****

**Web开发与应用**

**实验报告**

|  |
| --- |
| 专业班级：计科2101 |
| 学 号：202112123 |
| 姓 名： 宋子墨 |

**东北大学秦皇岛分校计算机与通信工程学院**

**2024**

|  |  |
| --- | --- |
| **【实验编号】** | **1** |
| **【实验名称】** | **WEB开发环境与网页设计 学时：2** |
| **【实验目的】**  1）熟悉开发环境的使用。  2）掌握 Tomcat 安装及配置。  3）运行一个简单网站  4）熟悉CSS/DIV/HTML/XML的使用  **【实验内容】**  测试Tomcat，编写一个简单的网站。  **【关键代码】**   1. <!DOCTYPE html> 2. <html lang="en"> 3. <head> 4. <meta charset="UTF-8"> 5. <meta name="viewport" content="width=device-width, initial-scale=1.0"> 6. <title>Document</title> 7. <script src="./js/axios.js"></script> 8. <script src="./lib/vue.js"></script> 9. <script src="./lib/elementui.js"></script> 10. <link rel="stylesheet" href="./lib/elementui.css"> 11. <script src="./content/utf8-net/ueditor.config.js"></script> 12. <script src="./content/utf8-net/ueditor.all.min.js"></script> 13. <script src="./content/utf8-net/lang/zh-cn/zh-cn.js"></script> 14. </head> 15. <body> 16. <div id="postDetail"> 17. <div class="send-pad"> 18. <el-button type="primary" size="small">回帖</el-button> 19. </div> 20. <div class="reply-content"> 21. <div class="post-title"> 22. <div class="left"> 23. 查看: 1000 | 回复: 1000 24. </div> 25. <div class="right"> 26. 帖子标题 27. </div> 28. </div> 29. </div> 30. <ul> 31. <li> 32. <div class="content-item"> 33. <div class="left"> 34. <div class="user-no"> 35. 225225225225 36. </div> 37. <div class="user-info"> 38. <image 39. src="./assets/images/user-head-default.png" 40. width="80px" 41. height="80px"> 42. </image> 43. <p> 44. 第一个用户名 45. </p> 46. <p> 47. 其他信息 48. </p> 49. </div> 50. </div> 51. <div class="right"> 52. <div class="sent-time"> 53. 发表于: 2024-4-16 21:12:08 54. </div> 55. <div class="content-detail"> 56. 一段回复内容 57. </div> 58. <div class="content-foot"> 59. <span>回复</span> 60. <span>点赞(1)</span> 61. <span>踩(0)</span> 62. </div> 63. </div> 64. </li> 65. <li> 66. <div class="content-item"> 67. <div class="left"> 68. <div class="user-no"> 69. 225225225225 70. </div> 71. <div class="user-info"> 72. <image 73. src="./assets/images/user-head-default.png" 74. width="80px" 75. height="80px"> 76. </image> 77. <p> 78. 第一个用户名 79. </p> 80. <p> 81. 其他信息 82. </p> 83. </div> 84. </div> 85. <div class="right"> 86. <div class="sent-time"> 87. 发表于: 2024-4-16 21:12:08 88. </div> 89. <div class="content-detail"> 90. 一段回复内容 91. </div> 92. <div class="content-foot"> 93. <span>回复</span> 94. <span>点赞(1)</span> 95. <span>踩(0)</span> 96. </div> 97. </div> 98. </li> 99. <li> 100. <div class="content-item"> 101. <div class="left"> 102. <div class="user-no"> 103. 225225225225 104. </div> 105. <div class="user-info"> 106. <image 107. src="./assets/images/user-head-default.png" 108. width="80px" 109. height="80px"> 110. </image> 111. <p> 112. 第一个用户名 113. </p> 114. <p> 115. 其他信息 116. </p> 117. </div> 118. </div> 119. <div class="right"> 120. <div class="sent-time"> 121. 发表于: 2024-4-16 21:12:08 122. </div> 123. <div class="content-detail"> 124. 