# **源代码**

## **main.cpp**

1. #include <iostream>
2. #include <regex>
3. #include <mysqlx/xdevapi.h>
4. #include "services/service.h"
5. #include "data\_processing.h"
6. using namespace std;
7. */\*\**
8. \* @brief 员工管理系统的主要功能
9. \*
10. \* 此函数是程序的入口点。它显示欢迎消息，然后进入一个无限循环，不断提示用户输入对员工数据执行操作的命令
11. \*
12. \* @return 该函数不返回值。当用户退出程序时选择退出系统
13. \*/
14. int main() {
15. cout << ">> 筱锋公司员工信息管理系统 <<" << endl;
16. cout << "欢迎使用本系统，系统正在启动中......" << endl;
17. while (true) {
18. cout << "========================" << endl;
19. cout << "1. 员工信息查询" << endl;
20. cout << "2. 员工信息录入" << endl;
21. cout << "3. 员工信息修改" << endl;
22. cout << "4. 员工信息删除" << endl;
23. cout << "5. 退出系统" << endl;
24. cout << "========================" << endl;
25. *// 输入数据*
26. std::string input\_string;
27. cout << "请输入选项: ";
28. cin >> input\_string;
29. if (regex\_match(input\_string, regex("^[0-9]+$"))) {
30. switch (stoi(input\_string)) {
31. case 1: {
32. *// 查询员工信息*
33. processing::select\_user();
34. cout << "[系统] 员工系统查询完成" << endl;
35. break;
36. }
37. case 2: {
38. *// 添加员工信息*
39. if (processing::add\_user()) {
40. cout << "[系统] 员工信息添加成功！" << endl;
41. } else {
42. cout << "[系统] 员工信息添加失败！" << endl;
43. }
44. break;
45. }
46. case 3: {
47. *// 修改员工信息*
48. if (processing::edit\_user()) {
49. cout << "[系统] 员工信息修改成功！" << endl;
50. } else {
51. cout << "[系统] 员工信息修改失败！" << endl;
52. }
53. break;
54. }
55. case 4: {
56. *// 删除员工信息*
57. if (processing::delete\_user()) {
58. cout << "[系统] 员工信息删除成功！" << endl;
59. } else {
60. cout << "[系统] 员工信息删除失败！" << endl;
61. }
62. break;
63. }
64. case 5: {
65. *// 退出系统*
66. cout << "[系统] 感谢您的使用，再见！" << endl;
67. exit(0);
68. break;
69. }
70. default:
71. *// 错误输入*
72. break;
73. }
74. cout << endl;
75. } else {
76. cout << "输入错误，请重新输入！" << endl;
77. }
78. }
79. }

## **processing\_util.cpp**

1. *//*
2. *// Created by 筱锋 on 24-4-19.*
3. *//*
4. #include <iostream>
5. #include <chrono>
6. #include <cstdlib>
7. namespace util {
8. */\*\**
9. \* 生成一个随机数
10. \* 这个随机数是生成用于工号使用的随机数
11. \*
12. \* @param size
13. \* @return
14. \*/
15. static std::string generate\_number(const int size) {
16. srand((unsigned)time(nullptr));
17. std::string number = "XF";
18. for (int i = 0; i < size-2; i++)
19. number =  number + std::to\_string(rand()%10);
20. return number;
21. }
22. */\*\**
23. \* getCurrentTimestamp
24. \* 获取当前时间戳
25. \*
26. \* @return 返回当前时间戳
27. \*/
28. static unsigned long long getCurrentTimestamp() {
29. return std::chrono::duration\_cast<std::chrono::milliseconds>(std::chrono::system\_clock::now()
30. .time\_since\_epoch()).count();
31. }
32. }

