#_Toc59442519)

[多个请求的处理 8](#_Toc59442520)

[退出 8](#_Toc59442521)

[JSP 8](#_Toc59442522)

[session.jsp 8](#_Toc59442523)

[session2.jsp 9](#_Toc59442524)

[session3.jsp 9](#_Toc59442525)

[JavaBean 10](#_Toc59442526)

[JavaBean属性 10](#_Toc59442527)

[JavaBean 程序示例 10](#_Toc59442528)

[StudentBean.java 11](#_Toc59442529)

[访问JavaBean 11](#_Toc59442530)

[Student.jsp 12](#_Toc59442531)

[访问 JavaBean 对象的属性 12](#_Toc59442532)

[JDBC 13](#_Toc59442533)

[创建JDBC应用程序 13](#_Toc59442534)

[导入包 13](#_Toc59442535)

[注册JDBC驱动程序 13](#_Toc59442536)

[打开一个连接 13](#_Toc59442537)

[执行一个查询 13](#_Toc59442538)

[从结果集中提取数据 14](#_Toc59442539)

[清理环境 14](#_Toc59442540)

[第一个JDBC 程序：FirstExample.java 14](#_Toc59442541)

[网页制作和网站建设 15](#_Toc59442542)

[网站建设流程 15](#_Toc59442543)

[网站策划 15](#_Toc59442544)

[网站设计 16](#_Toc59442545)

[代码实现 16](#_Toc59442546)

[网站测试与完善 16](#_Toc59442547)

[网站发布 16](#_Toc59442548)

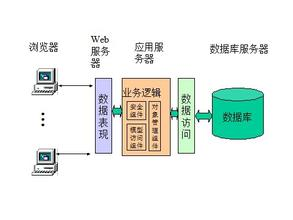
[后期更新与维护 16](#_Toc59442549)

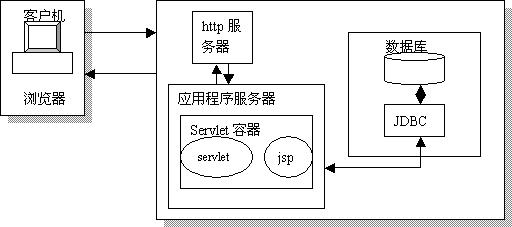
# JavaEE概述

## 三层体系结构应用程序

### 定义

* + 客户层
  + 应用服务器层：中间件服务器层
  + 数据服务器层





在客户层和服务器层之间增加第三层：应用服务器层

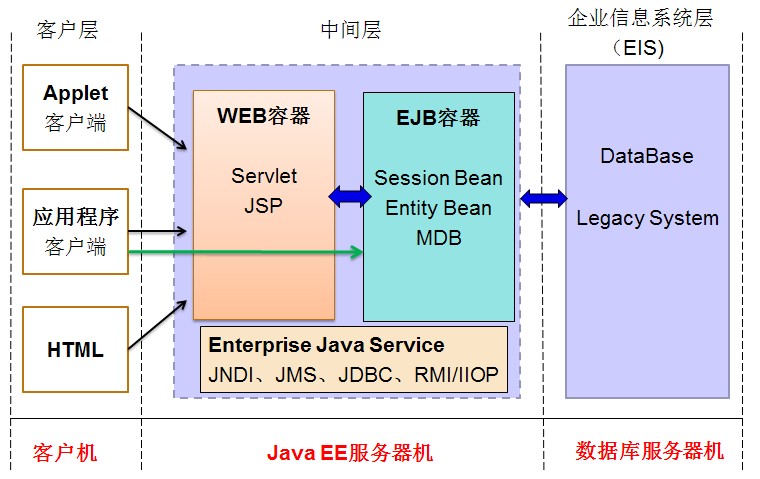
客户层应用程序由一个通用的浏览器（Browser）程序实现

B/S模式或“瘦客户机”模式

### 优点

* 安全性高
  + 隔离直接访问，保护数据信息安全。
* 易维护
  + 业务规则变化后，客户端程序基本不改动，升级应用服务器层的程序即可。
* 快速响应
  + 通过中间件服务器的负载均衡以及缓存数据能力，可以大大提高对客户端的响应速度。
* 系统扩展灵活
  + 可以通过在应用服务器部署新的程序组件来扩展系统规模；当系统性能降低时，在中间件服务器层部署更多的应用服务器来提升系统性能，缩短用户端的响应。