一段回复内容 125. </div> 126. <div class="content-foot"> 127. <span>回复</span> 128. <span>点赞(1)</span> 129. <span>踩(0)</span> 130. </div> 131. </div> 132. </li> 133. </ul> 134. <div class="reply-page"> 135. <el-pagination 136. background layout="prev, pager, next" 137. :page-size="30" 138. :total="1000"> 139. </el-pagination> 140. </div> 141. <div class="write-pad"> 142. <div id="content"></div> 143. <p> 144. <el-button type="primary" size="small">回帖</el-button> 145. </p> 146. </div> 147. </div> 148. <script> 149. new Vue({ 150. el: "#postDetail", 151. mounted() { 152. var ue = UE.getEditor('content', { 153. initialFrameHeight: '100%' 154. }) 155. } 156. }) 157. </script> 158. <style> 159. \* { 160. margin: 0; 161. padding: 0; 162. box-sizing: border-box; 163. } 164. #postDetail { 165. padding: 0 16px; 166. } 167. .send-pad { 168. height: 50px; 169. margin-top: 20px; 170. } 171. .reply-page { 172. text-align: right; 173. margin: 16px 0; 174. } 175. .reply-content .post-title { 176. background-color: #e5edf2; 177. height: 42px; 178. line-height: 36px; 179. font: 14px; 180. border-bottom: 6px solid #c2d5e3; 181. } 182. .reply-content .post-title .left { 183. width: 200px; 184. float: left; 185. } 186. ul { 187. list-style: none; 188. } 189. .content-item { 190. border-bottom: 6px solid #c2d5e3; 191. } 192. .content-item .left { 193. width: 200px; 194. background-color: #e5edf2; 195. padding-bottom: 6px; 196. float: left; 197. } 198. .content-item .left .user-no { 199. height: 36px; 200. line-height: 36px; 201. border-bottom: 1px #ccc dashed; 202. padding-left: 16px; 203. } 204. .content-item .left .user-info { 205. padding-top: 16px; 206. text-align: center; 207. } 208. .content-item .left .user-info img { 209. border: 5px #fff solid; 210. width: 90px; 211. height: 90px; 212. } 213. .content-item .left .user-info p { 214. text-align-last: left; 215. padding-left: 55px; 216. font-size: 14px; 217. padding-bottom: 10px; 218. } 219. .content-item .right { 220. min-height: 210px; 221. background-color: #fff; 222. font-size: 14px; 223. overflow: hidden; 224. padding: 0 6px; 225. } 226. .content-item .right .sent-time { 227. height: 36px; 228. line-height: 36px; 229. border-bottom: 1px #ccc dashed; 230. } 231. .content-item .right .content-detail { 232. min-height: 138px; 233. } 234. .content-item .right .content-foot { 235. height: 36px; 236. line-height: 36px; 237. border-top: 1px #ccc dashed; 238. } 239. .content-item .right .content-foot span { 240. margin-right: 16px; 241. } 242. .write-pad { 243. width: 90%; 244. margin: auto; 245. } 246. .write-pad #content { 247. height: 180px; 248. } 249. .write-pad p { 250. margin-top: 106px; 251. margin-bottom: 36px; 252. } 253. </style> 254. </body> 255. </html>   **【实验结果及说明】**  使用Html+Vue+ElementUI+Ueditor编写了一个论坛回帖页面：    图 1 Html页面 | |

|  |  |
| --- | --- |
| **【实验编号】** | **2** |
| **【实验名称】** | **Request对象的使用 学时：2** |
| **【实验目的】**  1）编写具有Request、response、session对象的程序实验。  2）掌握怎样在JSP中使用内置的Request对象、怎样使用Response对象动态响应用户  的请求、怎样使用Session对象存储和用户有关的数据。  **【实验内容】**  编写一个JSP页面，该页面提供一个表单，用户通过表单输入两个数和四则运算符号提交给该页面；编写两个JSP页面，一个页面提交数字给另一个页面，另一个页面使用response对象作出动态响应。  **【关键代码】**  Todo.razor：   1. @page "/todo" 2. <h3>Todo</h3> 3. @inject NavigationManager NavigationManager 4. <div> 5. <label for="number1">Number 1:</label> 6. <input type="number" id="number1" @bind="@number1" /> 7. </div> 8. <div> 9. <label for="number2">Number 2:</label> 10. <input type="number" id="number2" @bind="@number2" /> 11. </div> 12. <div> 13. <label for="selectedOperator">Operator:</label> 14. <select id="selectedOperator" @bind="@selectedOperator"> 15. <option value="+">+</option> 16. <option value="-">-</option> 17. <option value="\*">\*</option> 18. <option value="/">/</option> 19. </select> 20. </div> 21. <button