## **data\_processing.h**

1. *//*
2. *// Created by 筱锋 on 24-4-12.*
3. *//*
4. #ifndef WORKERMANAGER\_DATA\_PROCESSING\_H
5. #define WORKERMANAGER\_DATA\_PROCESSING\_H
6. #include <iostream>
7. using namespace std;
8. namespace processing {
9. *// 跳入员工信息的查询*
10. *// 该方法用于查询员工信息*
11. *// 返回值为员工信息*
12. *// 该方法会调用 get\_user 方法*
13. void select\_user() {
14. cout << "========== select\_user ==========" << endl;
15. cout << "您正在进行员工信息的查询;" << endl;
16. cout << "用户的查询可以按姓名、电话号码、工号查询;" << endl;
17. cout << "========== select\_user ==========" << endl;
18. cout << "请输入需要查询的信息：";
19. get\_vo::user\_select user\_select\_vo;
20. cin >> user\_select\_vo.user;
21. service::user::get\_user(user\_select\_vo);
22. }
23. *// 添加员工信息*
24. *// 该方法用于添加员工信息*
25. *// 返回值为是否添加成功*
26. *// 该方法会调用 add\_user 方法*
27. bool add\_user() {
28. get\_vo::user\_create user\_create\_vo;
29. cout << "========== add\_user ==========" << endl;
30. cout << "您正在进行员工信息的添加;" << endl;
31. cout << "请输入员工的姓名：";
32. cin >> user\_create\_vo.user\_name;
33. cout << "请输入员工的性别：";
34. cin >> user\_create\_vo.user\_sex;
35. cout << "请输入员工的部门：";
36. cin >> user\_create\_vo.user\_department;
37. cout << "请输入员工的工资：";
38. cin >> user\_create\_vo.user\_wages;
39. cout << "请输入员工的级别：";
40. cin >> user\_create\_vo.user\_level;
41. cout << "请输入员工职位：";
42. cin >> user\_create\_vo.user\_position;
43. cout << "========== add\_user ==========" << endl;
44. service::user user\_service;
45. return user\_service.add\_user(&user\_create\_vo);
46. }
47. *// 修改员工信息*
48. *// 该方法用于修改员工信息*
49. *// 返回值为是否修改成功*
50. *// 该方法会调用 select\_user 方法*
51. bool edit\_user() {
52. service::user user\_service;
53. *// 请输入您需要修改的员工信息*
54. string getUsername;
55. cout << "请输入需要修改用户的用户名: ";
56. cin >> getUsername;
57. std::string userInfo = user\_service.get\_user\_by\_username(getUsername);
58. if (!userInfo.empty()) {
59. int check\_edit = 0;
60. *// 是否确认修改*
61. cout << "是否确认修改员工信息？(1 是，0 不是）：";
62. cin >> check\_edit;
63. if (!check\_edit) {
64. cout << "您已取消修改员工信息!" << endl;
65. return false;
66. }
67. *// 开始修改员工信息*
68. get\_vo::user\_edit user\_edit\_vo;
69. cout << "========== edit\_user ==========" << endl;
70. cout << "您正在进行员工 [" << userInfo << "] 信息的修改;" << endl;
71. user\_edit\_vo.user\_number = userInfo;
72. cout << "请输入员工的姓名：";
73. cin >> user\_edit\_vo.user\_name;
74. cout << "请输入员工的性别：";
75. cin >> user\_edit\_vo.user\_sex;
76. cout << "请输入员工的部门：";
77. cin >> user\_edit\_vo.user\_department;
78. cout << "请输入员工的工资：";
79. cin >> user\_edit\_vo.user\_wages;
80. cout << "请输入员工的级别：";
81. cin >> user\_edit\_vo.user\_level;
82. cout << "请输入员工的职位：";
83. cin >> user\_edit\_vo.user\_position;
84. cout << "========== edit\_user ==========" << endl;
85. return user\_service.edit\_user(&user\_edit\_vo);
86. } else {
87. cout << "[系统] 用户不存在" << endl;
88. return false;
89. }
90. }
91. *// 删除员工信息*
92. *// 该方法用于删除员工信息*
93. *// 返回值为是否删除成功*
94. *// 该方法会调用 select\_user 方法*
95. bool delete\_user() {
96. *// 请输入您需要删除的员工信息*
97. cout << "========== delete\_user ==========" << endl;
98. cout << "您正在进行员工信息的删除;" << endl;
99. std::string userName;
100. cout << "请输入员工的姓名：";
101. cin >> userName;
102. cout << "========== delete\_user ==========" << endl;
103. service::user user\_service;
104. std::string userInfo = user\_service.get\_user\_by\_username(userName);
105. if (!userInfo.empty()) {
106. int check\_delete = 0;
107. *// 是否确认删除*
108. cout << "是否确认删除员工信息？(1 是，0 不是）：";
109. cin >> check\_delete;
110. if (!check\_delete) {
111. cout << "您已取消删除员工信息!" << endl;
112. return false;
113. }
114. *// 开始删除员工信息*
115. return user\_service.delete\_user(userInfo);
116. } else {
117. cout << "未找到员工信息！" << endl;
118. return false;
119. }
120. }
122. }
123. #endif *//WORKERMANAGER\_DATA\_PROCESSING\_H*

