

# The JAVA Dev. Tech.

## JAVA软件开发技术

荆欣

Jingxin@netcop.com.cn



# Outline...

- ▷ 内容回顾
- ▷ 3中常用的Java WEB JDBC技术路线
- ▷ JSP
- ▷ HTTP
- ▷ Web项目的部署

# 内容回顾



# 课程内容



# Java面向对象中的1234

- ▷ 一种工具：UML – United Modeling Language
- ▷ 两个概念：类与对象
- ▷ 三个特点：封装、继承与多态
- ▷ 四种关系：继承、关联、组成（聚合与组合）与消息传递

# 抽象类与接口

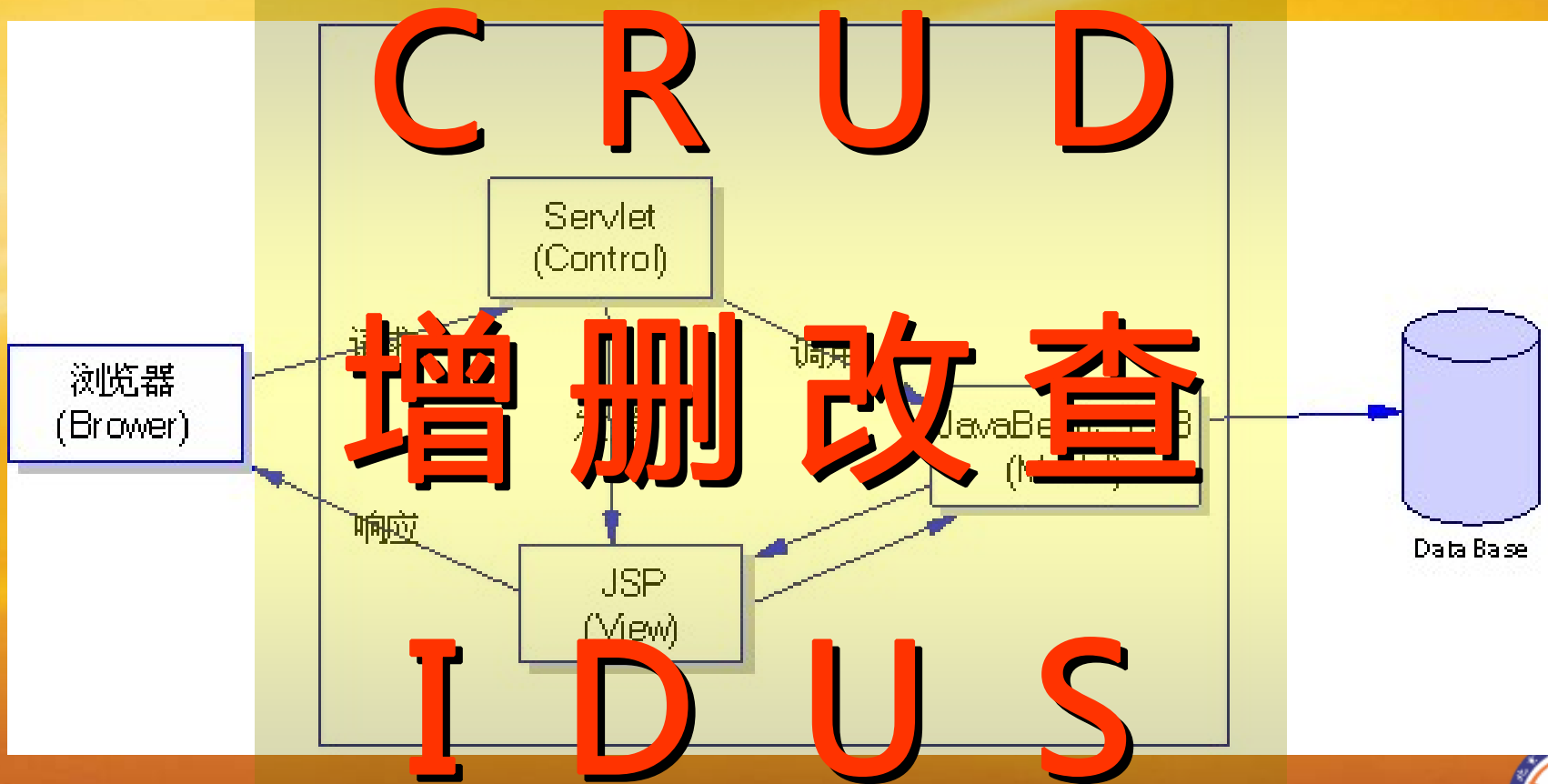
- ▷ 抽 → 纯抽 → 接口
- ▷ 一般使用接口声明方法或常量，接口中的方法只能是声明，不能是具体的实现
- ▷ 接口是一种规范，该规范声明了类中那些方法对外是公开的。
- ▷ 类来实现这种规范。
- ▷ 面向接口编程。







# More About JDBC





# 3-way : Using JDBC

- ▷ Register: 向表中写入数据
- ▷ Login: 从表中读出一条数据
- ▷ ShowUserList: 从表中读出一组数据

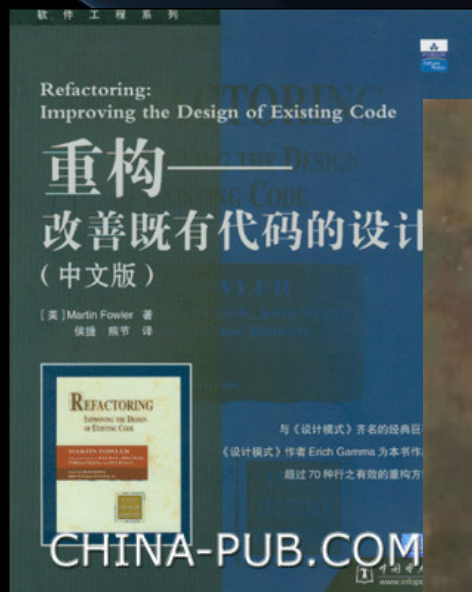


# Register – 写入数据 (Insert/Delete/Update)



# 目标：迭代1 – Register功能实现

- ▷ 没有面向对象，没有复杂设计
- ▷ 单纯实现功能 – 面向过程式的



Martin Fowler

***Refactoring***

# 迭代1-纯技术实现Register

register.html - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Reload Home Search Favorites