class="btn btn-primary" @onclick="Calculate">Calculate</button> 22. @code { 23. private int number1; 24. private int number2; 25. private string selectedOperator = "+"; 26. private int result; 27. private void Calculate() 28. { 29. *// Perform the calculation based on the selected operator* 30. switch (selectedOperator) 31. { 32. case "+": 33. result = number1 + number2; 34. break; 35. case "-": 36. result = number1 - number2; 37. break; 38. case "\*": 39. result = number1 \* number2; 40. break; 41. case "/": 42. result = number1 / number2; 43. break; 44. default: 45. result = 0; 46. break; 47. } 48. NavigationManager.NavigateTo("/result/" + result.ToString()); 49. } 50. }   Result.razor：   1. @page "/result/{result}" 2. @inject NavigationManager NavigationManager 3. <h3>Result</h3> 4. <p>The result is: @result</p> 5. <button @onclick="GoBack">Go Back</button> 6. @code { 7. [Parameter] 8. public String result { get; set; } = "0"; 9. private void GoBack() 10. { 11. NavigationManager.NavigateTo("/todo"); 12. } 13. }   **【实验结果及说明】**  基于Blazor编写了计算四则运算的组件页面。在示例Blazor项目的基础上，添加了Todo.razor和Result.razor。功能展示如下：    图 2 输入数字与运算符页面    图 3 结果页面 | |

|  |  |
| --- | --- |
| **【实验编号】** | **3** |
| **【实验名称】** | **Servlet与JDBC的使用 学时：2** |
| **【实验目的】**  1)学习使用servlet，  2)采用JDBC访问mysql数据库，将程序部署在服务器上，观察程序的运行.  3)掌握通过JDBC的方式实现对数据库表中数据的增删改。  **【实验内容】**  将工程jdbc导入Eclipse开发环境，并部署到tomcat服务器；创建数据库和表；启动tomcat服务器，输入“http://localhost:8080/jdbc”运行程序，查看运行效果。  **【关键代码】**  LoginApiController.cs：   1. using Microsoft.AspNetCore.Mvc; 2. using WebapiDemo.Dal; 3. namespace WebapiDemo.Controllers 4. { 5. [ApiController] 6. [Route("api/[controller]")] 7. public class LoginApiController : ControllerBase 8. { 9. [HttpGet] 10. public string Get(string UserNo, string password) 11. { 12. UserDal userDal = new(); 13. if (userDal.GetUserByUserNoAndPassword(UserNo, password)) 14. { 15. return "登录成功"; 16. } 17. else 18. { 19. return "用户名或密码错误"; 20. } 21. } 22. [HttpPost] 23. public string Insert(string userNo, string userName, int userLevel, string password) 24. { 25. UserDal userDal = new(); 26. return userDal.AddUser(userNo, userName, userLevel, password) > 0 ? "插入成功" : "插入失败"; 27. } 28. [HttpPut] 29. public string Update( 30. int id, 31. [FromForm] string? userNo, 32. [FromForm] string? userName, 33. [FromForm] int? userLevel, 34. [FromForm] string? password 35. ) 36. { 37. UserDal userDal = new(); 38. return userDal.UpdateUser(id, userNo, userName, userLevel, password) > 0 39. ? "更新成功": "更新失败";} 40. [HttpDelete] 41. public string Remove(int id) 42. { 43. UserDal userDal = new(); 44. return userDal.RemoveUser(id) > 0 ? "删除成功" : "删除失败"; 45. } 46. } 47. }   UserDal.cs：   1. using System.Data; 2. using Microsoft.Data.SqlClient; 3. using WebapiDemo.Core; 4. using WebapiDemo.Models; 5. namespace WebapiDemo.Dal 6. { 7. public class UserDal 8. { 9. public bool GetUserByUserNoAndPassword(string userNo, string password) 10. { 11. DataTable res = SqlHelper.ExecuteTable( 12. "SELECT \* FROM Users WHERE UserNo = @UserNo AND Password = @Password", 13. new SqlParameter("@UserNo", userNo), 14. new SqlParameter("@Password", password) 15. ); 16. if (res.Rows.Count == 1) 17. { 18. return true; 19. } 20. else 21. { 22. return false; 23. } 24. } 25. public Users? GetUserById(int id) 26. { 27. DataRow row; 28. DataTable res = SqlHelper.ExecuteTable( 