## **user\_do.cpp**

1. *//*
2. *// Created by 筱锋 on 24-4-28.*
3. *//*
4. #include <iostream>
5. namespace entity {
6. */\*\**
7. \* UserDO 类
8. \* 该类用于存储用户信息
9. \*/
10. class UserDO {
11. public:
12. std::string user\_number;
13. std::string user\_name;
14. std::string user\_sex;
15. std::string user\_department;
16. std::string user\_wages;
17. std::string user\_level;
18. std::string user\_position;
19. };
20. }

## **user\_info.cpp**

1. *//*
2. *// Created by 筱锋 on 24-4-12.*
3. *//*
4. #include <iostream>
5. namespace back\_vo {
6. class user\_info {
7. public:
8. std::string user\_number;
9. std::string user\_name;
10. std::string user\_sex;
11. std::string user\_department;
12. std::string user\_wages;
13. std::string user\_level;
14. std::string user\_posts;
15. };
16. }

## **user\_create.cpp**

1. *//*
2. *// Created by 筱锋 on 24-4-12.*
3. *//*
4. #include <iostream>
5. namespace get\_vo {
6. class user\_create {
7. public:
8. *// 用户姓名*
9. std::string user\_name;
10. *// 用户性别*
11. std::string user\_sex;
12. *// 用户部门*
13. std::string user\_department;
14. *// 用户工资*
15. std::string user\_wages;
16. *// 用户级别*
17. std::string user\_level;
18. *// 用户职位*
19. std::string user\_position;
20. };
21. class user\_edit {
22. public:
23. *// 用户号*
24. std::string user\_number;
25. *// 用户姓名*
26. std::string user\_name;
27. *// 用户性别*
28. std::string user\_sex;
29. *// 用户部门*
30. std::string user\_department;
31. *// 用户工资*
32. std::string user\_wages;
33. *// 用户级别*
34. std::string user\_level;
35. *// 用户职位*
36. std::string user\_position;
37. };
38. }

## **user\_select.cpp**

1. *//*
2. *// Created by 筱锋 on 24-4-12.*
3. *//*
4. #include <iostream>
5. namespace get\_vo {
6. class user\_select {
7. public:
8. std::string user;
9. };
10. }

## **service.h**

1. *//*
2. *// Created by 筱锋 on 24-4-12.*
3. *//*
4. #include <iostream>
5. #include <mysqlx/xdevapi.h>
6. #include "../model/vo/back/user\_info.cpp"
7. #include "../model/vo/get/user\_create.cpp"
8. #include "../model/vo/get/user\_select.cpp"
9. #include "../model/entity/user\_do.cpp"
10. #include "../processing\_util.cpp"
11. using namespace util;
12. using namespace std;
13. using namespace entity;
14. namespace service {
15. class user {
16. public:
17. bool add\_user(get\_vo::user\_create \*user\_info);
18. bool edit\_user(const get\_vo::user\_edit \*user\_info);
19. bool delete\_user(const std::string number);
20. static void get\_user(const get\_vo::user\_select &user\_info);
21. std::string get\_user\_by\_username(const std::string &username);
22. };
23. }