Address <http://localhost:8080/RegisterDemo1/register.html>

## Register!

Username:

Password:

Sex: ☐ M ☐ F ☐ E

ID:

Email:

Nationality: China

```
mysql> desc userinfo;
```

Field	Type	Null	Key	Default	Extra
username	varchar(20)	YES		NULL	
password	varchar(60)	YES		NULL	
sex	varchar(4)	YES		NULL	
email	varchar(50)	YES		NULL	
nationality	varchar(20)	YES		NULL	
id	varchar(18)	NO	PRI		

```
6 rows in set (0.00 sec)
```

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    System.out.println("into Servlet!!");
    // get data
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    String sex = request.getParameter("sex");
    String id = request.getParameter("id");
    String email = request.getParameter("email");
    String nation = request.getParameter("nation");
```



# Register - Demo

## ▷ 用户注册基本功能需求

- ▶ 用户名【>6chars, 字母开头】
- ▶ 密码【>8chars】
- ▶ 性别
- ▶ 身份证号【15/18】
- ▶ Email
- ▶ 国籍



# Presentation Layer – Core HTML

▷ `<form action="" method="">`

▷ `<input type="" name="" value="">`

“目的地”

```
<form action="servlet/RegisterServlet" method="post">
  Username:<input type="text" name="username" /><br>
  Password:<input type="password" name="password" />
  Sex:<input type="radio" name="sex" value="M"/>M <i
  ID:<input type="text" name="id" /><br />
```

为了在Servlet中取值

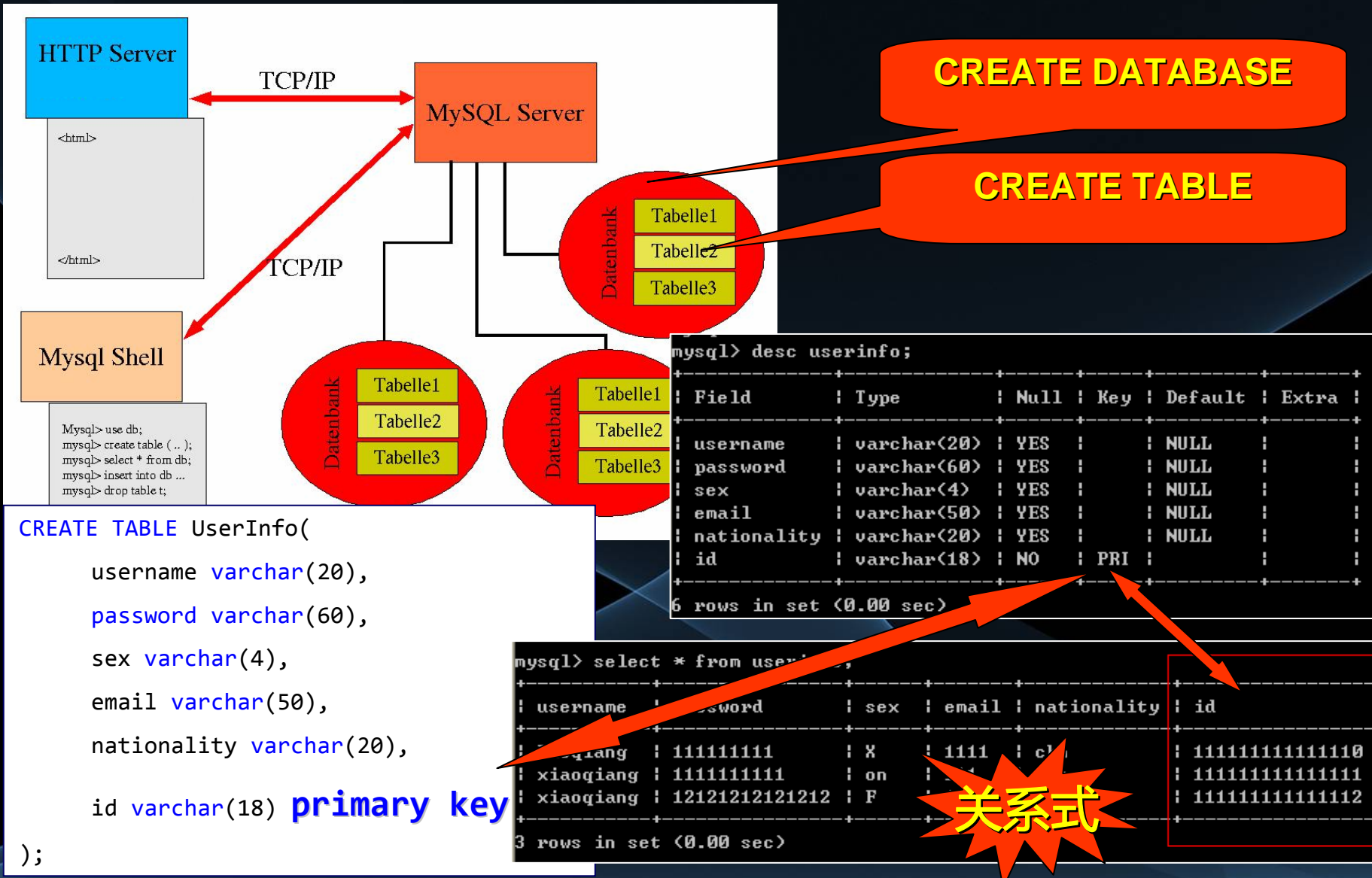
```
Request request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("into Servlet!!");
    String data;
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    String sex = request.getParameter("sex");