## Java EE技术体系结构



# Servlet

## Servlet定义

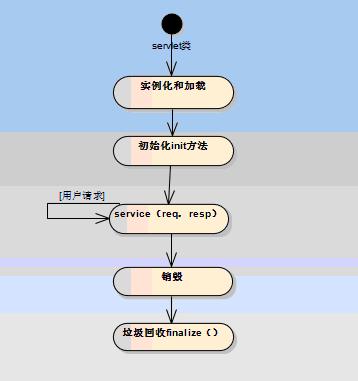
* **Servlet**（Server Applet），全称**Java Servlet**，未有中文译文，是用Java编写的服务器端程序（Servlet 是在服务器上运行的小程序。），独立于操作系统平台和网络传输协议。其主要功能在于交互式地浏览和修改数据，生成动态Web内容。
* Servlet运行于支持Java的应用服务器中。从原理上讲，Servlet可以响应任何类型的请求，但绝大多数情况下Servlet只用来扩展基于HTTP协议的Web服务器。

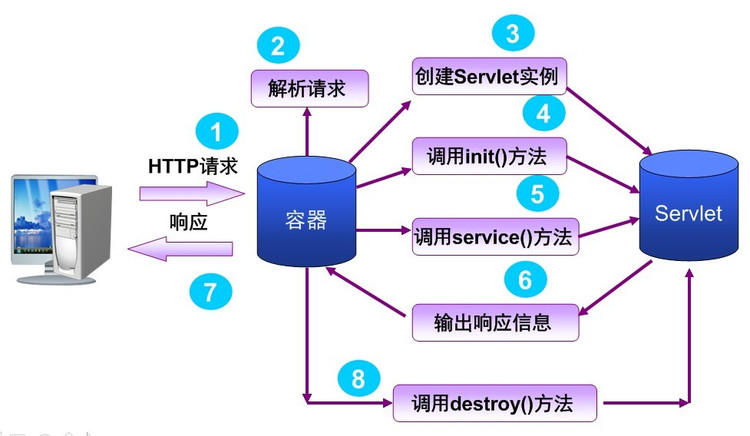
## Servlet和Applet对比

* Applet是运行于客户端浏览器的Java应用程序，Servlet与 Applet 的比较
* **相似之处：**
* \* 它们不是独立的应用程序，没有 [main](http://baike.baidu.com/view/746175.htm)() 方法。
* \* 它们不是由用户或程序员调用，而是由另外一个应用程序(容器)调用。
* \* 它们都有一个生存周期，包含 init() 和 destroy() 方法。
* **不同之处：**
* \* [Applet](http://baike.baidu.com/view/150834.htm)具有很好的图形界面([AWT](http://baike.baidu.com/view/209544.htm))，与浏览器一起，在客户端运行。
* \* Servlet 则没有图形界面，运行在[服务器](http://baike.baidu.com/view/899.htm)端。
* Applet
  + 实现浏览器与客户的交互，需要图形用户界面
* Servlet
  + 扩展服务器功能，实现业务逻辑

## Servlet工作流程

* Servlet运行在Web服务器上的Web容器中，其生命周期由Web容器来管理。
  + Web容器装入并初始化Servlet，管理Servlet的多个实例，并充当请求调度器，将客户端的请求传递到Servlet，并将Servlet的响应返回给客户端。
  + 服务器关闭时， Web容器从内存中卸载和除去Servlet。
* Servlet基本工作流程
* Servlet基本工作流程的说明