## **service.cpp**

1. *//*
2. *// Created by 筱锋 on 24-4-12.*
3. *//*
4. #include "service.h"
5. */\*\**
6. \* add\_user 方法实现
7. \* 该方法用于添加用户信息
8. \* 返回值为是否添加成功
9. \*
10. \* @param user\_info 用户信息
11. \* @return 是否添加成功
12. \*/
13. bool service::user::add\_user(get\_vo::user\_create \*user\_info) {
14. *// 对用户的信息进行检查*
15. if (user\_info->user\_name.empty()) {
16. std::cout << "[服务] 用户姓名为空" << std::endl;
17. return false;
18. }
19. *// 对用户的信息进行检查*
20. int sex;
21. if (user\_info->user\_sex.empty()) {
22. std::cout << "[服务] 用户性别为空" << std::endl;
23. return false;
24. } else {
25. if (user\_info->user\_sex == "男") {
26. sex = 0;
27. } else {
28. sex = 1;
29. }
30. }
31. *// 对用户的信息进行检查*
32. if (user\_info->user\_department.empty()) {
33. std::cout << "[服务] 用户部门为空" << std::endl;
34. return false;
35. }
36. *// 对用户的信息进行检查*
37. if (user\_info->user\_wages.empty()) {
38. std::cout << "[服务] 用户工资为空" << std::endl;
39. return false;
40. }
41. *// 对用户的信息进行检查*
42. if (user\_info->user\_level.empty()) {
43. std::cout << "[服务] 用户级别为空" << std::endl;
44. return false;
45. }
46. *// 数据转换*
47. try {
48. mysqlx::Session sqlSession("localhost", 33060, "root", "");
49. std::string sql =
50. "INSERT INTO work\_manager.xf\_user (number, name, sex, department, money, level, posts) VALUES ('"
51. + util::generate\_number(10) + "', '"
52. + user\_info->user\_name + "', "
53. + to\_string(sex) + ", '"
54. + user\_info->user\_department + "', '"
55. + user\_info->user\_wages + "', '"
56. + user\_info->user\_level + "', '"
57. + user\_info->user\_position + "')";
58. sqlSession
59. .sql(sql)
60. .execute();
61. return true;
62. } catch (const mysqlx::Error &e) {
63. cerr << "Error: " << e.what() << endl;
64. } catch (std::exception &e) {
65. cerr << "Error: " << e.what() << endl;
66. } catch (...) {
67. cerr << "Unknown error!" << endl;
68. }
69. return false;
70. }
71. */\*\**
72. \* get\_user 方法实现
73. \* 该方法用于查询用户信息
74. \* 返回值为用户信息
75. \*
76. \* @param user\_info 用户信息
77. \* @return 用户信息
78. \*/
79. void service::user::get\_user(const get\_vo::user\_select &user\_info) {
80. mysqlx::Session sqlSession("localhost", 33060, "root", "");
81. mysqlx::RowResult getData = sqlSession.sql(
82. "SELECT \* FROM work\_manager.xf\_user WHERE name = '" + user\_info.user + "' LIMIT 1")
83. .execute();
84. while (auto row = getData.fetchOne()) {
85. *// 展示用户的信息*
86. cout << "员工信息如下：" << endl;
87. cout << "员工编号：" << row[1] << endl;
88. cout << "员工姓名：" << row[2] << endl;
89. cout << "员工性别：" << row[3] << endl;
90. cout << "员工部门：" << row[4] << endl;
91. cout << "员工工资：" << row[5] << endl;
92. cout << "员工级别：" << row[6] << endl;
93. cout << "员工职位：" << row[7] << endl;
94. }
95. }
96. */\*\**
97. \* edit\_user 方法实现
98. \* 该方法用于修改用户信息
99. \* 返回值为是否修改成功
100. \*
101. \* @param user\_info 用户信息
102. \* @return 是否修改成功
103. \*/
104. bool service::user::edit\_user(const get\_vo::user\_edit \*user\_info) {
105. *// 对用户的信息进行检查*
106. if (user\_info->user\_name.empty()) {
107. std::cout << "[服务] 用户姓名为空" << std::endl;
108. return false;
