Joshua McCann  
DAO LAB 2

|  |
| --- |
| package astontech.dao.mysql;  */\*\*  \* Created by Joshua.McCann on 6/29/2017.  \*/* public abstract class Procedures {  final static String *GetClient* = "{ call usp\_GetClient(?,?) }";  final static String *GetEmail* = "{ call usp\_GetEmail(?,?) }";  final static String *GetEmployee* = "{ call usp\_GetEmployee(?,?,?) }";  final static String *GetPerson* = "{call usp\_GetPerson(?,?)}";  final static String *GetPhone* = "{ call usp\_GetPhone(?,?) }";  final static String *ExecClient* = "{ call usp\_ExecuteClient(?,?,?) }";  final static String *ExecEmail* = "{call usp\_ExecuteEmail(?,?,?,?,?)}";  final static String *ExecEmployee* = "{ call usp\_ExecuteEmployee(?,?,?,?,?,?) }";  final static String *ExecPerson* = "{call usp\_ExecutePerson(?,?,?,?,?,?,?)}";  final static String *ExecPhone* = "{ call usp\_ExecutePhone(?,?,?,?,?,?,?,?) }";  ; } |
| package astontech.dao.mysql;  import astontech.dao.ClientDAO; import com.astontech.bo.Client;  import java.sql.CallableStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Types; import java.util.ArrayList; import java.util.List;  */\*\*  \* Created by Joshua.McCann on 6/28/2017.  \*/* public class ClientDAOImpl extends MySQL implements ClientDAO {   @Override  public Client getClientById(int clientId) {  *Connect*();  Client client = null;  try{  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*GetClient*);   cStmnt.setInt(1, *GET\_BY\_ID*);  cStmnt.setInt(2, clientId);   ResultSet rs = cStmnt.executeQuery();   while(rs.next()){  client = new Client();  client = *HydrateClient*(rs);  }  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }   return client;  }   @Override  public List<Client> getClientList() {  *Connect*();  List<Client> clientList = new ArrayList<Client>();  try {  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*GetClient*);   cStmnt.setInt(1, *GET\_COLLECTION*);  cStmnt.setInt(2, 0);   ResultSet rs = cStmnt.executeQuery();   while(rs.next()){  clientList.add(*HydrateClient*(rs));  }   } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }   return clientList;  }   @Override  public int insertClient(Client client) {  int id = 0;  try {  id = *ModifyClient*(*INSERT*, client);   } catch (SQLException SqlEx) {  *logger*.error(SqlEx);  }   return id;  }   @Override  public boolean updateClient(Client client) {  int id = 0;  try {  id = *ModifyClient*(*UPDATE*, client);   } catch (SQLException SqlEx) {  *logger*.error(SqlEx);  }   return id>0;  }   @Override  public boolean deleteClient(int clientId) {  int id = 0;  Client client = new Client();  client.setClientId(clientId);  try {  id = *ModifyClient*(*DELETE*, client);   } catch (SQLException SqlEx) {  *logger*.error(SqlEx);  }   return id>0;  }   private static Client HydrateClient(ResultSet rs) throws SQLException{  /\*  \* ClientId int index 1  \* ClientName varchar index 2  \* CreateDate date index 3  \*/   Client client = new Client();  client.setClientId(rs.getInt(1));  client.setClientName(rs.getString(2));  client.setCreateDate(rs.getDate(3));   return client;  }   private static int ModifyClient(int key, Client client) throws SQLException{  *Connect*();  //`usp\_ExecuteClient`(queryId int, clientId int, clientName varchar(100))   CallableStatement cStmnt = *conn*.prepareCall(Procedures.*ExecClient*);   cStmnt.setInt(1, key);  if(client.getClientId()==0)  cStmnt.setNull(2, Types.*INTEGER*);  else  cStmnt.setInt(2, client.getClientId());  cStmnt.setString(3, client.getClientName());   ResultSet rs = cStmnt.executeQuery();   if(rs.next())  return rs.getInt(1);  else  return 0;  } } |
