**成果三 Web数据库实验**

一、.实验目的

1、掌握Servlet的开发、配置

2、掌握Filter的开发与配置

3、熟悉JDBC以及DAO的概念及工作原理

4、能够熟练运用JDBC以及DAO模式对数据库进行访问，实现数据查询、添加、修改等常用操作。

二、实验要求

1、实验之前认真查阅相关资料，准备好实验方案。

2、完成实验报告和实验成果

三、实验内容

1、登录页面表单提交后转到Servlet进行处理，根据用户类型转到不同的页面。

2、请设计一种过滤器实现权限控制机制。如果用户进入Web应用没有登录时，要求用户必须进行登录页面。

3、使用JDBC技术进行数据库的连接与访问。

4、采用DAO设计模式开发。使用自己熟悉的网络数据库，设计一个小的系统（可以在之间的页面上实现），要求如下：

1. 至少包括2张数据表（字段数不少于4）；
2. 实现对2张表格的增、删、改、查操作。
3. 具有不同类型用户的登录控制页面。
4. 要求每一项操作对应与一张页面。

四、实验结果和截图。

1.使用的sevlet配置：

<servlet>

<servlet-name>dwr</servlet-name>

<servlet-class>dwr.MyDwrSpringServlet</servlet-class>

<init-param>

<param-name>debug</param-name>

<param-value>true</param-value>

</init-param>

<!-- 可能有时我们需要配置多个dwr.xml可以采用如下所示 <param-value>WEB-INF/dwr.xml,WEB-INF/dwr1.xml,WEB-INF/dwr2.xml</param-value> -->

<init-param>

<param-name>config-dwr</param-name>

<param-value>/WEB-INF/dwr.xml</param-value>

</init-param>

<!-- 允许跨域调用 -->

<init-param>

<param-name>allowScriptTagRemoting</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>allowGetForSafariButMakeForgeryEasier</param-name>

<param-value>true</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>dwr</servlet-name>

<url-pattern>/dwr/\*</url-pattern>

</servlet-mapping>

<!-- 验证码 -->

<servlet>

<servlet-name>RandomImageServlet</servlet-name>

<servlet-class>util.code.RandomImageServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>RandomImageServlet</servlet-name>

<url-pattern>/servlet/RandomImageServlet</url-pattern>

</servlet-mapping>

<!-- 获取公钥 -->

<servlet>

<servlet-name>PubKeyServlet</servlet-name>

<servlet-class>com.ztesoft.servlet.PubKeyServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>PubKeyServlet</servlet-name>

<url-pattern>/servlet/PubKeyServlet.do</url-pattern>

</servlet-mapping>

2.使用的登陆DWR过滤器配置：

<!-- 登录状态过滤 -->

<filter>

<filter-name>checkLogin</filter-name>

<filter-class>dwr.ajax.filter.LoginFilter</filter-class>

<init-param>

<param-name>signOnPage</param-name>

<param-value>/portal/index.jsp</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>checkLogin</filter-name>

<url-pattern>\*.jsp</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>checkLogin</filter-name>

<url-pattern>\*.html</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>checkLogin</filter-name>

<url-pattern>\*.htm</url-pattern>

</filter-mapping>

3.使用的JDBC配置：

<bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"*

value=*"com.microsoft.jdbc.sqlserver.SQLServerDriver"* />

<property name=*"url"* value=*"jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=upp\_db "* />