    String id = request.getParameter("id");
    String email = request.getParameter("email");
    String nation = request.getParameter("nation");
```

# Presentation Layer – Core Servlet

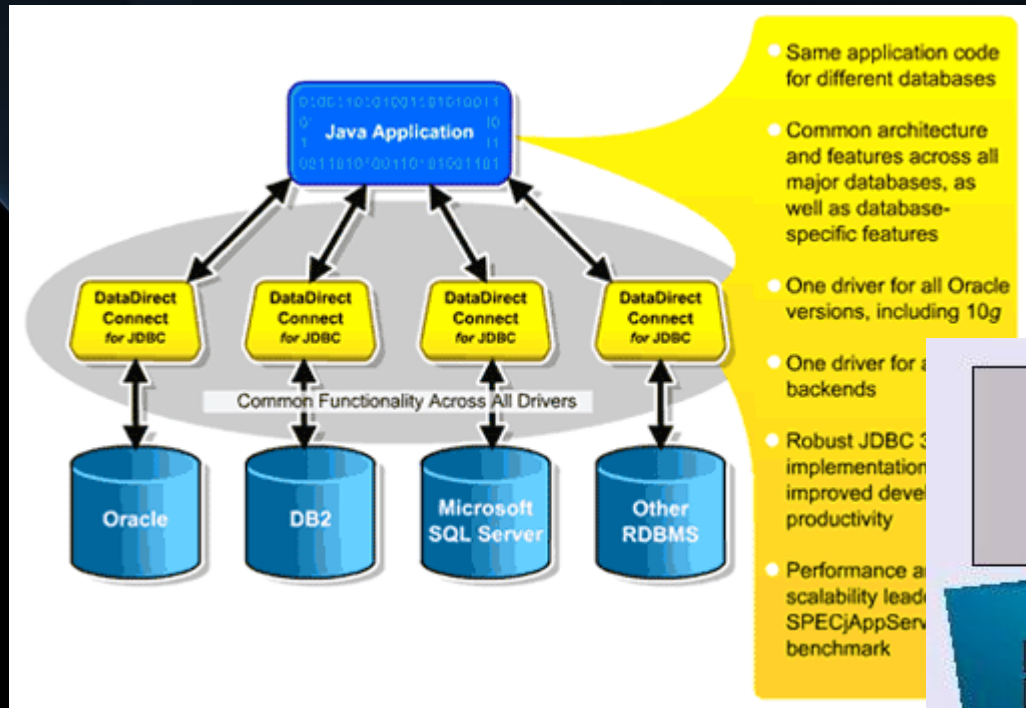
- ▷ Servlet实质就是一个类 – Class (属性/变量 + 方法/函数)
- ▷ Servlet的核心功能
  - ▶ 接收请求(Request)
  - ▶ 调用业务层业务类，完成相应功能
  - ▶ 发送响应(Response)
- ▷ 以上功能是通过相应类中的方法完成的。



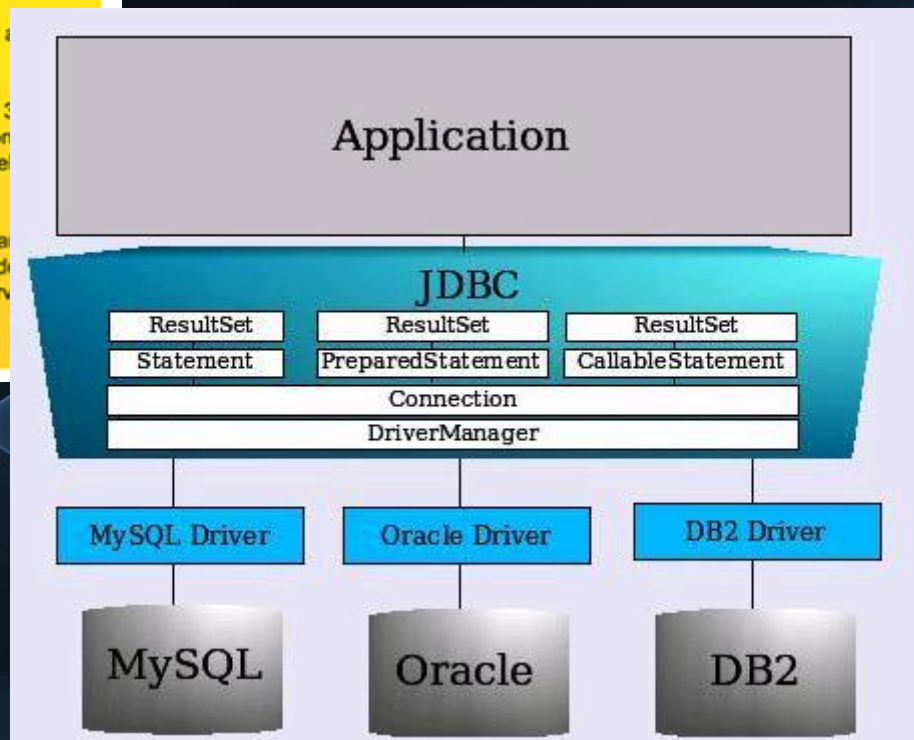
# Database – Core MySQL



# Persistence Layer - JDBC



**Java Data Base Connectivity**



# Persistence Layer – JDBC Driver

## ▷ Type 1: JDBC-ODBC Bridge

‣ *Application <--> JDBC-ODBC Bridge <--> ODBC Driver <--> Database*

## ▷ Type 2: Native-API Bridge

‣ *Application <--> Native-API Bridge <--> Native Driver <--> Database*

## ▷ Type 3: JDBC-middleware

‣ *Application <--> JDBC-middleware <--> middleware <--> Database*

## ▷ Type 4: Pure Java Driver

‣ *Application <--> Pure Java Driver <--> Database*



# Persistence Layer – JDBC

▷ 导入java.sql.\*;

▷ 连接Connection

- ▶ `Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/Garbage?user=root&password=root");`
- ▶ 关键信息
  - 主机地址（及相应端口）
  - 数据库名称
  - 登录的用户名和密码

# Persistence Layer – JDBC

## ▷ SQL命令的发送与执行 – PreparedStatement

```
PreparedStatement ps = conn.prepareStatement(  
    "INSERT INTO UserInfo VALUES(?,?,?,?,?,?,?)");  
ps.setString(1, username);  
ps.setString(2, password);  
ps.setString(3, sex);  
ps.setString(4, email);  
ps.setString(5, nation);  
ps.setString(6, id);  
ps.execute();
```

# Persistence Layer – JDBC

## ▷ SQL命令的发送与执行 – PreparedStatement