29. "SELECT \* FROM Users WHERE Id = @Id", 30. new SqlParameter("@Id", id) 31. ); 32. if (res.Rows.Count != 1) 33. { 34. return null; 35. } 36. row = res.Rows[0]; 37. Users user = 38. new() 39. { 40. Id = (int)row["Id"], 41. UserNo = row["UserNo"].ToString(), 42. UserName = row["UserName"].ToString(), 43. UserLevel = (int)row["UserLevel"], 44. Password = row["Password"].ToString() 45. }; 46. return user; 47. } 48. public int AddUser(string userNo, string userName, int userLevel, string password) 49. { 50. return SqlHelper.ExecuteNonQuery( 51. "INSERT INTO Users(UserNo, UserName, UserLevel, Password) VALUES(@UserNo, @UserName, @UserLevel, @Password)", 52. new SqlParameter("@UserNo", userNo), 53. new SqlParameter("@UserName", userName), 54. new SqlParameter("@UserLevel", userLevel), 55. new SqlParameter("@Password", password) 56. ); 57. } 58. public int UpdateUser( 59. int id, 60. string? userNo, 61. string? userName, 62. int? userLevel, 63. string? password 64. ) 65. { 66. DataTable res = SqlHelper.ExecuteTable( 67. "SELECT \* FROM Users WHERE Id = @Id", 68. new SqlParameter("@Id", id) 69. ); 70. if (res.Rows.Count == 0) 71. { 72. return 0; 73. } 74. DataRow row = res.Rows[0]; 75. Users user = 76. new() 77. { 78. Id = (int)row["Id"], 79. UserNo = userNo ?? row["UserNo"].ToString(), 80. UserName = userName ?? row["UserName"].ToString(), 81. UserLevel = userLevel ?? (int)row["UserLevel"], 82. Password = password ?? row["Password"].ToString() 83. }; 84. return SqlHelper.ExecuteNonQuery( 85. "UPDATE Users SET UserNo = @UserNo, UserName = @UserName, UserLevel = @UserLevel, Password = @Password WHERE Id = @Id", 86. new SqlParameter("@UserNo", user.UserNo), 87. new SqlParameter("@UserName", user.UserName), 88. new SqlParameter("@UserLevel", user.UserLevel), 89. new SqlParameter("@Password", user.Password), 90. new SqlParameter("@Id", user.Id) 91. ); 92. } 93. public int RemoveUser(int id) 94. { 95. return SqlHelper.ExecuteNonQuery( 96. "DELETE FROM Users WHERE Id = @Id", 97. new SqlParameter("@Id", id) 98. ); 99. } 100. } 101. }   **【实验结果及说明】**  使用ADO.net连接SqlServer，实现了对已有数据表基本的增删改查，通过DAO模式封装数据层操作，通过查询语句参数化避免了Sql注入。功能展示如下：    图 4 登录接口-成功登录    图 5 登录接口-登录失败    图 6 插入接口    图 7 插入的数据已存在于数据表中    图 8 更新接口    图 9 更新后id = 6的数据的UserName变为Update    图 10 删除接口    图 11 删除后id = 6的数据从表中删除 | |

|  |  |
| --- | --- |
| **【实验编号】** | **4** |
| **【实验名称】** | **SSH框架 学时：2** |
| **【实验目的】**  1）熟悉SSH，并掌握基本的配置；  2）使用SSH框架，尝试编写一个简单的小程序  **【实验内容】**  将工程导入eclipse开发环境，并部署到tomcat服务器；创建数据库和表；启动tomcat服务器，输入程序运行网址，查看运行效果；增加address列，按照ssh框架标准编写程序，然后再运行程序，查看运行结果  **【关键代码】**  前端Blazor部分：  Program.cs:   1. using Microsoft.AspNetCore.Components.Web; 2. using Microsoft.AspNetCore.Components.WebAssembly.Hosting; 3. using BlazorWADemo; 4. var builder = WebAssemblyHostBuilder.CreateDefault(args); 5. builder.RootComponents.Add<App>("#app"); 6. builder.RootComponents.Add<HeadOutlet>("head::after"); 7. builder.Services.AddAntDesign(); 8. builder.Services.AddScoped(sp => new HttpClient { BaseAddress = new Uri(builder.HostEnvironment.BaseAddress) }); 9. await builder.Build().RunAsync();   UserList.razor:   1. @page "/userlist" 2. <div style="display: flex; justify-content: space-between; align-items: center;"> 3. <h3>User List</h3> 4. <Button OnClick="NavigateToAddUserPage">新增用户</Button> 5. </div> 6. @inject NavigationManager NavigationManager 