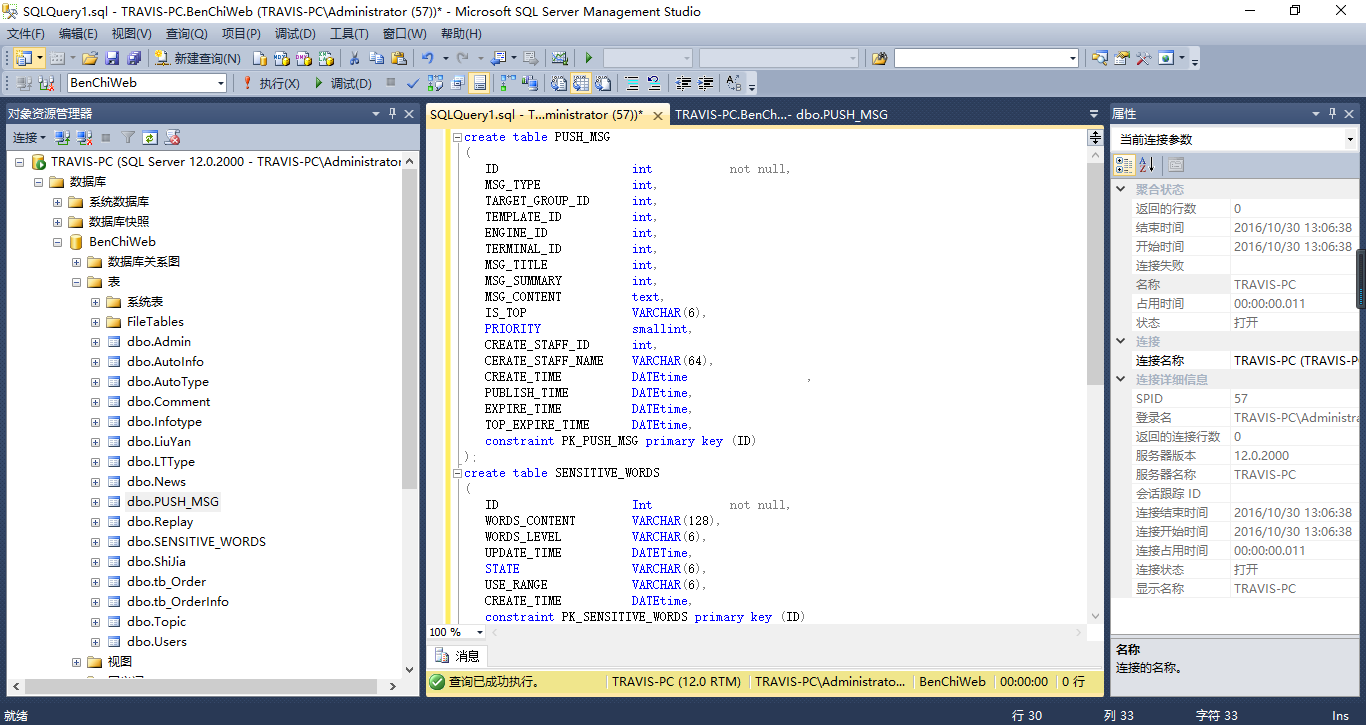
<property name=*"upp"* value=*"upp"*/>

</bean>

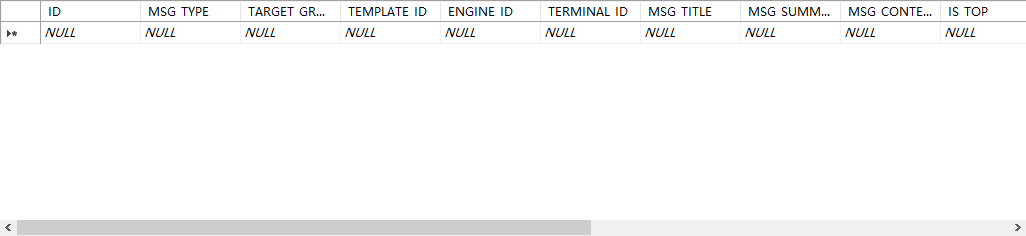
4.进行数据库操作的截图：

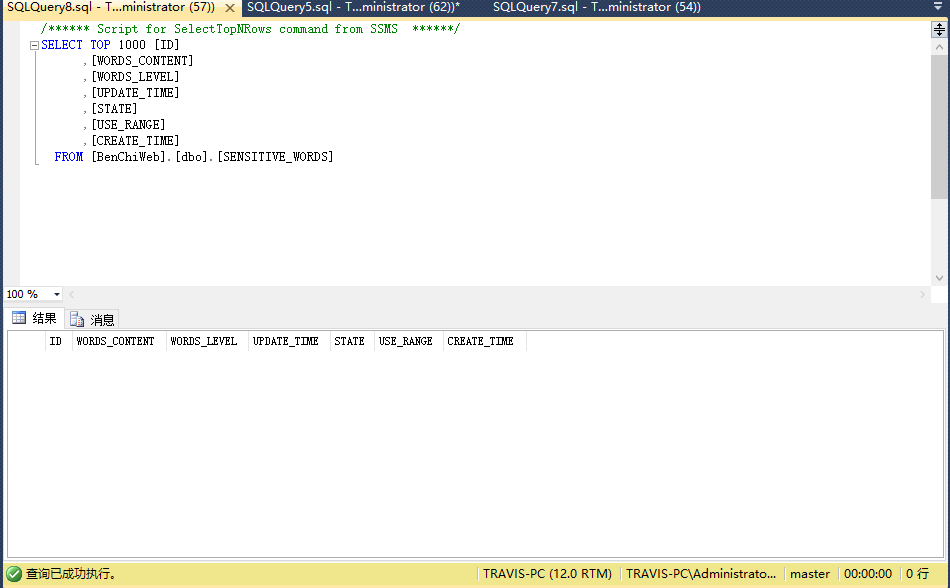
通过报告二附加的代码可以看到，JDBC操作都是封装在各自模块的DAO类中的：接下来我在SQL SERVER STUDIO中给大家演示一下数据库的操作：