* 客户端将请求发送到服务器
* 服务器上的Web容器实例化（装入）Servlet，并为Servlet进程创建线程。 Servlet是出现第一个请求时装入的，在服务器关闭之前不会卸载它。
* Web容器将请求信息发送到Servlet
* Servlet创建一个响应，并将其返回到Web容器
* Web容器将响应返回客户端
* 服务器关闭或Servlet空闲时间超过一定限度时，调用destory()方法退出。

### Servlet装载和初始化

* + 当Servlet容器启动时，或者在容器检测到需要这个Servlet来响应第一个请求时，创建Servlet实例。
  + 当Servlet容器启动后，它必须要知道所需的Servlet类在什么位置，Servlet容器可以从本地文件系统、远程文件系统或者其他的网络服务中通过类加载器加载Servlet类，成功加载后，容器创建Servlet的实例。
  + 在Servlet实例化之后，容器将调用Servlet的init()方法初始化这个对象。初始化的目的是为了让Servlet对象在处理客户端请求前完成一些初始化的工作，如建立数据库的连接，获取配置信息等。对于每一个Servlet实例，init()方法只被调用一次。

init()两个版本：没有参数，以ServletConfig对象为参数

### 调用Servlet

* + 每个Servlet都对应一个url地址。（图3-9）
  + Servlet可以作为显式url引用调用，或者嵌入html中从web应用程序调用。
  + URL是英文“Uniform Resource Locators”的缩写，意思是“统一资源定位器”。它不仅可用来定位网络上信息资源的地址，也可用来定位本地系统要访问的文件。
  + URL由三部分组成：资源类型、存放资源的主机域名、资源文件名。
  + URL可分为两种类型，一种是绝对URL，另一种是相对URL。

### 处理请求

* + Servlet容器调用Servlet的service()方法对请求进行处理。要注意的是，在service()方法调用之前，init()方法必须成功执行。在service()方法中，Servlet实例通过ServletRequest对象得到客户端的相关信息和请求信息，在对请求进行处理后，调用ServletResponse对象的方法设置响应信息。在service()方法执行期间，如果发生错误，Servlet实例可以抛出 ServletException异常或者UnavailableException异常。如果UnavailableException异常指示了该实例永久不可用，Servlet容器将调用实例的destroy()方法，释放该实例。此后对该实例的任何请求，都将收到容器发送的HTTP 404（请求的资源不可用）响应。如果UnavailableException异常指示了该实例暂时不可用，那么在暂时不可用的时间段内，对该实例的任何请求，都将收到容器发送的HTTP 503（服务器暂时忙，不能处理请求）响应。

### 多个请求的处理

* + 一个Servlet同一时刻只有一个实例
  + 同时有多个请求发送到同一个Servlet，服务器将会为每个请求创建一个新的线程来处理客户端的请求

### 退出

* + 如果Web应用程序关闭或者Servlet已经空闲了很长时间，Web容器会将Servlet实例从内存移除。
  + 移除之前Web容器会调用Servlet的destory()方法。
  + Servlet使用destory()方法关闭数据库连接、中断后台线程。。。

# JSP

* 会话处理变得必不可少，当一个请求数据需要被持续保持以供进一步使用。 由于HTTP协议认为每个请求是一个新的请求，它不能保持过去访问的数据，因此会话处理就变得很重要。以下是一些来处理会话的方法。  
  JSP中每当发起一个请求，服务器产生一个存储在客户机的唯一会话ID。
* Cookies存储信息在客户端浏览器
* URL重写会话信息附加到URL的末尾
* 隐藏的表单域将SessionID嵌入到GET和POST命令。

## session.jsp

* <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>
* <html>
* <head><title>Session示例</title></head>
* <body>
* <h2>Session示例：</h2>
* <form method = "post" action="session2.jsp">
* <font>Username<input type = "text" name = "name"></font>
* </font><br><br>
* <input type = "submit" name = "submit" value = "提交" >
* </form>
* </body>
* </html>