7. @inject HttpClient Http 8. @using System.ComponentModel 9. @using AntDesign 10. @using AntDesign.TableModels 11. <Table DataSource="@userList" Bordered=@true HidePagination="@true"> 12. <PropertyColumn Property="c=>c.Id" Title="ID" /> 13. <PropertyColumn Property="c=>c.Username" Title="USERNAME" /> 14. <PropertyColumn Property="c=>c.Password" Title="PASSWORD" /> 15. <PropertyColumn Property="c=>c.Address" Title="ADDRESS" /> 16. <ActionColumn Title="操作"> 17. <Button Type="@ButtonType.Primary" OnClick="() => NavigateToUpdateUserPage(context.Id)">修改</Button> 18. <Button Danger Type="@ButtonType.Primary" OnClick="() => DeleteUser(context.Id)">删除</Button> 19. </ActionColumn> 20. </Table> 21. @code { 22. User[]? userList; 23. private async Task LoadUserList() 24. { 25. userList = await Http.GetFromJsonAsync<User[]>("http://localhost:5286/LoginApi/web4user"); 26. } 27. protected override async Task OnInitializedAsync() 28. { 29. await LoadUserList(); 30. } 31. private void NavigateToAddUserPage() 32. { 33. NavigationManager.NavigateTo("/userlist/adduser"); 34. } 35. private void NavigateToUpdateUserPage(int id) 36. { 37. NavigationManager.NavigateTo($"/userlist/updateuser/{id}"); 38. } 39. private async void DeleteUser(int id) 40. { 41. await Http.DeleteAsync($"http://localhost:5286/LoginApi/web4user/{id}"); 42. await LoadUserList(); 43. StateHasChanged(); 44. } 45. public class User 46. { 47. [DisplayName("ID")] 48. public int Id { get; set; } 49. [DisplayName("USERNAME")] 50. public string? Username { get; set; } 51. [DisplayName("PASSWORD")] 52. public string? Password { get; set; } 53. [DisplayName("ADDRESS")] 54. public string? Address { get; set; } 55. } 56. }   UserListAddUser.razor:   1. @page "/userlist/adduser" 2. <h3>User List</h3> 3. @inject NavigationManager NavigationManager 4. @inject HttpClient Http 5. @using System.ComponentModel 6. @using AntDesign 7. @using AntDesign.TableModels 8. <div class="user-update"> 9. <div class="input"> 10. Username: 11. <Input Placeholder="Username" @bind-Value="@Username" /> 12. <br /> 13. <br /> 14. Password: 15. <Input Placeholder="Password" @bind-Value="@Password" /> 16. <br /> 17. <br /> 18. Address: 19. <Input Placeholder="Address" @bind-Value="@Address" /> 20. <br /> 21. <br /> 22. <Button Block type="primary" OnClick="@(e => AddUser(Username, Password, Address))">新增用户</Button> 23. <br /> 24. <br /> 25. <Button Block OnClick="NavigateToUserList">返回列表</Button> 26. </div> 27. </div> 28. <style> 29. .user-update { 30. border: 1px solid #ccc; 31. width: 300px; 32. margin: 0 auto; 33. margin-top: 300px; 34. } 35. .user-update .input { 36. padding: 20px; 37. } 38. </style> 39. @code { 40. public string? Username { get; set; } 41. public string? Password { get; set; } 42. public string? Address { get; set; } 43. public class User 44. { 45. public int Id { get; set; } 46. public string? Username { get; set; } 47. public string? Password { get; set; } 48. public string? Address { get; set; } 49. } 50. private async void AddUser(string? username, string? password, string? address) 51. { 52. await Http.PostAsJsonAsync("http://localhost:5286/LoginApi/web4user", new User 53. { 54. Username = username, 55. Password = password, 56. Address = address 57. }); 58. } 59. private void NavigateToUserList() 60. { 61. NavigationManager.NavigateTo("/userlist"); 