建表的SQL：

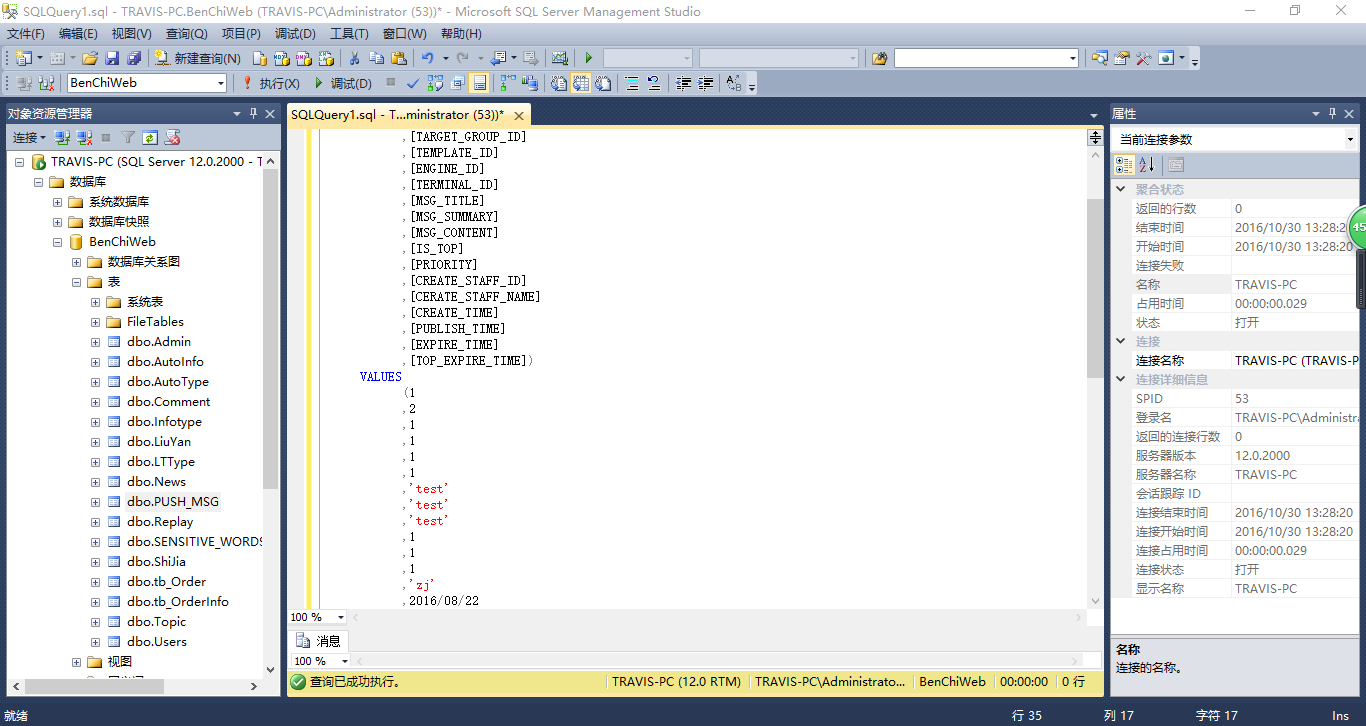


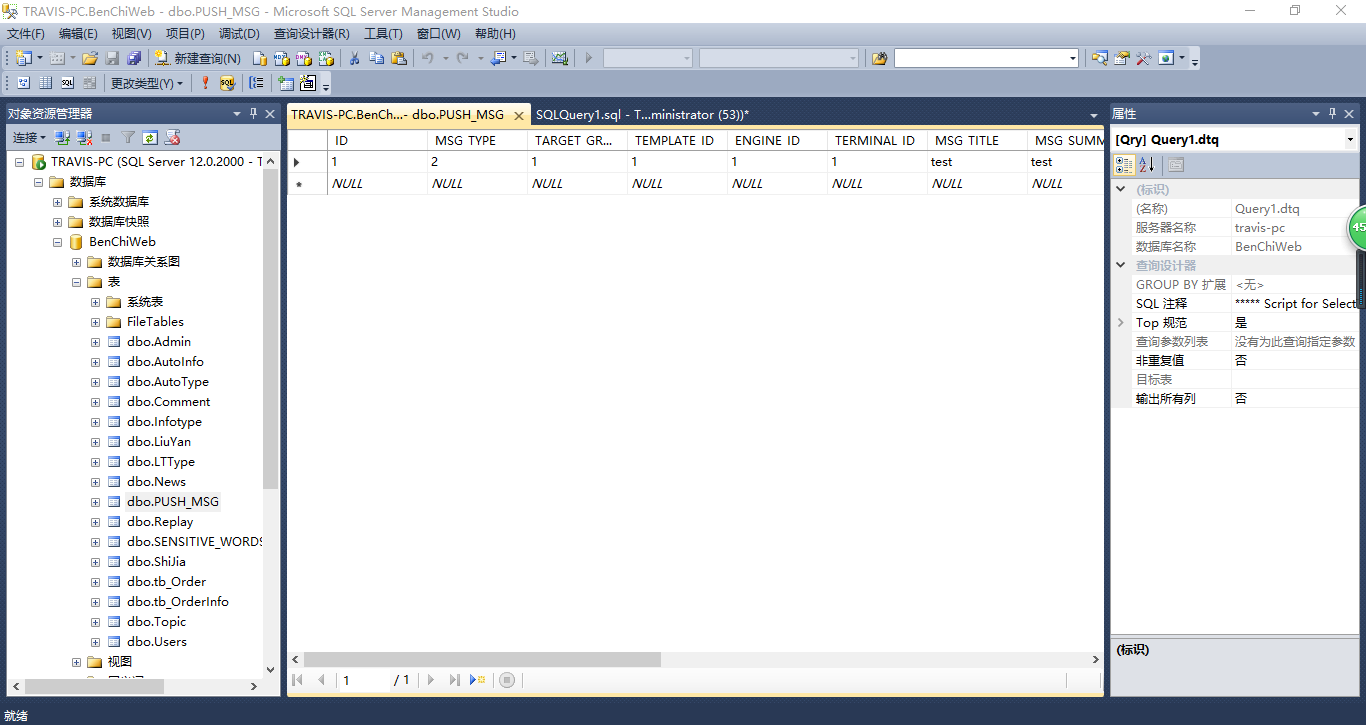
执行效果：