## session2.jsp

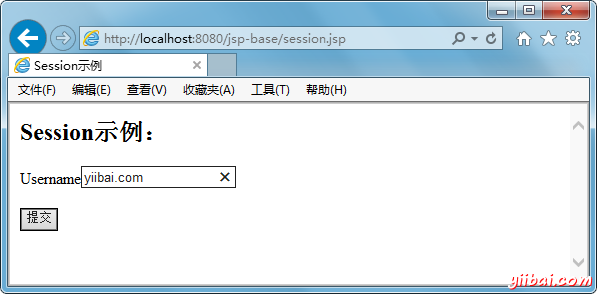
* <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  
  <html>
* <head><title>Session示例2</title></head>
* <body>   
  <%  
  String name = request.getParameter("name");  
  if((name!=null))  
  {  
      session.setAttribute("username",name);  
  }  
  %>  
  <a href="session3.jsp">继续，跳转到session3.jsp</a>
* </body>  
  </html>

## session3.jsp

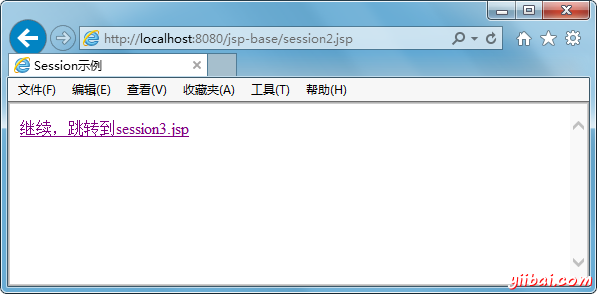
* <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>
* <html>
* <head><title>Session示例3</title></head>
* <body>
* <font>欢迎您，</font> <%= session.getAttribute("username") %>
* </body>
* </html>

在第一个示例中的 "session1.jsp" 是用来提供一个表单以获得用户名。 当提交表单时它转到第二个文件session2.jsp，它调用表单的“action”属性。一个Session的属性使用 "session.setAttribute"设置在这里. 在第三个文件 "session3.jsp" 相同的值使用"session.getAttribute" 来显示出来。

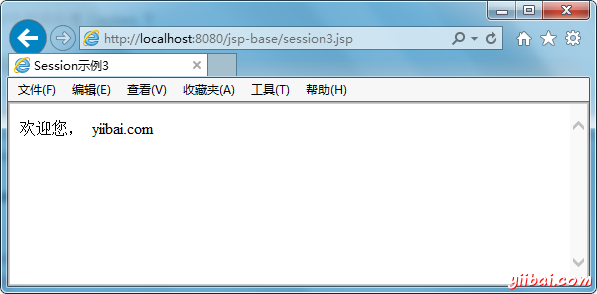
在浏览器中打开网址：http://localhost:8080/jsp-base/session.jsp，得到以下结果：



填写好用户名，点击“提交”，进入第二个过度页面，如下：



然后再点击页面中的链接，进入第三个页面，测试Session记录的结果信息，如下图：



# JavaBean

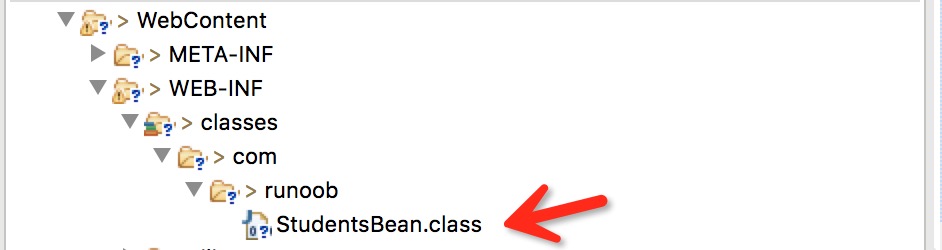
## JavaBean属性

* 一个JavaBean对象的属性应该是可访问的。这个属性可以是任意合法的Java数据类型，包括自定义Java类。
* 一个JavaBean对象的属性可以是可读写，或只读，或只写。JavaBean对象的属性通过JavaBean实现类中提供的两个方法来访问：

