在OracleJDBC访问中加入Spring特性（5） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/143/2021_2022__E5_9C_A8OracleJ_c102_143569.htm 代码清单 14 中的使用基于 Spring 的代码来访问存储过程的示例演示了 org.springframework.jdbc.object.StoredProcedure 类的用法。（包含这个 StoredProcedure 类的 Spring 程序包也包含了除存储过程调用之外的其他类型的 SQL 语句的对象表示。关于其他 SQL 语句类型的对象程序包和对象表示的更多详细信息，请参见 Spring 的文档。） 代码清单 14 private class SalaryCalculator extends StoredProcedure{/** Name of procedure in database. */public static final String PROC_NAME = "salary_percentile". /*** Constructor for this StoredProcedure class.* @param ds Data Source. */public SalaryCalculator(DataSource ds) {setDataSource(ds).setSql(PROC_NAME).// Parameters should be declared in same order here that// they are declared in the stored procedure. declareParameter(new SqlParameter("salary", Types.DOUBLE)).declareParameter(new SqlParameter("low", Types.INTEGER)).declareParameter(new SqlParameter("high", Types.INTEGER)).declareParameter(new SqlOutParameter("percentile",Types.DOUBLE)).compile(). } /*** Execute stored procedure.* @return Results of running stored procedure. */public Map executeCalculation(double aSalary,int aLow,int aHigh) {Map inParameters = new HashMap().inParameters.put("salary", new Double(aSalary)).inParameters.put("low", new Integer(aLow)).inParameters.put("high", new Integer(aHigh)).Map out =

execute(inParameters). // Call on parent classreturn out. } }// . . .// Code below is all that is needed to call your Stored Procedure// created above. // . . .SalaryCalculator calcPercentile =new SalaryCalculator(this.myDataSource).Map calcResults =calcPercentile.executeCalculation(aSalary, aLow, aHigh). return ((Double)calcResults.get("percentile")).doubleValue().// . . .// remainder of class// . . .代码清单 14 中的程序在单个类中表示代码。该代码清单中显示的代码的主体是一个扩展了 Spring 的 StoredProcedure 的内部类 (SalaryCalculator)。这个由开发人员创建的类包装了代码清单 12 中显示的存储过程。调用 SalaryCalculator 类只需要几行代码。因此，SalaryCalculator 类对调用存储过程所涉及的大部分问题进行了抽象化。 虽然开发人员必须编写扩展 StoredProcedure 的类，但这么做（而不是直接编写存储过程访问）将带来相应的好处。一个好处是能够使用 Spring 的特有的非强制 DAO 和 JDBC 异常而不是通用的强制 SQLException。此外，正如代码代码清单所示，处理关闭资源的烦琐工作被抽象化了。 注意在将小于 0 的薪水值传给存储过程时代码清单 13 和代码清单 14 返回的结果的差异是很有趣的。在直接的 JDBC 的情况下（代码清单 13），将返回值 0.0，而 wasNull() 是确定结果是否真正为空的唯一方式。在基于 Spring 的代码中（代码清单 14），这种情况下将返回一个 Java null，无需 wasNull() 调用。 利用 Spring 访问 Oracle 对象 Spring Framework 的 JDBC 抽象可以与 Oracle 对象（例如在代码清单 15 中创建的对象）结合使用。 代码清单 15 CREATE OR REPLACE TYPE address_type AS OBJECT(STREET VARCHAR2(20),CITY

VARCHAR2(15),STATE CHAR(2),ZIP CHAR(5))./CREATE TABLE emp_address_table(EMP_NUM NUMBER(4),ADDRESS ADDRESS_TYPE).利用 JDBC 访问 Oracle 对象存在两种常用的方法。一种方法是将标准的 JDBC 接口 java.sql.Struct 与其 Oracle 驱动程序特有的类实施 oracle.sql.STRUCT 结合使用。第二种方法是创建与 Oracle 对象类型映射的 Java 类。将 Oracle 对象与 Oracle JDeveloper 10g IDE 和 JPublisher 结合使用是件轻而易举的事。 使用 java.sql.Struct 方法或 JPublisher 方法的一个有趣的“ 副作用 ”：如果您想在这些对象中访问数据，那么必须处理 SQLException。例如，使用 java.sql.Struct 方式，getAttributes() 方法将抛出 SQLException。同样，JDeveloper/JPublisher 创建的 Java 类也将包含抛出 SQLException 的方法。访问这些 Java 对象的开发人员将必须处理这些 SQLException，或者可以使用 Spring（如代码清单 16 和 17 所示）。 代码清单 16 String queryStr = "SELECT address FROM emp_address_table WHERE " "emp_num = " aEmpNum. // aEmpNum passed infinal List addresses = new ArrayList(). JdbcTemplate jt = new JdbcTemplate(this.myDataSource).jt.query(queryStr,new RowCallbackHandler() {public void processRow(ResultSet rs)throws SQLException {// The Struct and ResultSet methods throw// SQLException, so throws above is necessaryjava.sql.Struct address =(java.sql.Struct) rs.getObject(1).String street =address.getAttributes()[0].toString().String city =address.getAttributes()[1].toString().String state =address.getAttributes()[2].toString().String zipCode

=address.getAttributes()[3].toString().String addressStr = street ", " city ", " state " " zipCode.addresses.add(addressStr). } }).代码清单 17 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com