

v1.0.0 ▾

...

[OrchardCore](#) / [src](#) / [OrchardCore.Modules](#) / [OrchardCore.Users](#) / [Controllers](#) / [ResetPasswordController.cs](#) / [Code](#) Jump to ▾

domonkosgabor Audit Trail module (Lombiq Technologies: OCORE-23) (#6679) ✓

[History](#)

9 contributors



162 lines (141 sloc) | 6.02 KB

...

```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Authorization;
6 using Microsoft.AspNetCore.Identity;
7 using Microsoft.AspNetCore.Mvc;
8 using Microsoft.Extensions.Localization;
9 using Microsoft.Extensions.Logging;
10 using OrchardCore.Entities;
11 using OrchardCore.Modules;
12 using OrchardCore.Settings;
13 using OrchardCore.Users.Events;
14 using OrchardCore.Users.Models;
15 using OrchardCore.Users.Services;
16 using OrchardCore.Users.ViewModels;
17
18 namespace OrchardCore.Users.Controllers
19 {
20     [Feature("OrchardCore.Users.ResetPassword")]
21     public class ResetPasswordController : Controller
22     {
23         private readonly IUserService _userService;
24         private readonly UserManager<User> _userManager;
25         private readonly ISiteService _siteService;
26         private readonly IEnumerable<IPasswordRecoveryFormEvents> _passwordRecoveryFormEvents;
27         private readonly ILogger _logger;
28         private readonly IStringLocalizer S;
29
30         public ResetPasswordController(
31             IUserService userService,
32             UserManager<User> userManager,
33             ISiteService siteService,
34             IStringLocalizer<ResetPasswordController> stringLocalizer,
35             ILogger<ResetPasswordController> logger,
36             IEnumerable<IPasswordRecoveryFormEvents> passwordRecoveryFormEvents)
37         {
38             _userService = userService;
39             _userManager = userManager;
40             _siteService = siteService;
41
42             S = stringLocalizer;
43             _logger = logger;
44             _passwordRecoveryFormEvents = passwordRecoveryFormEvents;
45         }
46
47         [HttpGet]
48         [AllowAnonymous]
49         public async Task<ActionResult> ForgotPassword()
50         {
51             if (!await _siteService.GetSiteSettingsAsync().As<ResetPasswordSettings>().AllowResetPassword)
52             {
53                 return NotFound();
54             }
55
56             return View();
57         }
58
59         [HttpPost]
60         [AllowAnonymous]
61         public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)
62         {
63             if (!await _siteService.GetSiteSettingsAsync().As<ResetPasswordSettings>().AllowResetPassword)
64             {
65                 return NotFound();
66             }
67
68             await _passwordRecoveryFormEvents.InvokeAsync((e, modelState) => e.RecoveringPasswordAsync((key, message) => modelState.AddModelError(key, message)), ModelState, _logg
69
70             if (TryValidateModel(model) && ModelState.IsValid)
71             {
72                 var user = await _userService.GetForgotPasswordUserAsync(model.Email) as User;
73                 if (user == null || (
74                     (await _siteService.GetSiteSettingsAsync().As<RegistrationSettings>().UsersMustValidateEmail
75                     && !await _userManager.IsEmailConfirmedAsync(user))
76                 )
77                 {
78                     // returns to confirmation page anyway: we don't want to let scrapers know if a username or an email exist
```

```

79         return RedirectToLocal(Url.Action("ForgotPasswordConfirmation", "ResetPassword"));
80     }
81
82     user.ResetToken = Convert.ToBase64String(Encoding.UTF8.GetBytes(user.ResetToken));
83     var resetPasswordUrl = Url.Action("ResetPassword", "ResetPassword", new { code = user.ResetToken }, HttpContext.Request.Scheme);
84     // send email with callback link
85     await this.SendEmailAsync(user.Email, S["Reset your password"], new LostPasswordViewModel() { User = user, LostPasswordUrl = resetPasswordUrl });
86
87     var context = new PasswordRecoveryContext(user);
88
89     await _passwordRecoveryFormEvents.InvokeAsync((handler, context) => handler.PasswordRecoveredAsync(context), context, _logger);
90
91     return RedirectToLocal(Url.Action("ForgotPasswordConfirmation", "ResetPassword"));
92 }
93
94 // If we got this far, something failed, redisplay form
95 return View(model);
96 }
97
98 [HttpGet]
99 [AllowAnonymous]
100 public IActionResult ForgotPasswordConfirmation()
101 {
102     return View();
103 }
104
105 [HttpGet]
106 [AllowAnonymous]
107 public async Task<IActionResult> ResetPassword(string code = null)
108 {
109     if (!await _siteService.GetSiteSettingsAsync().As<ResetPasswordSettings>().AllowResetPassword)
110     {
111         return NotFound();
112     }
113     if (code == null)
114     {
115         //"A code must be supplied for password reset.";
116     }
117     return View(new ResetPasswordViewModel { ResetToken = code });
118 }
119
120 [HttpPost]
121 [AllowAnonymous]
122 [ValidateAntiForgeryToken]
123 public async Task<IActionResult> ResetPassword(ResetPasswordViewModel model)
124 {
125     if (!await _siteService.GetSiteSettingsAsync().As<ResetPasswordSettings>().AllowResetPassword)
126     {
127         return NotFound();
128     }
129
130     await _passwordRecoveryFormEvents.InvokeAsync((e, ModelState) => e.ResettingPasswordAsync((key, message) => ModelState.AddModelError(key, message)), ModelState, _logger);
131
132     if (TryValidateModel(model) && ModelState.IsValid)
133     {
134         if (await _userService.ResetPasswordAsync(model.Email, Encoding.UTF8.GetString(Convert.FromBase64String(model.ResetToken)), model.NewPassword, (key, message) => ModelState.AddModelError(key, message)))
135         {
136             return RedirectToLocal(Url.Action("ResetPasswordConfirmation", "ResetPassword"));
137         }
138     }
139
140     return View(model);
141 }
142
143 [HttpGet]
144 [AllowAnonymous]
145 public IActionResult ResetPasswordConfirmation()
146 {
147     return View();
148 }
149
150 private IActionResult RedirectToLocal(string returnUrl)
151 {
152     if (Url.IsLocalUrl(returnUrl))
153     {
154         return Redirect(returnUrl);
155     }
156     else
157     {
158         return Redirect("~/");
159     }
160 }
161 }
162 }

```