|  |  |
| --- | --- |
| * 方法 | * 描述 |
| * get**PropertyName**() | * 举例来说，如果属性的名称为myName，那么这个方法的名字就要写成getMyName()来读取这个属性。这个方法也称为访问器。 |
| * set**PropertyName**() | * 举例来说，如果属性的名称为myName，那么这个方法的名字就要写成setMyName()来写入这个属性。这个方法也称为写入器。 |

## JavaBean 程序示例

* StudentBean.java
* 编译 StudentBean.java 文件：$ javac StudentsBean.java
* 编译后获得 **StudentBean.class** 文件，将其拷贝到 **<JSP 项目>/WebContent/WEB-INF/classes/com/runoob**，如下图所示:



## StudentBean.java

* private String firstName = null;
* private String lastName = null;
* private int age = 0;
* public StudentsBean() {
* }
* public String getFirstName(){
* return firstName;
* }
* public String getLastName(){
* return lastName;
* }
* public int getAge(){
* return age;
* }
* public void setFirstName(String firstName){
* this.firstName = firstName;
* }
* public void setLastName(String lastName){
* this.lastName = lastName;
* }
* public void setAge(int age) {
* this.age = age;
* }

## 访问JavaBean

* <jsp:useBean> 标签可以在JSP中声明一个JavaBean，然后使用。声明后，JavaBean对象就成了脚本变量，可以通过脚本元素或其他自定义标签来访问。<jsp:useBean>标签的语法格式如下：
* <jsp:useBean id="bean 的名字" scope="bean 的作用域" typeSpec/>
* 其中，根据具体情况，scope的值可以是page，request，session或application。id值可任意只要不和同一JSP文件中其它<jsp:useBean>中id值一样就行了。

## Student.jsp

* <jsp:useBean id="students"
* class="com.runoob.StudentsBean">

## 访问 JavaBean 对象的属性

* 在 **<jsp:useBean>** 标签主体中使用 **<jsp:getProperty/>** 标签来调用 **getter** 方法，使用 **<jsp:setProperty/>** 标签来调用 **setter** 方法，语法格式如下：
* <jsp:useBean id="id" class="bean 编译的类" scope="bean 作用域">
* <jsp:setProperty name="bean 的 id" property="属性名"
* value="value"/>
* <jsp:getProperty name="bean 的 id" property="属性名"/>
* ...........
* </jsp:useBean>
* <jsp:useBean id="students"
* class="com.runoob.StudentsBean">
* <jsp:setProperty name="students" property="firstName"
* value="小强"/>
* <jsp:setProperty name="students" property="lastName"
* value="王"/>
* <jsp:setProperty name="students" property="age"
* value="10"/>
* </jsp:useBean>
* <p>学生名字:
* <jsp:getProperty name="students" property="firstName"/>
* </p>
* <p>学生姓氏:
* <jsp:getProperty name="students" property="lastName"/>
* </p>
* <p>学生年龄:
* <jsp:getProperty name="students" property="age"/>
* </p>
* name属性指的是Bean的id属性。property属性指的是想要调用的getter或setter方法。
* 实例：student.jsp
* 访问以上 JSP，运行结果如下：
* 学生名字: 小强
* 学生姓氏: 王
* 学生年龄: 10