```
Statement st = conn.createStatement();
```

```
st.executeUpdate("INSERT INTO UserInfo  
VALUES('"+username+"','"+password+"','"+sex  
+"','"+email+"','"+nation+"','"+id+"')");
```

# Persistence Layer – JDBC

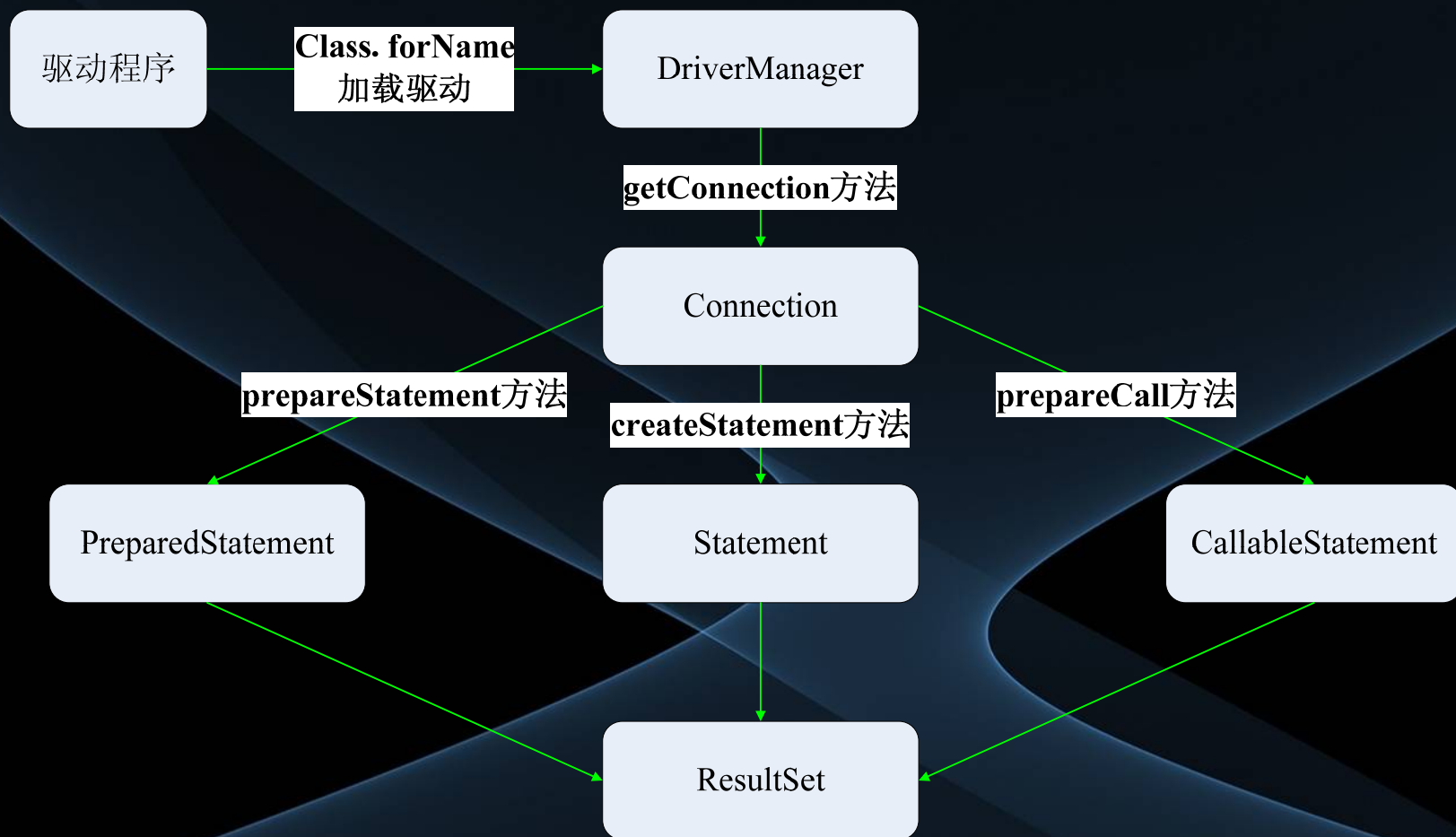
▷ 资源关闭 - close



# Package - java.sql

类名	说明
Connection	此接口表示与数据的连接
PreparedStatement	此接口用于执行预编译的 SQL 语句
ResultSet	此接口表示了查询出来的数据库数据结果集
Statement	此接口用于执行 SQL 语句并将数据检索到 ResultSet 中
DriverManager	此类用于加载和卸载各种驱动程序并建立与数据库的连接
Date	此类包含将 SQL 日期格式转换成 Java 日期格式的各种方法
Time	此类用于表示时间
TimeStamp	此类通过添加纳秒字段为时间提供更高的精确度

# JDBC主要接口和类的关系



# Register – 写入数据(I/D/U)

此种操作，不需要从DB返回什么结果

```
Connection conn =  
    DriverManager.getConnection("jdbc:mysql.....=root");
```

```
PreparedStatement ps = conn.prepareStatement(  
    "INSERT INTO UserInfo VALUES(?,?,?,?,?,?,?)");
```

```
ps.setString(1, username); // set 2,3,4,5.....
```

```
ps.setString(6, id);
```

可以为Insert/Delete/Update

```
int count = ps.executeUpdate();
```

```
ps.close();
```

```
conn.close();
```

对应Insert/Delete/Update，返回操作行数；  
对应其余DDL，返回0。

# Login – 读出一条数据 (Select)

Login

Xiaoqiang | 12345678



# Login – 读出一条数据(Select)

此种操作，需要从DB返回一条数据