| package astontech.dao.mysql;  import astontech.dao.EmailDAO; import com.astontech.bo.Email; import com.astontech.bo.Employee; import com.astontech.bo.EntityType;  import java.sql.CallableStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Types; import java.util.ArrayList; import java.util.List;  */\*\*  \* Created by Joshua.McCann on 6/28/2017.  \*/* public class EmailDAOImpl extends MySQL implements EmailDAO {   @Override  public Email getEmailById(int emailId) {  *Connect*();  Email email = null;  try{  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*GetEmail*);   cStmnt.setInt(1, *GET\_BY\_ID*);  cStmnt.setInt(2, emailId);   ResultSet rs = cStmnt.executeQuery();   if(rs.next()){  email = new Email();  email = HydrateEmail(rs);  }   } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }  return email;   }   @Override  public List<Email> getEmailList() {  *Connect*();  List<Email> emailList = new ArrayList<Email>();   try{  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*GetEmail*);   cStmnt.setInt(1, *GET\_COLLECTION*);  cStmnt.setInt(2, 0);   ResultSet rs = cStmnt.executeQuery();   while(rs.next()){  emailList.add(HydrateEmail(rs));  }   } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }   return emailList;  }   @Override  public int insertEmail(Email email) {  int id = 0;  try{  id = *ModifyEmail*(*INSERT*, email);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }  return id;  }   @Override  public boolean updateEmail(Email email) {  int id = 0;  try{  id = *ModifyEmail*(*UPDATE*, email);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }  return id>0;  }   @Override  public boolean deleteEmail(int emailId) {  int id = 0;  Email email = new Email();  email.setEmailId(emailId);  email.setEmailType(new EntityType());  email.setEmailEmployee(new Employee());   try{  id = *ModifyEmail*(*DELETE*, email);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }  return id>0;   }   public Email HydrateEmail(ResultSet rs) throws SQLException{  /\*  \* EmailId int index 1  \* EmailAddress varchar index 2  \* EmployeeId int index 3  \* EntityTypeId int index 4  \*/   Email email = new Email();  EntityType entityType = new EntityType();  entityType.setEntityTypeId(rs.getInt(4));   email.setEmailId(rs.getInt(1));  email.setEmailAddress(rs.getString(2));  email.setEmailEmployee(new EmployeeDAOImpl().getEmployeeById(rs.getInt(3)));  email.setEmailType(entityType);   return email;  }   public static int ModifyEmail(int key, Email email) throws SQLException {  *Connect*();  CallableStatement cStmnt = *conn*.prepareCall(Procedures.ExecEmail);   cStmnt.setInt(1,key);  cStmnt.setString(2,email.getEmailAddress());  if(email.getEmailEmployee().getEmployeeId()==0)  cStmnt.setNull(3, Types.*INTEGER*);  else  cStmnt.setInt(3,email.getEmailEmployee().getEmployeeId());  if(email.getEmailType().getEntityTypeId()==0)  cStmnt.setNull(4, Types.*INTEGER*);  else  cStmnt.setInt(4,email.getEmailType().getEntityTypeId());  if(email.getEmailId()==0)  cStmnt.setNull(5, Types.*INTEGER*);  else  cStmnt.setInt(5,email.getEmailId());   ResultSet rs = cStmnt.executeQuery();  if(rs.next())  return rs.getInt(1);  else  return 0;  } } |
