# 数据源和异常处理

# 数据库连接池

- 在实际应用开发中，特别是在WEB应用系统中，如果JSP、Servlet或EJB使用JDBC直接访问数据库中的数据，每一次数据访问请求都必须经历建立数据库连接、打开数据库、存取数据和关闭数据库连接等步骤，而连接并打开数据库是一件既消耗资源又费时的工作，如果频繁发生这种数据库操作，系统的性能必然会急剧下降，甚至会导致系统崩溃。数据库连接池技术是解决这个问题最常用的方法，在许多应用程序服务器（例如：Weblogic,WebSphere,JBoss）中，基本都提供了这项技术，无需自己编程，但是，深入了解这项技术是非常必要的。

- 数据库连接池技术的思想非常简单，将数据库连接作为对象存储在一个Vector对象中，一旦数据库连接建立后，不同的数据库访问请求就可以共享这些连接，这样，通过复用这些已经建立的数据库连接，可以克服上述缺点，极大地节省系统资源和时间

# 数据库连接池的主要操作

（1）建立数据库连接池对象（服务器启动）。

（2）按照事先指定的参数创建初始数量的数据库连接（即：空闲连接数）。

（3）对于一个数据库访问请求，直接从连接池中得到一个连接。如果数据库连接池对象中没有空闲的连接，且连接数没有达到最大（即：最大活跃连接数），创建一个新的数据库连接。

（4）存取数据库。

（5）关闭数据库，释放所有数据库连接（此时的关闭数据库连接，并非真正关闭，而是将其放入空闲队列中。如实际空闲连接数大于初始空闲连接数则释放连接）。

（6）释放数据库连接池对象（服务器停止、维护期间，释放数据库连接池对象，并释放所有连接）。

# 数据源需要的jar文件

- 如果使用struts框架提供的数据源，需要以下jar文件
  - commons-dbcp-1.2.2.jar
  - commons-pool-1.3.jar

# 配置struts-config.xml

```
<data-sources>
    <data-source type="org.apache.commons.dbcp.BasicDataSource">
            <set-property property="autoCommit" value="true" />
            <set-property property="description" value="my data source" />
            <set-property property="driverClassName"
    value="oracle.jdbc.driver.OracleDriver" />
            <set-property property="maxCount" value="10" />
            <set-property property="minCount" value="2" />
            <set-property property="username" value=“scott" />
            <set-property property="password" value=“tiger" />
            <set-property property="url"
                    value="jdbc:oracle:thin:@127.0.0.1:1521:orcl" />
    </data-source>
</data-sources>
```

# <data-source>配置解释

- autoCommit 自动提交
- description 描述
- driverClassName 驱动类完整类名
- maxCount 最大连接数
- minCount 最少闲置连接数
- username 用户名
- password 密码
- url 连接url

# 在Action中使用数据源

```
import javax.servlet.http.*;
import org.apache.struts.action.*;
import java.sql.*;
import javax.sql.DataSource;
public class MyDataSourceAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
                        HttpServletRequest request, HttpServletResponse response) throws
    Exception {
            //获得数据源
            DataSource ds = this.getDataSource(request);
            //获得数据库连接对象
            Connection conn = ds.getConnection();

            //将连接对象还给连接池
            conn.close();

            return null;
    }
}
```

# Struts异常处理

- Struts框架允许以配置的方式处理异常
- 可以配置<exception>标签来处理Action中发生的异常
- 示例

```
<global-exceptions>
        <exception key="global.error.msg" type="java.lang.Exception"
                path="/error.jsp"></exception>
</global-exceptions>
```

- 按照上面的配置，当任何Action中抛出异常的时候都会转发到一个error.jsp的错误页，并且传递global.error.msg的错误消息文本
- 在error.jsp中应该用<html:errors>标签显示错误消息