```
Connection conn = DriverManager.getConnection("...");
```

```
PreparedStatement ps = conn.prepareStatement(
```

```
    "SELECT * FROM UserInfo WHERE Username=? and  
    Password =?");
```

```
ps.setString(1, username);
```

```
ps.setString(2, password);
```

```
ResultSet rs = ps.executeQuery();
```

```
// 处理ResultSet
```

查询结果被放在结果集ResultSet中

```
// .....
```

```
conn.close();
```

```
mysql> SELECT * FROM UserInfo WHERE Username='xiaoqiang' and password='1111111111';
```

username	password	sex	email	nationality	id
xiaoqiang	1111111111	on	111	chn	111111111111111111

```
1 row in set (0.00 sec)
```

# 处理ResultSet

```
while(rs.next()){
```

数据库表中的  
列名

```
    username = rs.getString("username");
```

```
    password = rs.getString("password");
```

```
}
```

rs的初始位置  
在这里，要先  
next()一下，才  
能读到第一行  
数据

```
mysql> SELECT * FROM USERINFO WHERE Username='xiaoqiang' and password='1111111111';
```

username	password	sex	email	nationality	id
xiaoqiang	1111111111	on	111	chn	1111111111111111

```
1 row in set (0.00 sec)
```

# ShowUserList – 读出一组数据(Select)

ShowUserList

Xiaoqiang | 12345678  
Laoqiang | 87654321  
Xiaoming | 11223344

DB





# ShowUserList – 读出一组数据

此种操作，需要从DB返回一组数据

```
Connection conn =  
    DriverManager.getConnection("...");  
  
PreparedStatement ps = conn.prepareStatement(  
    "SELECT * FROM UserInfo");
```

```
ResultSet rs = ps.executeQuery();
```

```
// 处理ResultSet
```

要将多条数据显示在网页上

```
// .....
```

```
conn.close();
```

```
mysql> SELECT * FROM USERINFO;  
+-----+-----+-----+-----+-----+-----+  
| username | password | sex | email | nationality | id |  
+-----+-----+-----+-----+-----+-----+  
| xiaoqiang | 1111111111 | on | 111 | chn | 1111111111111111 |  
| laoqiang | 2222222222 | X | eee | chn | 2222222222222222 |  
| xiaoming | 3333333333 | X | www | usa | 3333333333333333 |  
+-----+-----+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

# Questions?

- ▷ 程序中，读出的一组数据如何存储？
- ▷ 这些数据如何传递到网页上？
- ▷ 在网页上如何显示出数据？用HTML？

```
mysql> SELECT * FROM USERINFO;
```

username	password	sex	email	nationality	id
xiaoqiang	1111111111	on	111	chn	1111111111111111
laoqiang	2222222222	X	eee	chn	2222222222222222
xiaoming	3333333333	X	www	usa	3333333333333333

3 rows in set (0.00 sec)

# 表中数据的存储 - 结构

## User

-username : string  
-password : string  
-sex : string  
-email : string  
-nationality : string  
-id : string  
+getXXX() : string  
+setXXX()

注意类和表结构之间的对应关系

```
mysql> desc userinfo;
```

Field	Type	Null	Key	Default	Extra
username	varchar(20)	YES		NULL	
password	varchar(60)	YES		NULL	
sex	varchar(4)	YES		NULL	
email	varchar(50)	YES		NULL	
nationality	varchar(20)	YES		NULL	
id	varchar(18)	NO	PRI		

6 rows in set (0.00 sec)

```
String username = request.getParameter("username")
String password = request.getParameter("password")
String sex = request.getParameter("sex");
String id = request.getParameter("id");
String email = request.getParameter("email");
String nation = request.getParameter("nation");
```

# 表中数据的存储 - 实例化

```
mysql> SELECT * FROM USERINFO;
```

username	password	sex	email	nationality	id
xiaoqiang	1111111111	on	111	chn	1111111111111111
laoqiang	2222222222	X	eee	chn	2222222222222222
xiaoming	3333333333	X	www	usa	3333333333333333

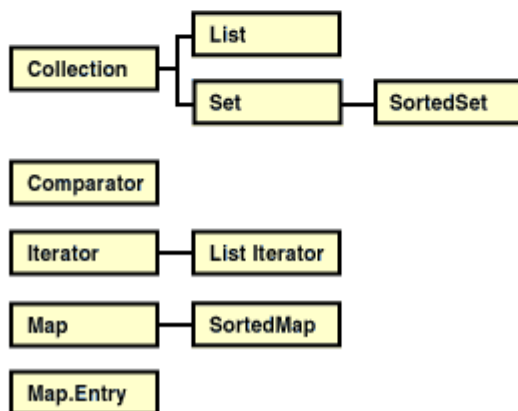
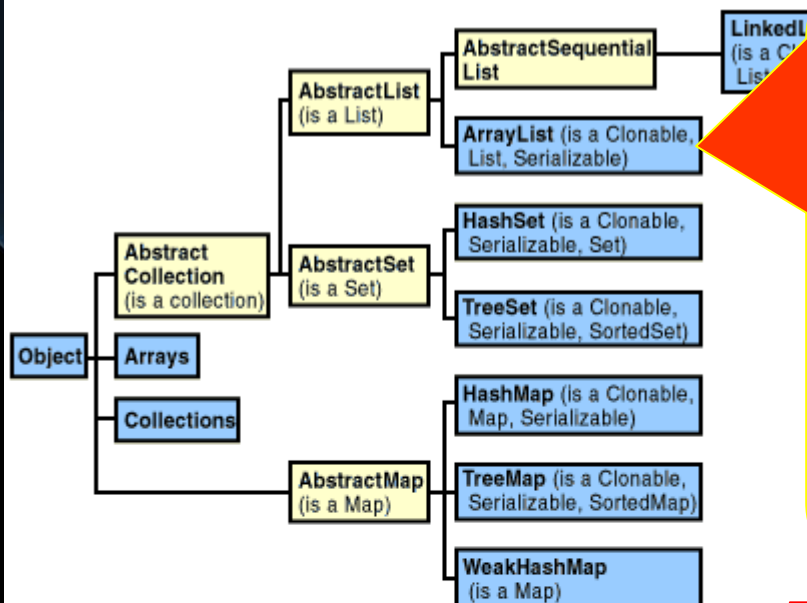
3 rows in set (0.00 sec)

```
User user0 = new User();// xiaoqiang  
User user1 = new User();// laoqiang  
User user2 = new User();// xiaoming
```

```
User [] user = new User[3];  
user[0] = new User();// xiaoqiang  
user[1] = new User();// laoqiang  
user[2] = new User();// xiaoming
```