62. } 63. }   UserListUpdate.razor:   1. @page "/userlist/updateuser/{Id:int}" 2. <h3>User List</h3> 3. @inject NavigationManager NavigationManager 4. @inject HttpClient Http 5. @using System.ComponentModel 6. @using AntDesign 7. @using AntDesign.TableModels 8. <div class="user-update"> 9. <div class="input"> 10. ID: 11. <Input Placeholder="Id" @bind-Value="@Id" Disabled /> 12. <br /> 13. <br /> 14. Username: 15. <Input Placeholder="Username" @bind-Value="@Username" /> 16. <br /> 17. <br /> 18. Password: 19. <InputPassword Placeholder="Password" @bind-Value="@Password" /> 20. <br /> 21. <br /> 22. Address: 23. <Input Placeholder="Address" @bind-Value="@Address" /> 24. <br /> 25. <br /> 26. <Button Block type="primary" OnClick="@(e => UpdateUser(Id, Username, Password, Address))">修改用户</Button> 27. <br /> 28. <br /> 29. <Button Block OnClick="NavigateToUserList">返回列表</Button> 30. </div> 31. </div> 32. <style> 33. .user-update { 34. border: 1px solid #ccc; 35. width: 400px; 36. margin: 0 auto; 37. margin-top: 200px; 38. } 39. .user-update .input { 40. padding: 20px; 41. } 42. </style> 43. @code { 44. [Parameter] 45. public int Id { get; set; } 46. public string? Username { get; set; } 47. public string? Password { get; set; } 48. public string? Address { get; set; } 49. public class User 50. { 51. public int Id { get; set; } 52. public string? Username { get; set; } 53. public string? Password { get; set; } 54. public string? Address { get; set; } 55. } 56. private async void UpdateUser(int id, string? username, string? password, string? address) 57. { 58. await Http.PutAsJsonAsync($"http://localhost:5286/LoginApi/web4user/{id}", new User 59. { 60. Username = username, 61. Password = password, 62. Address = address 63. }); 64. } 65. private void NavigateToUserList() 66. { 67. NavigationManager.NavigateTo("/userlist"); 68. } 69. }   后端WebApi部分：  LoginApiController.cs：   1. using Microsoft.AspNetCore.Cors; 2. using Microsoft.AspNetCore.Mvc; 3. using WebapiDemo.Models; 4. using WebApiDemo.BLL; 5. using WebApiDemo.Entities; 6. using WebApiDemoCommon; 7. namespace WebapiDemo.Controllers 8. { 9. [ApiController] 10. [Route("[controller]")] 11. [EnableCors("any")] 12. public class LoginApiController : ControllerBase 13. { 14. private readonly IUserBll userBll; 15. public LoginApiController(IUserBll userBll) 16. { 17. this.userBll = userBll; 18. } 19. [HttpGet("web4user")] 20. public List<Web4User> GetAllWeb4Users() 21. { 22. return userBll.GetAllWeb4Users(); 23. } 24. [HttpPut("web4user/{id}")] 25. public string UpdateWeb4User(int id, [FromBody] Web4UserUpdateRequest request) 26. { 27. if (request == null) 28. { 29. return "请求参数为空"; 30. } 31. Web4User web4User = 32. new() 33. { 34. Id = id, 35. UserName = request.UserName, 36. Password = request.Password, 37. Address = request.Address 38. }; 39. return userBll.UpdateWeb4User(web4User); 40. } 41. [HttpPost("web4user")] 42. public string InsertWeb4User([FromBody] Web4UserInsertRequest request) 43. { 44. if (request == null) 45. { 46. return "请求参数为空"; 47. } 48. Web4User web4User = 49. new() 50. { 51. UserName = request.UserName, 52. Password = request.Password, 53. Address = request.Address 54. }; 55. return userBll.InsertWeb4User(web4User); 56. } 57. [HttpDelete("web4user/{id}")] 58. public string RemoveWeb4User(int id) 59. { 60. return userBll.RemoveWeb4User(id); 