109. }
110. *// 对用户的信息进行检查*
111. int sex;
112. if (user\_info->user\_sex.empty()) {
113. std::cout << "[服务] 用户性别为空" << std::endl;
114. return false;
115. } else {
116. if (user\_info->user\_sex == "男") {
117. sex = 0;
118. } else {
119. sex = 1;
120. }
121. }
122. *// 对用户的信息进行检查*
123. if (user\_info->user\_department.empty()) {
124. std::cout << "[服务] 用户部门为空" << std::endl;
125. return false;
126. }
127. *// 对用户的信息进行检查*
128. if (user\_info->user\_wages.empty()) {
129. std::cout << "[服务] 用户工资为空" << std::endl;
130. return false;
131. }
132. *// 对用户的信息进行检查*
133. if (user\_info->user\_level.empty()) {
134. std::cout << "[服务] 用户级别为空" << std::endl;
135. return false;
136. }
137. *// 数据转换*
138. try {
139. mysqlx::Session sqlSession("localhost", 33060, "root", "");
140. std::string sql =
141. "UPDATE work\_manager.xf\_user SET"
142. " name='" + user\_info->user\_name + "',"
143. " sex='" + to\_string(sex) + "',"
144. " department='" + user\_info->user\_department + "',"
145. " money='" + user\_info->user\_wages + "',"
146. " level='" + user\_info->user\_level + "',"
147. " posts='" + user\_info->user\_position + "' WHERE number='" + user\_info->user\_number + "'";
148. sqlSession
149. .sql(sql)
150. .execute();
151. return true;
152. } catch (const mysqlx::Error &e) {
153. cerr << "Error: " << e.what() << endl;
154. } catch (std::exception &e) {
155. cerr << "Error: " << e.what() << endl;
156. } catch (...) {
157. cerr << "Unknown error!" << endl;
158. }
159. return false;
160. }
161. */\*\**
162. \* delete\_user 方法实现
163. \* 该方法用于删除用户信息
164. \* 返回值为是否删除成功
165. \*
166. \* @param user\_info 用户信息
167. \* @return 是否删除成功
168. \*/
169. bool service::user::delete\_user(const std::string number) {
170. try {
171. mysqlx::Session sqlSession("localhost", 33060, "root", "");
172. std::string sql =
173. "DELETE FROM work\_manager.xf\_user WHERE number = '"+ number +"'";
174. sqlSession
175. .sql(sql)
176. .execute();
177. return true;
178. } catch (const mysqlx::Error &e) {
179. cerr << "Error: " << e.what() << endl;
180. } catch (std::exception &e) {
181. cerr << "Error: " << e.what() << endl;
182. } catch (...) {
183. cerr << "Unknown error!" << endl;
184. }
185. return false;
186. }
187. */\*\**
188. \* get\_user\_by\_username 方法实现
189. \* 该方法用于查询用户信息
190. \* 返回值为用户信息
191. \*
192. \* @param user\_info 用户信息
193. \* @return 用户信息
194. \*/
195. std::string service::user::get\_user\_by\_username(const std::string &username) {
196. mysqlx::Session sqlSession("localhost", 33060, "root", "");
197. mysqlx::RowResult getData = sqlSession.sql(
198. "SELECT number FROM work\_manager.xf\_user WHERE name = '" + username + "' LIMIT 1")
199. .execute();
200. cout << "有操作" << endl;
201. while (auto row = getData.fetchOne()) {
202. cout << "有数据：" << row[0] << endl;
203. mysqlx::Value getNumber = row[0];
204. std::string \*strValue;
205. if (getNumber.getType() == mysqlx::Value::Type::STRING) {
206. return getNumber.get<std::string>();
207. } else {
208. return "";
209. }
210. }
211. return "";
212. }