# JDBC

## 创建JDBC应用程序

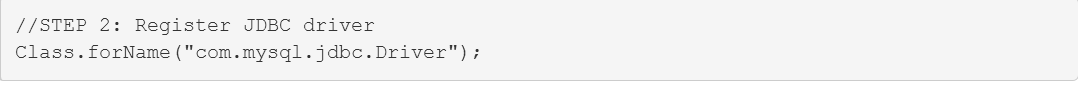
* 建立一个JDBC应用程序，本教程中按六个步骤进行：

## 导入包

* 这需要你有软件包包含了数据库编程所需的JDBC类。大多数情况下，使用import java.sql.\* 就足够了，如下所示：
* 

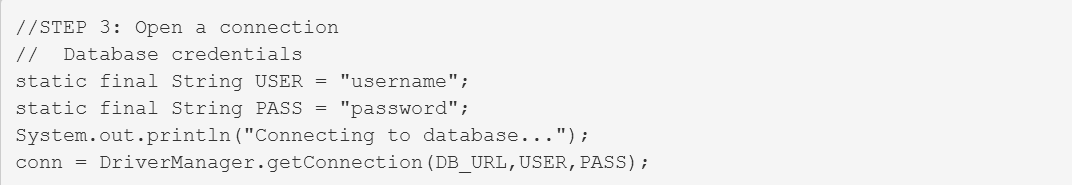
## 注册JDBC驱动程序

* 需要初始化驱动程序，这样就可以打开与数据库的通信信道。以下是代码片段实现这一目标：



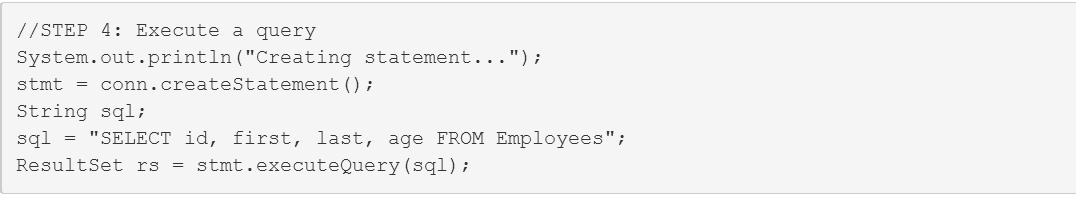
## 打开一个连接

* 需要使用DriverManager.getConnection()方法来创建一个Connection对象，它代表一个物理连接的数据库，如下所示：

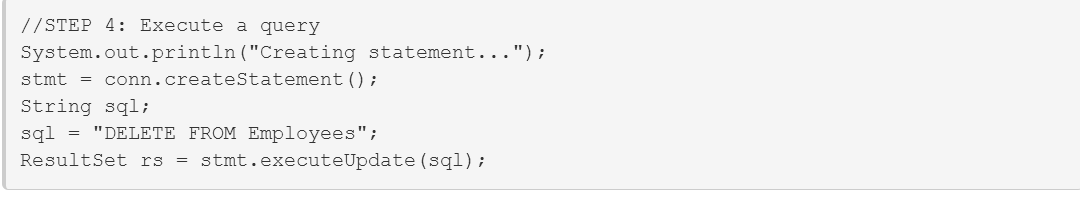


## 执行一个查询

* 需要使用一个对象类型Statement或PreparedStatement构建，并提交一个SQL语句到数据库。如下：

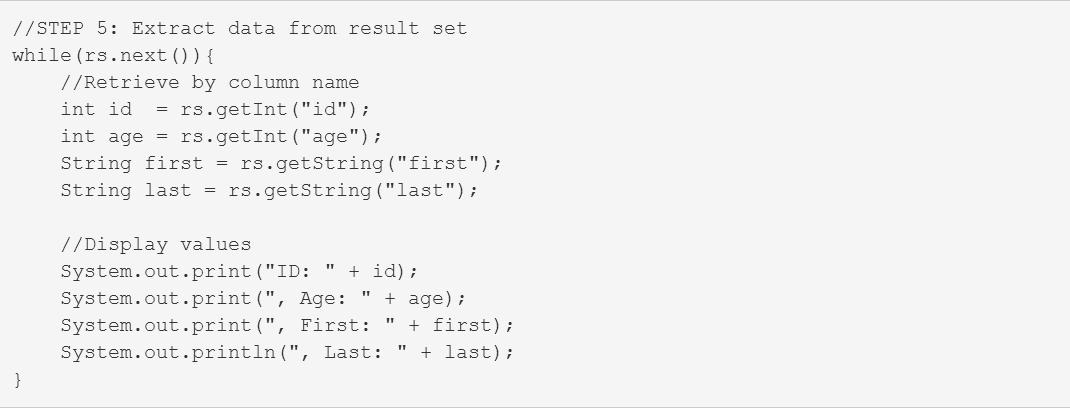


* 如果有一个SQL UPDATE，INSERT或DELETE语句，那么需要下面的代码片段

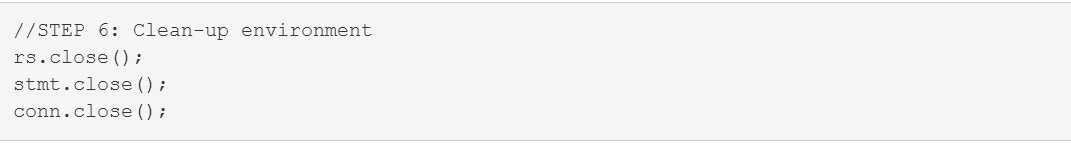


## 从结果集中提取数据

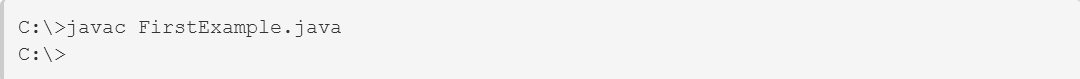