61. } 62. } 63. }   UserBll.cs：   1. using WebApiDemo.DAL; 2. using WebApiDemo.Entities; 3. namespace WebApiDemo.BLL 4. { 5. public class UserBll : IUserBll 6. { 7. readonly UserDal userDal = new(); 8. public List<User> GetAllUsers() 9. { 10. return UserDal.GetAllUsers(); 11. } 12. public List<Web4User> GetAllWeb4Users() 13. { 14. return UserDal.GetAllWeb4Users(); 15. } 16. public string UpdateWeb4User(Web4User web4User) 17. { 18. return UserDal.UpdateWeb4User(web4User) > 0 ? "更新成功" : "更新失败"; 19. } 20. public string InsertWeb4User(Web4User web4User) 21. { 22. return UserDal.InsertWeb4User(web4User) > 0 ? "插入成功" : "插入失败"; 23. } 24. public string RemoveWeb4User(int id) 25. { 26. return UserDal.RemoveWeb4User(id) > 0 ? "删除成功" : "删除失败"; 27. } 28. } 29. }   UserDal.cs：   1. using System.Data; 2. using Microsoft.Data.SqlClient; 3. using WebApiDemo.Entities; 4. using WebApiDemo.Entities.Factorys; 5. namespace WebApiDemo.DAL; 6. public class UserDal 7. { 8. public static List<Web4User> GetAllWeb4Users() 9. { 10. using var context = DbContextFactory.GetDbContext(); 11. var users = context.Web4Users.ToList(); 12. return users; 13. } 14. public static int UpdateWeb4User(Web4User web4user) 15. { 16. using var context = DbContextFactory.GetDbContext(); 17. var userToUpdate = context.Web4Users.Find(web4user.Id); 18. if (userToUpdate == null) 19. { 20. return 0; 21. } 22. var properties = typeof(Web4User).GetProperties().Where(p => p.Name != "Id"); 23. foreach (var property in properties) 24. { 25. var value = property.GetValue(web4user); 26. if (value is string str && !string.IsNullOrWhiteSpace(str)) 27. { 28. property.SetValue(userToUpdate, value); 29. } 30. else if (value is not string && value != null) 31. { 32. property.SetValue(userToUpdate, value); 33. } 34. } 35. context.Web4Users.Update(userToUpdate); 36. return context.SaveChanges(); 37. } 38. public static int InsertWeb4User(Web4User web4user) 39. { 40. using var context = DbContextFactory.GetDbContext(); 41. context.Web4Users.Add(web4user); 42. return context.SaveChanges(); 43. } 44. public static int RemoveWeb4User(int id) 45. { 46. using var context = DbContextFactory.GetDbContext(); 47. var userToRemove = context.Web4Users.Find(id); 48. if (userToRemove == null) 49. { 50. return 0; 51. } 52. context.Web4Users.Remove(userToRemove); 53. return context.SaveChanges(); 54. } 55. }   Web4User.cs：   1. namespace WebApiDemo.Entities 2. { 3. public class Web4User 4. { 5. public int Id { get; set; } 6. public string? UserName { get; set; } 7. public string? Password { get; set; } 8. public string? Address { get; set; } 9. } 10. }   **【实验结果及说明】**  本次实验基于.net环境代替SSH框架，使用Blazor代替jsp，ASP.NET Core MVC 代替Struts，ASP.NET Core代替Spring，Entity Framework Core代替Hibernate，SqlServer代替MYSQL等价完成。项目结构如图所示：    图 12 项目结构  其中，WebApiDemo提供接口，WebApiDemo.BLL处理业务逻辑，WebApiDemo.DAL封装数据访问，WebApiDemo.Entities引入EFCore用于数据库交互，WebApiDemoCommon提供工具类库，BlazorWADemo提供前端页面。修改前前端页面如图所示：    图 13 修改前页面  修改BlazorWADemo中的UserList.razor，在前端添加ADDRESS列；修改WebApiDemo.Entities中的Web4User.cs添加Address属性，在数据库上下文中指定约束，迁移数据库并更新：    图 14 添加属性    图 15 指定Address最大长度200    图 16 迁移并更新数据库    图 17 更新后的数据库  修改后前端页面如图所示：    图 18 修改后的前端页面  点击修改按钮进入修改页：    图 19 修改页  将密码修改为ACE,地址修改为Here,返回列表后如图所示：    图 20 数据修改成功  点击新增用户按钮进入新增页：    图 21 新增页  点击新增用户后返回，用户已增加：    图 22 新增用户  点击删除后删除一条数据并自动刷新：    图 23 删除用户 | |