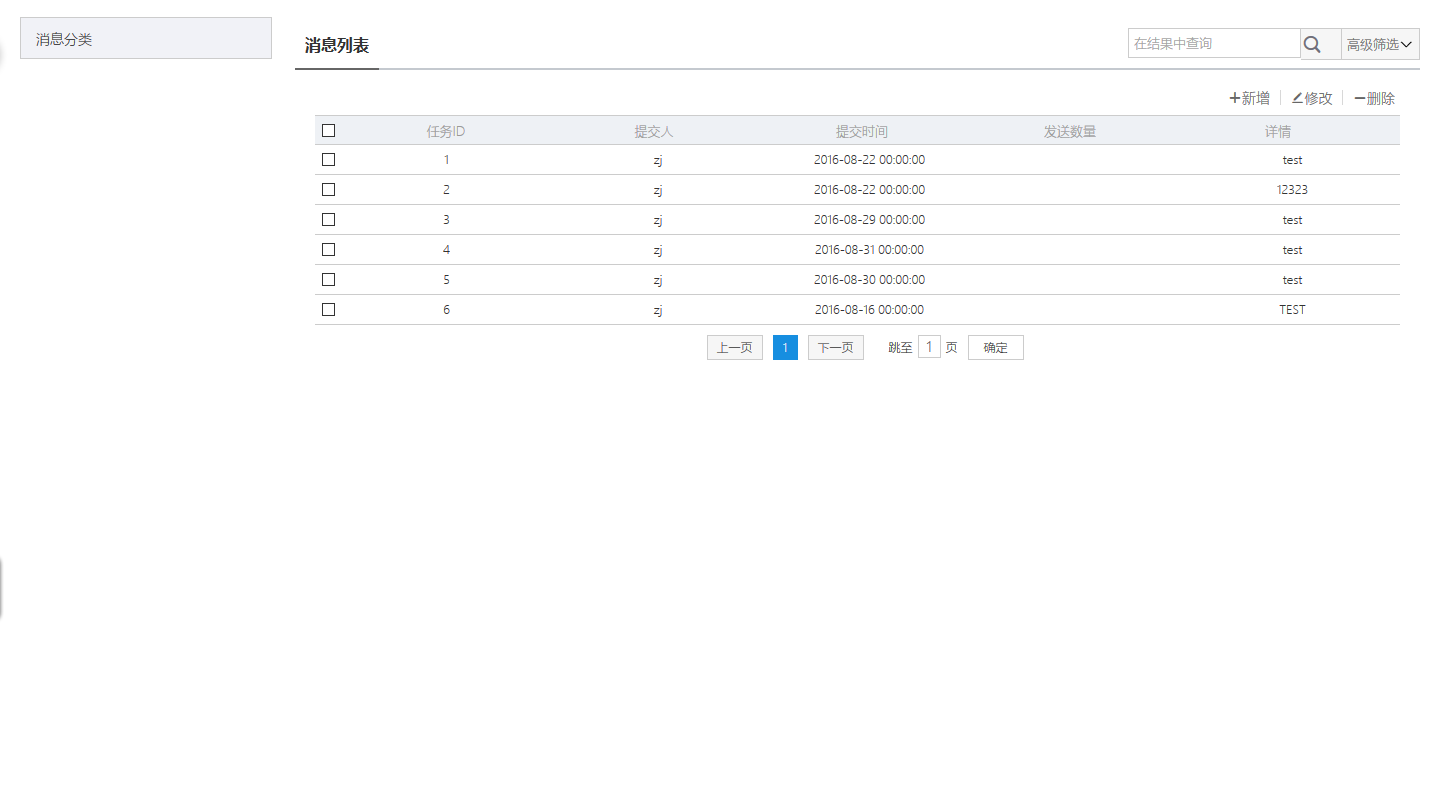


增加条目操作：INSERT INTO操作：

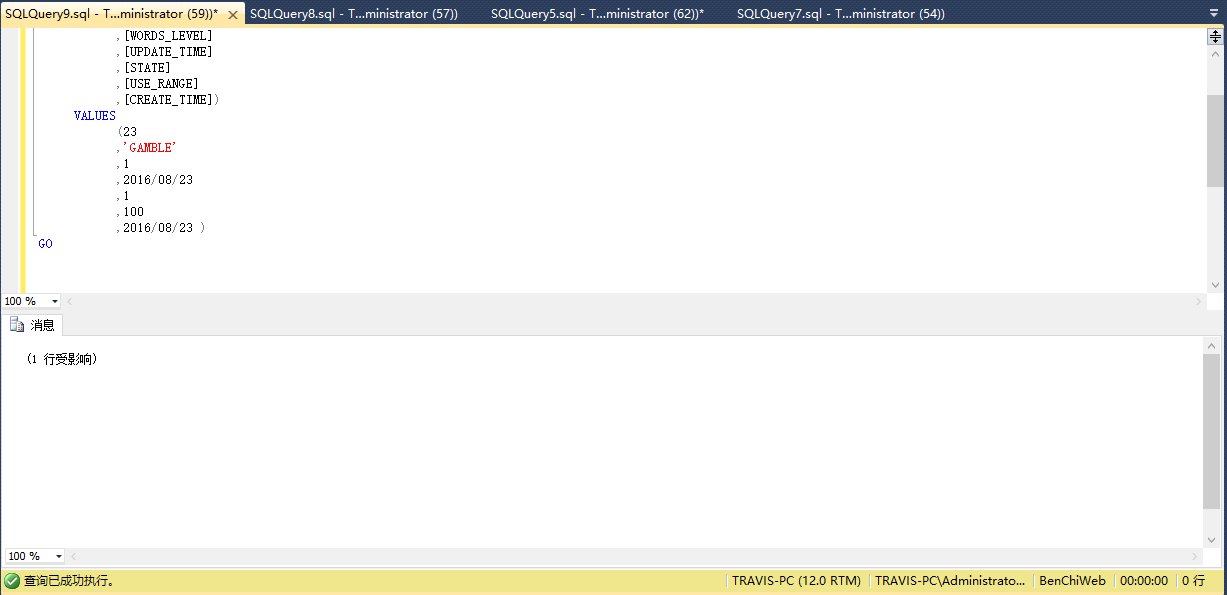




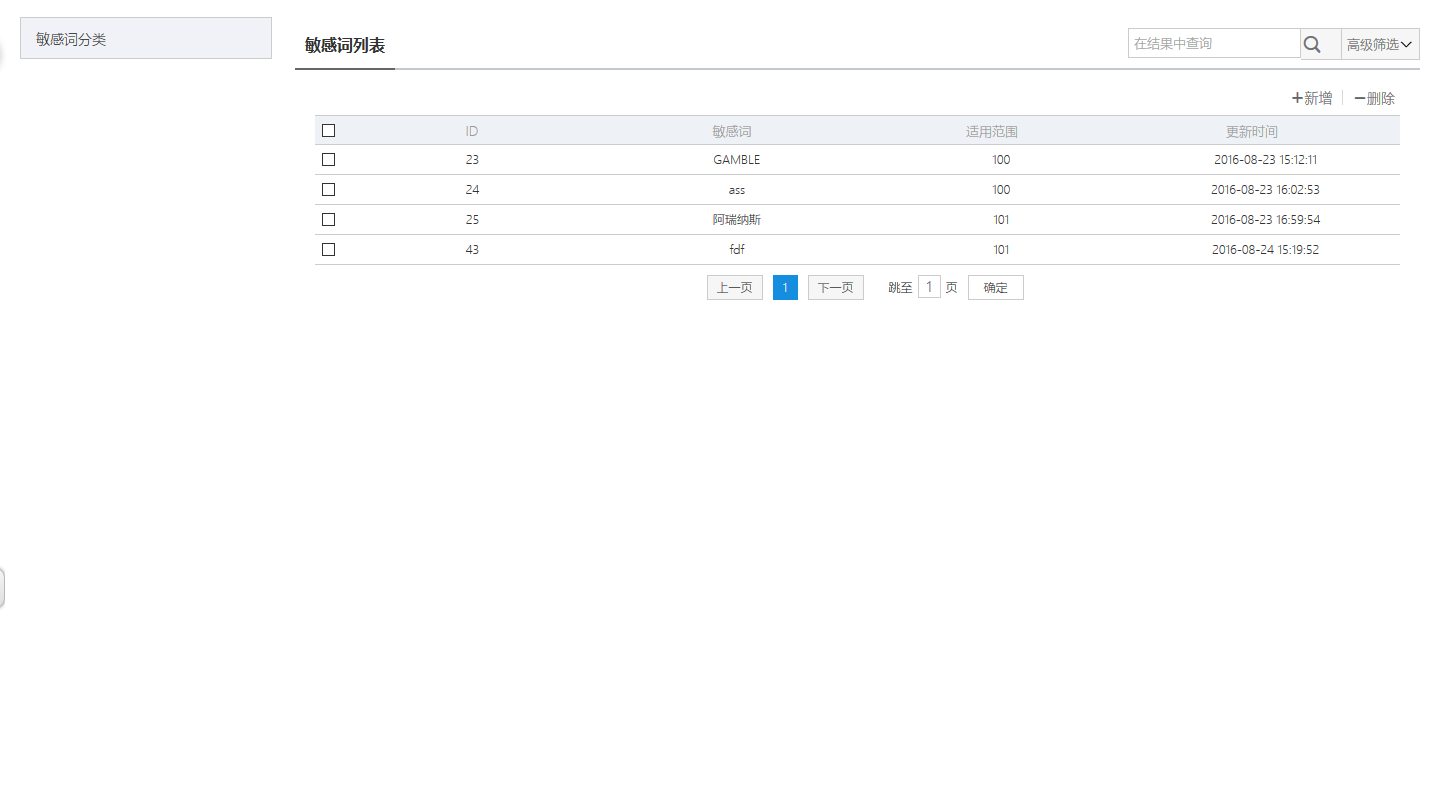