| package astontech.dao.mysql;  import astontech.dao.EmployeeDAO; import com.astontech.bo.Employee; import com.astontech.bo.Person; import common.helpers.DateHelper;  import java.sql.CallableStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Types; import java.util.ArrayList; import java.util.List;  */\*\*  \* Created by Joshua.McCann on 6/28/2017.  \*/* public class EmployeeDAOImpl extends MySQL implements EmployeeDAO {   @Override  public Employee getEmployeeById(int employeeId) {  *Connect*();  Employee employee = null;   try{  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*GetEmployee*);   cStmnt.setInt(1, *GET\_BY\_ID*);  cStmnt.setInt(2, employeeId);  cStmnt.setInt(3, 0);   ResultSet rs = cStmnt.executeQuery();   if(rs.next()){  employee = new Employee();  employee = HydrateEmployee(rs);  }   } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }   return employee;  }   @Override  public List<Employee> getEmployeeList() {  *Connect*();  List<Employee> employeeList = new ArrayList<Employee>();   try{  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*GetEmployee*);   cStmnt.setInt(1, *GET\_COLLECTION*);  cStmnt.setInt(2, 0);  cStmnt.setInt(3, 0);   ResultSet rs = cStmnt.executeQuery();   while(rs.next()){  employeeList.add(HydrateEmployee(rs));  }   } catch (SQLException SqlEx){  *logger*.error(SqlEx);  }   return employeeList;  }   @Override  public int insertEmployee(Employee employee) {  int id;  try{  id = *ModifyEmployee*(*INSERT*, employee);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  id = 0;  }  return id;  }   @Override  public boolean updateEmployee(Employee employee) {  int id;  try{  id = *ModifyEmployee*(*UPDATE*, employee);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  id = 0;  }  return id>0;  }   @Override  public boolean deleteEmployee(int employeeId) {  int id;  Employee employee = new Employee();  employee.setEmployeeId(employeeId);  try{  id = *ModifyEmployee*(*DELETE*, employee);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  id = 0;  }  return id>0;  }   public Employee HydrateEmployee(ResultSet rs) throws SQLException {  /\*  \* EmployeeId int index 1  \* HireDate Date index 2  \* TermDate Date index 3  \* BirthDate Date index 4  \* PersonId int index 5  \* CreateDate Date index 6  \*/   Person person = new PersonDAOImpl().getPersonById(rs.getInt(5));   Employee employee = new Employee();  employee.setEmployeeId(rs.getInt(1));  employee.setHireDate(rs.getDate(2));  employee.setTermDate(rs.getDate(3));  employee.setBirthDate(rs.getDate(4));  employee.setPersonId(rs.getInt(5));  employee.setFirstName(person.getFirstName());  employee.setLastName(person.getLastName());  employee.setDisplayFirstName(person.getDisplayFirstName());  employee.setCreateDate(rs.getDate(6));   return employee;  }   public static int ModifyEmployee(int key, Employee employee) throws SQLException{  *Connect*();  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*ExecEmployee*);   cStmnt.setInt(1,key);  if(employee.getEmployeeId()==0)  cStmnt.setNull(2, Types.*INTEGER*);  else  cStmnt.setInt(2,employee.getEmployeeId());  if(employee.getHireDate()==null)  cStmnt.setNull(3,Types.*DATE*);  else  cStmnt.setDate(3,DateHelper.*utilDateToSqlDate*(employee.getHireDate()));  if(employee.getTermDate()==null)  cStmnt.setNull(4, Types.*DATE*);  else  cStmnt.setDate(4,DateHelper.*utilDateToSqlDate*(employee.getTermDate()));  if(employee.getBirthDate()==null)  cStmnt.setNull(5, Types.*DATE*);  else  cStmnt.setDate(5,DateHelper.*utilDateToSqlDate*(employee.getBirthDate()));  if(employee.getPersonId()==0)  cStmnt.setNull(6, Types.*INTEGER*);  else  cStmnt.setInt(6,employee.getPersonId());   ResultSet rs = cStmnt.executeQuery();  if(rs.next())  return rs.getInt(1);  else  return 0;  } } |