* 使用适当的ResultSet.getXXX()方法来检索的数据结果如下：



## 清理环境

* 明确地关闭所有的数据库资源，对依赖于JVM的垃圾收集如下：
* 

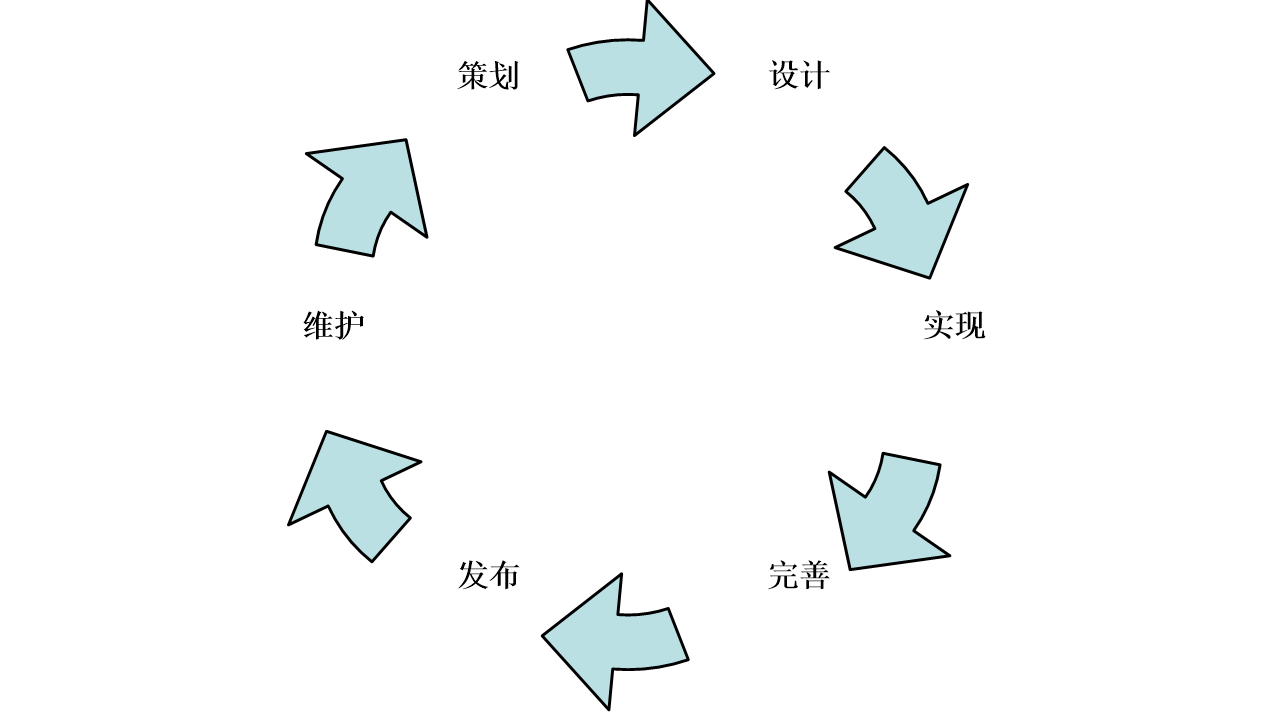
## 第一个JDBC 程序：FirstExample.java

* 于上面的步骤，我们可以有以下综合示例代码，我们可以用它作为模板而写JDBC代码： **FirstExample.java**
* 编译：
* 
* 运行FirstExample的，它会产生以下结果：



# 网页制作和网站建设

## 网站建设流程



## 网站策划

* **网站策划的主要内容**
* 网站策划就是通过客观的、科学的方法，借助策划学的知识，依托互联网的相关技术和行业知识，对一个网站从构思到建设再到推广运营而提出的一条龙的整体建设思路和实现方法，以保证这个网站能够顺利出生并且健康成长。
* **网站策划书的撰写**
* 每个网站的情况不同，网站策划书也是不同的，但是最终都不要离开主的框架，在网站建设前期，要进行市场分析，形成书面报告，对网站建设和运营进行有计划的指导和阶段性概述。

## 网站设计

* **门户网站：**门户网站，是指通向某类综合性互联网信息资源并提供有关信息服务的应用系统。在全球范围中，最为著名的门户网站则是谷歌以及雅虎，而在中国，最著名的门户网站有众所周知的中国四大门户网站（新浪、网易、搜狐、腾讯），其他也有百度、新华网、人民网、凤凰网等也较为著名。门户网站设计主要是简单，界面友好，要追求网页打开速度。门户网站的首页通常是比较简单中留出一点独特的风格，在不影响网页速度的前提下保留个性与独到。