插入若干条记录后在web页面的显示结果：



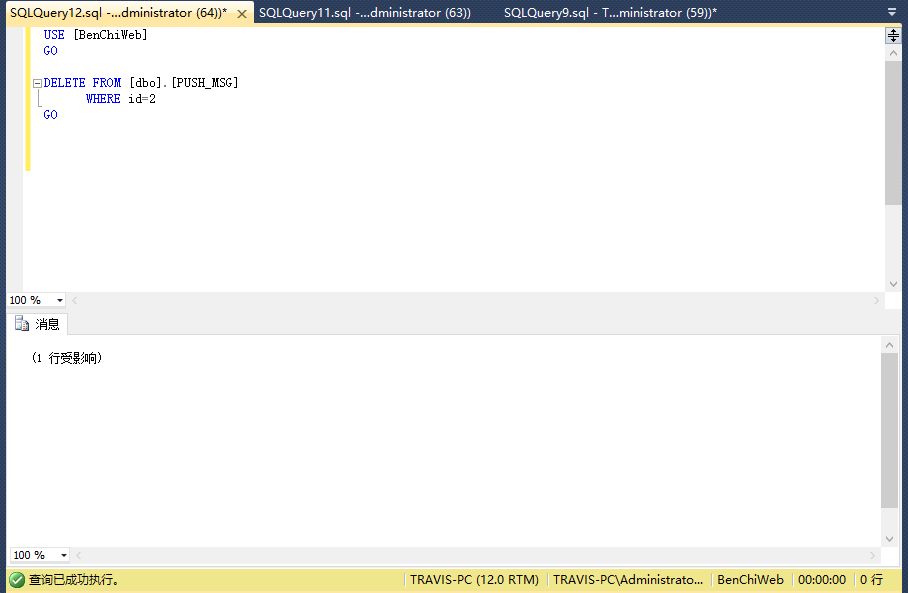
在SENSITIVE\_WORD表中也执行类似操作：

’