如何知道是  
3个？  
数量能否动  
态变化？

# 表中数据的存储 - 集合框架



**ArrayList** - 一个容器:  
存储

```
ArrayList list = new ArrayList();  
list.add(...);  
list.add(...);  
list.add(...);
```

**数量是动态的**

取出  
`list.get(0);`  
`list.get(1);`

```
User [] user = new User[3];  
user[0] = new User();// xiaoqiang  
user[1] = new User();// laoqiang  
user[2] = new User();// xiaoming  
// 比较上下两种方案  
ArrayList userList = new ArrayList();  
userList.add(new User());// xiaoqiang  
userList.add(new User());// laoqiang  
userList.add(new User());// xiaoming
```



# 数据的传输 – From DB To Page

- ▷ 数据如何从表中传递到程序里 – **JDBC**
- ▷ 数据如何从ArrayList传递到网页上?

```
<!DOCTYPE HTML PUBLIC "-//W3C//  
<html>  
  <head>  
    <title>ShowUser</title>  
  </head>  
  <body>  
    </body>  
</html>
```

```
User [] user = new User[3];  
user[0] = new User();// xiaoqiang  
user[1] = new User();// laoqiang  
user[2] = new User();// xiaoming  
// 比较上下两种方案  
ArrayList userList = new ArrayList();  
userList.add(new User());// xiaoqiang  
userList.add(new User());// laoqiang  
userList.add(new User());// xiaoming
```



# Session – 会话



# 解决方案

```
<!DOCTYPE HTML PUBLIC "-//W3C//  
<html>  
  <head>  
    <title>ShowUser</title>  
  </head>  
  <body>  
    ?  
  </body>  
</html>
```

```
User [] user = new User[3];  
user[0] = new User();// xiaoqiang  
user[1] = new User();// laoqiang  
user[2] = new User();// xiaoming  
// 比较上下两种方案  
ArrayList userList = new ArrayList();  
userList.add(new User());// xiaoqiang  
userList.add(new User());// laoqiang  
userList.add(new User());// xiaoming
```

`session.getAttribute("UserInSession")`

Session

`session.setAttribute("UserInSession", userList)`

`HttpSession session = request.getSession();`



# JSP – Java Server Pages

- ▷ JSP = HTML (<...>)+ Java(<%...%>)
- ▷ 动态网页
- ▷ JSP首先在Server被执行，将其中的Java运行，最后将结果连同原来的HTML一起发向Client。
  - 。

# JSP的执行示例

demo.jsp

请求 Request

HTTP

响应 Response

Client  
\_A\_

Server

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<body>
  这是<b>HTML</b>的内容。<p />
  这是动态的JSP内容：今天是Tue Nov 13 12:50:51 CST 2007
</body>
</html>
```

http://localhost:8080/RegisterDemo1/demo.jsp - Micros

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites

Address http://localhost:8080/RegisterDemo1/demo.jsp

这是HTML的内容。

这是动态的JSP内容：今天是Tue Nov 13 12:50:51 CST 2007

```
<%@ page language="java" import="java.util.*" pageEncoding="GBK"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <body>
    这是<b>HTML</b>的内容。<p />
    这是动态的JSP内容：今天是<% out.println(new Date()); %>
  </body>
</html>
```



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<body>
  这是<b>HTML</b>的内容。<p />
  这是动态的JSP内容：今天是Tue Nov 13 12:50:51 CST 2007

</body>
</html>
```



# JSP Components

静态内容

HTML静态文本

指令

以“<%@ ”开始，以“%> ”结束。比如：  
<%@ include file = " Filename" %>

表达式

<%=Java表达式 %>

**JSP 页面**

Scriptlet

<% Java 代码 %>

声明

<%! 函数或方法 %>

动作

以“<jsp: 动作名 ”开始，以“</jsp:动作名> ”结束  
比如：<jsp:include page=" Filename" />

注释

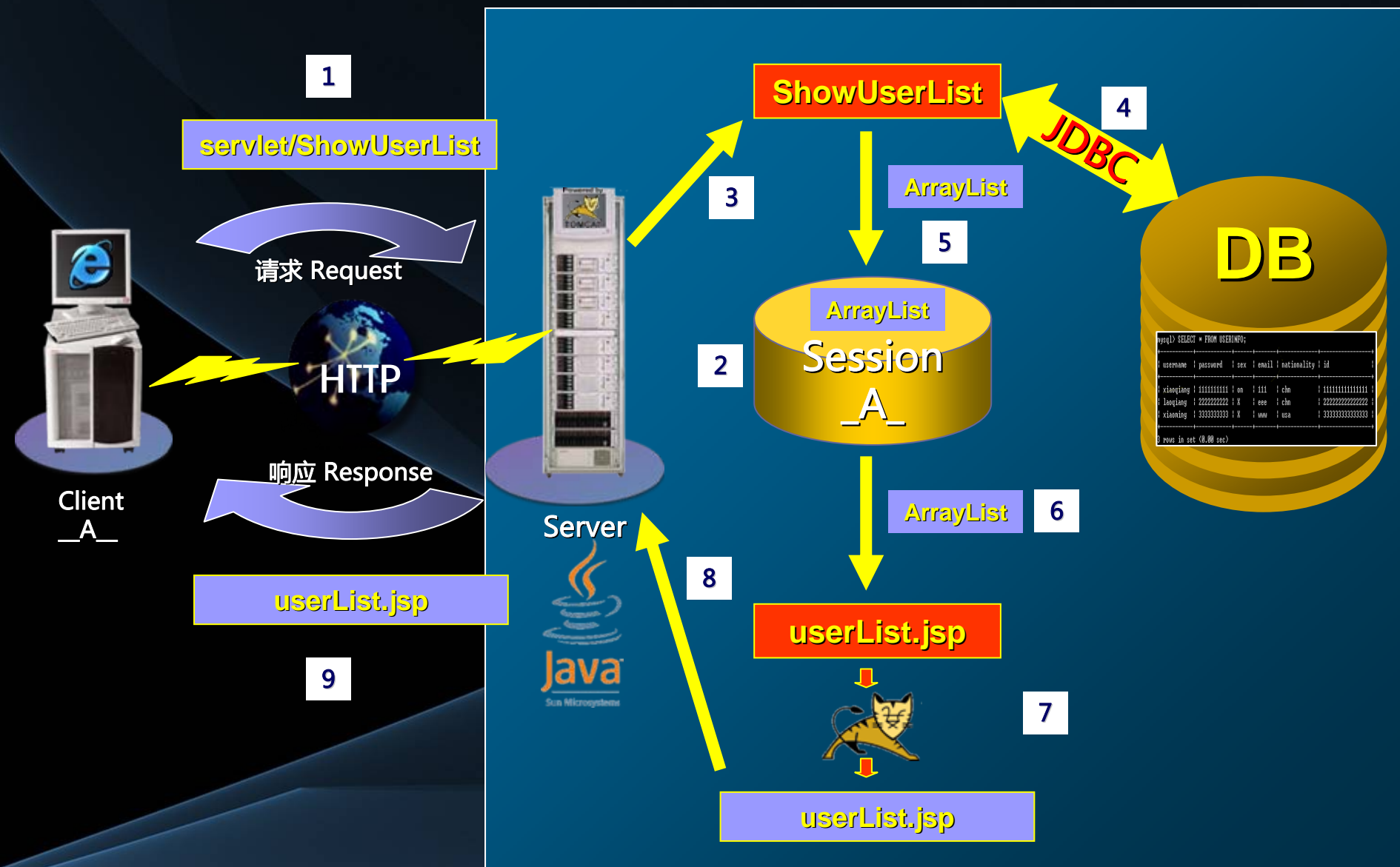
<!-- 这是注释,但客户端可以查看到 -->  
<%-- 这也是注释,但客户端不能查看到 --%>