| package astontech.dao.mysql;  import astontech.dao.PhoneDAO; import com.astontech.bo.EntityType; import com.astontech.bo.Phone;  import java.sql.CallableStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Types; import java.util.ArrayList; import java.util.List;  */\*\*  \* Created by Joshua.McCann on 6/28/2017.  \*/* public class PhoneDAOImpl extends MySQL implements PhoneDAO{   @Override  public Phone getPhoneById(int phoneId) {  *Connect*();  Phone phone = null;  try {  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*GetPhone*);   cStmnt.setInt(1,*GET\_BY\_ID*);  cStmnt.setInt(2,phoneId);   ResultSet rs = cStmnt.executeQuery();   if(rs.next()){  phone = new Phone();  phone = *HydratePhone*(rs);  }   } catch (SQLException SqlEx) {  *logger*.error(SqlEx);  }  return phone;  }   @Override  public List<Phone> getPhoneList() {  *Connect*();  List<Phone> phoneList = new ArrayList<Phone>();  try {  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*GetPhone*);   cStmnt.setInt(1,*GET\_COLLECTION*);  cStmnt.setInt(2,0);   ResultSet rs = cStmnt.executeQuery();   while(rs.next()){  phoneList.add(*HydratePhone*(rs));  }   } catch (SQLException SqlEx) {  *logger*.error(SqlEx);  }  return phoneList;  }   @Override  public int insertPhone(Phone phone) {  int id;  try{  id = *ModifyPhone*(*INSERT*, phone);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  return 0;  }  return id;  }   @Override  public boolean updatePhone(Phone phone) {  int id;  try{  id = *ModifyPhone*(*UPDATE*, phone);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  id = 0;  }  return id>0;  }   @Override  public boolean deletePhone(int phoneId) {   int id;  Phone phone = new Phone();  phone.setPhoneId(phoneId);  try{  id = *ModifyPhone*(*DELETE*, phone);  } catch (SQLException SqlEx){  *logger*.error(SqlEx);  id = 0;  }  return id>0;  }   private static Phone HydratePhone(ResultSet rs) throws SQLException{  /\*  \* PhoneId int index 1  \* EntityTypeId int index 2  \* ClientId int index 3  \* PersonId int index 4  \* AreaCode int index 5  \* PhoneNumber int index 6  \* PhoneNumberPost int index 7  \*/   EntityType entityType = new EntityType();  entityType.setEntityTypeId(rs.getInt(2));   Phone phone = new Phone();  phone.setPhoneId(rs.getInt(1));  phone.setPhoneType(entityType);  phone.setPhoneClient(new ClientDAOImpl().getClientById(rs.getInt(3)));  phone.setPhonePerson(new PersonDAOImpl().getPersonById(rs.getInt(4)));  phone.setAreaCode(rs.getInt(5));  phone.setPhoneNumber(rs.getInt(6));  phone.setPhoneNumberPost(rs.getInt(7));   return phone;  }   public static int ModifyPhone(int key, Phone phone) throws SQLException{  *Connect*();  //`usp\_ExecutePhone`(queryId, phoneId, entityTypeId, clientId, personId, areaCode, phoneNumber, phoneNumberPost)  CallableStatement cStmnt = *conn*.prepareCall(Procedures.*ExecPhone*);   cStmnt.setInt(1, key);  if(phone.getPhoneId()==0)  cStmnt.setNull(2, Types.*INTEGER*);  else  cStmnt.setInt(2, phone.getPhoneId());   if(phone.getPhoneType().getEntityTypeId()==0)  cStmnt.setNull(3, Types.*INTEGER*);  else  cStmnt.setInt(3, phone.getPhoneType().getEntityTypeId());   if(phone.getPhoneClient().getClientId()==0)  cStmnt.setNull(4, Types.*INTEGER*);  else  cStmnt.setInt(4, phone.getPhoneClient().getClientId());   if(phone.getPhonePerson().getPersonId()==0)  cStmnt.setNull(5, Types.*INTEGER*);  else  cStmnt.setInt(5, phone.getPhonePerson().getPersonId());   if(phone.getAreaCode()==0)  cStmnt.setNull(6, Types.*INTEGER*);  else  cStmnt.setInt(6, phone.getAreaCode());   if(phone.getPhoneNumber()==0)  cStmnt.setNull(7, Types.*INTEGER*);  else  cStmnt.setInt(7, phone.getPhoneNumber());   if(phone.getPhoneNumberPost()==0)  cStmnt.setNull(8, Types.*INTEGER*);  else  cStmnt.setInt(8, phone.getPhoneNumberPost());   ResultSet rs = cStmnt.executeQuery();  if(rs.next())  return rs.getInt(1);  else  return 0;  } } |