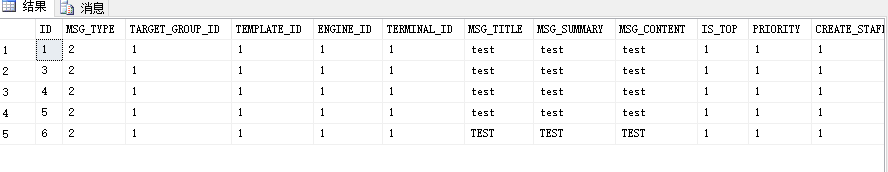
也可以在web页面中看到类似效果：



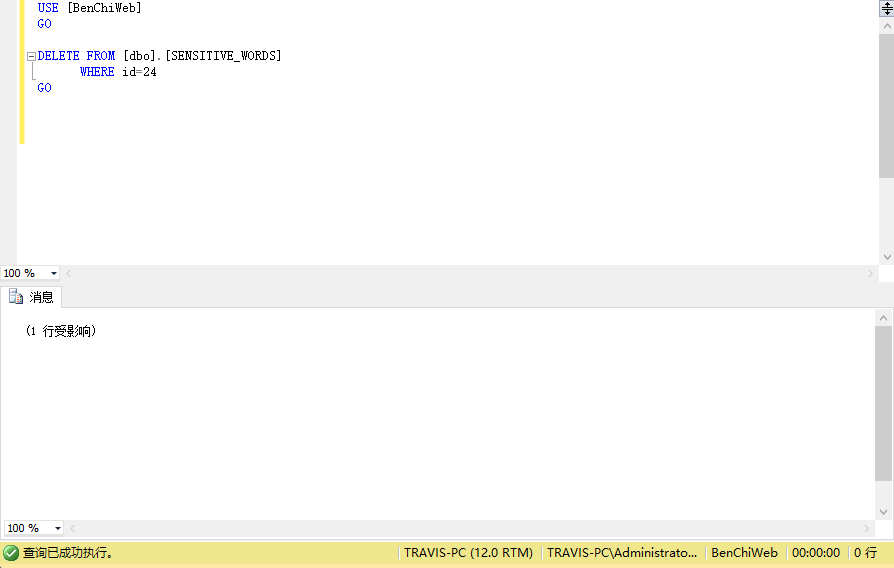
删除操作：（delete语句）



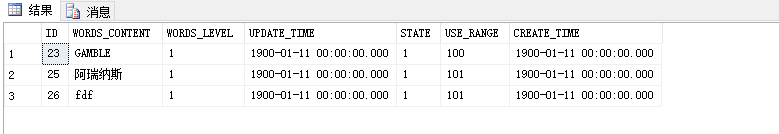
用select语句查看 id=2的条目已经没有了



同样，对sensitive\_word表进行类似操作：



ID=24的那行没有了



更新操作：UPDATE语句