# JSP Components – 隐式对象





# 图解ShowUserList的执行





# HTTP






# HTTP - Hyper Text Transfer Protocol



# HTTP – 主要特点

- ▷ **客户/服务器**模式
  - ▷ **简单快速**：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。
  - ▷ **灵活**：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。
- ▷ **无连接**：限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。
  - ▷ **无状态**：协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

# HTTP – Request Method : POST

Address  http://localhost:8080/RegisterDemo1/login.html  Go  Links >>

Login!!!

Username:

Password:

```
<form action="servlet/LoginServlet" method="post">
  Username:<input type="text" name="username" /><br />
  Password:<input type="password" name="password" /> <br />
</form>
```

x

POST /RegisterDemo1/servlet/LoginServlet HTTP/1.1  
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/msword, application/vnd.ms-excel, application/vnd.ms-powerpoint, \*/\*  
Referer: http://localhost:8080/RegisterDemo1/login.html  
Accept-Language: en-us  
Content-Type: application/x-www-form-urlencoded  
Accept-Encoding: gzip, deflate  
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)  
Host: localhost:8080  
Content-Length: 27  
Connection: Keep-Alive  
Cache-Control: no-cache  
Cookie: JSESSIONID=D4CABAACA5ABBACDADE0859BD8E46DD3

HTTPHeaders




username=name&password=pass

名字=值&名字=值...

```
String username = request.getParameter("username");
String password = request.getParameter("password");
```



# HTTP – Request Method : GET

Address  http://localhost:8080/RegisterDemo1/login.html  Go  Links >>

Login!!!

Username:

Password:

```
<form action="servlet/LoginServlet" method="get">
  Username:<input type="text" name="username" /><br />
  Password:<input type="password" name="password" /> <br />
</form>
```

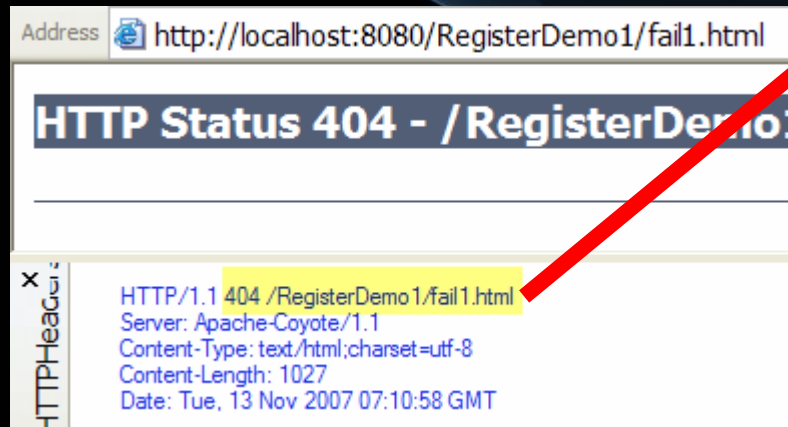
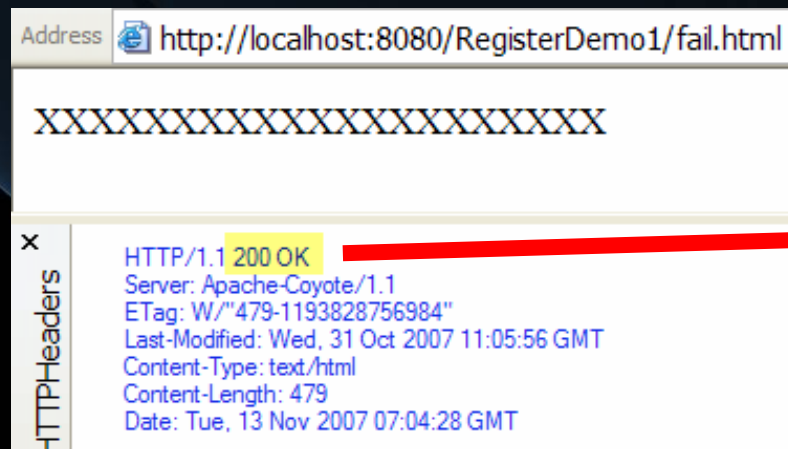
x GET /RegisterDemo1/servlet/LoginServlet?username=name&password=pass HTTP/1.1  
Accept: image/gif, image/x-bitmap, image/jpeg, image/png, application/x-shockwave-flash, application/msword, application/vnd.ms-excel, application/vnd.ms-powerpoint, \*/\*  
Referer: http://localhost:8080/RegisterDemo1/login.html  
Accept-Language: en-us  
Accept-Encoding: gzip, deflate  
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)  
Host: localhost:8080  
Connection: Keep-Alive  
Cookie: JSESSIONID=D4CABAACA5ABBACDAED0358BD8E46DD3