* **企业网站：**企业网站的范围很广，涉及各个领域。但它们的共同之处就是：以宣传为主。企业宣传自己的目的都是为了提升企业形象，希望有越来越多的人能关注自己的公司和产品，以获得更大的发展。企业宣传网站的设计就要根据企业所处的行业，结合自身特点，传达企业最新最全的信息。
* **个人网站：**个人网站通常是两种形式，一种是爱好型的非赢利网站，这种网站通常采用大量的图片来追求炫目的感觉。一种是个人门户型网站，和门户型网站的设计属于一类。

## 代码实现

* 网站代码编程是网站安全的基础，一个好的程序可以是网站受到攻击而产生不良后果的问题大大减少，代码实现需要专业的编程技术，一般来说网站流行的编程语言有以下几种：JSP、ASP、.NET、PHP等等。

## 网站测试与完善

* 网站的设计和编程全部做完之后，要对网站进行测试和上传。
* 首先应该将网站上传到网站空间，然后对网站进行测试，同时也是对网站空间进行测试。一般来说，网站测试需要进行的就是网站页面的完整程度，网站编程代码的繁简程度和完整性，网站空间的链接速度和网站空间的加压测试承受度。

## 网站发布

* 编写好的网站（如果是动态网站，需要事先导出数据库，并确保服务器支持此数据库的导入和连接）
* 服务器空间（需要知道服务器的域名或IP地址）
* FTP客户端工具（需要知道FTP上传的用户名和密码）

## 后期更新与维护

* 对网站进行数据库填充，对网站内容和版式进行实时维护和及时更新。