名字=值&名字=值...

```
String username = request.getParameter("username");
String password = request.getParameter("password");
```



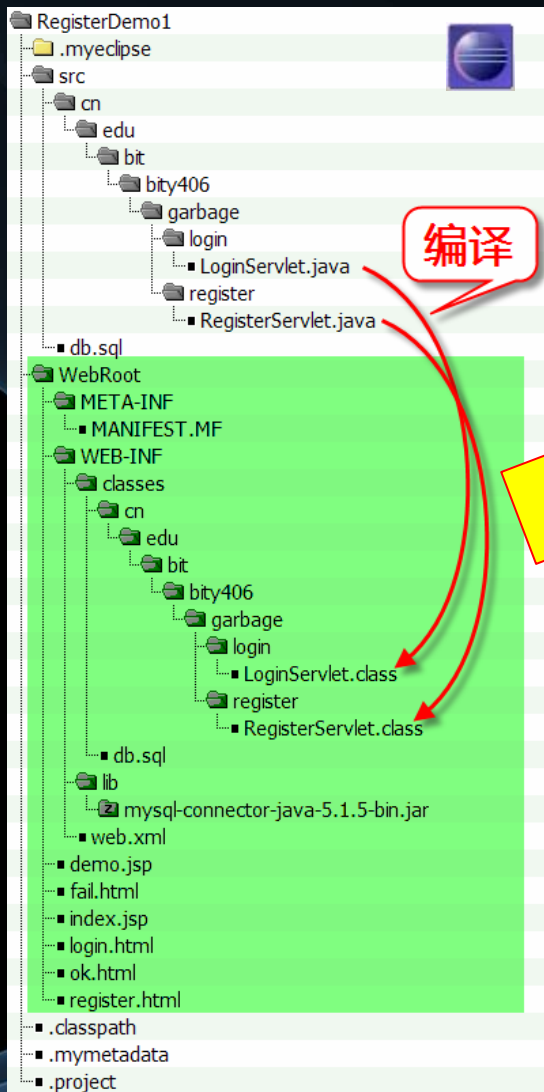
# HTTP – Response State



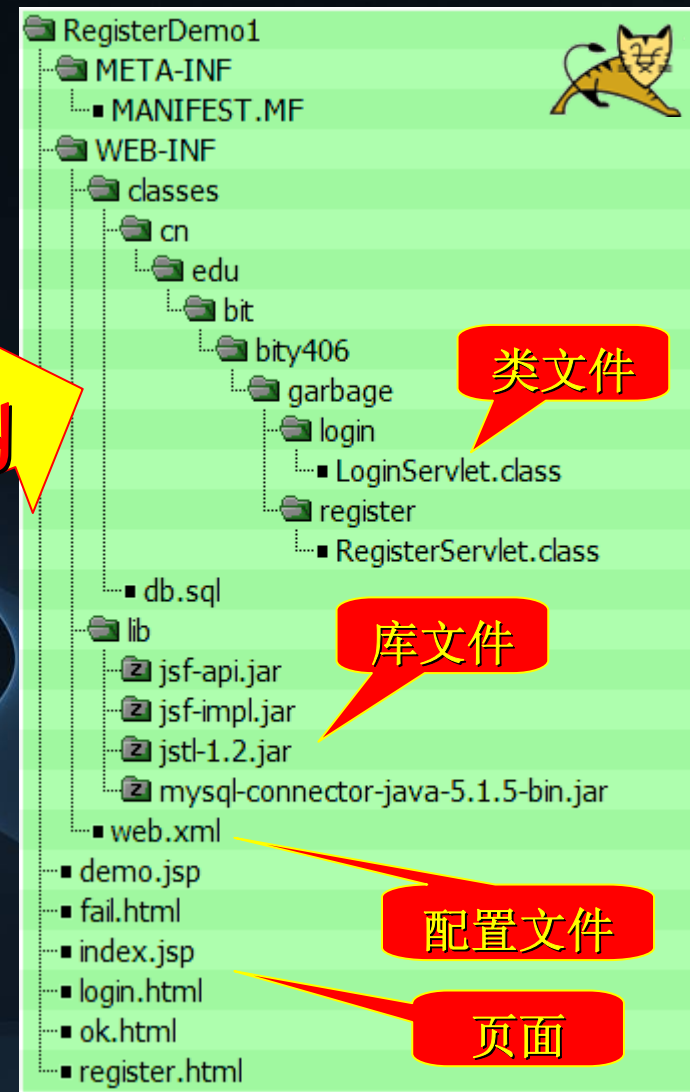
状态码	含义
200	ok
404	Not Found
500	Internal Server Error
503	Service Unavailable



# Deploy a WEB Project



Deploy



# web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <servlet>
    <servlet-name>RegisterServlet</servlet-name>
    <servlet-class>cn.edu.bit.bity406.garbage.register.RegisterServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>cn.edu.bit.bity406.garbage.login.LoginServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>RegisterServlet</servlet-name>
    <url-pattern>/servlet/RegisterServlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/servlet/LoginServlet</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

网页数据如何提交给Servlet的

`<form action="/servlet/RegisterServlet" method="post">`

`<form action="/servlet/LoginServlet" method="post">`

站点首页

register.html - Microsoft Internet Explorer

Back Forward Stop Search Favorites

Address http://localhost:8080/RegisterDemo1/register.html

**Register!**

Username:

Password:

Sex: ☐ M ☐ F ☐ E

ID:

Email:

Nationality: China



# Ref.



