Mariusz Wieczorek

mariusz.wieczorek@kabat.pl

Streszczenie

Uzupełnienie danych rejestracyjnych użytkownika dostarczonych przez szablon, dodatkowymi polami, które będą nam potrzebne w aplikacji.  
W tym celu dodajemy nowe pola w widoku, w modelu widoku oraz w kontrolerze.

ASP.NET MVC

Dodatkowe pola w formularzu rejestracyjnym  
dostarczonym przez szablon.

**Modyfikacja Modelu: ApplicationUser**

|  |
| --- |
| namespace InvoiceManager.Models  {  // You can add profile data for the user by adding more properties to your ApplicationUser class, please visit https://go.microsoft.com/fwlink/?LinkID=317594 to learn more.  public class ApplicationUser : IdentityUser  {  public ApplicationUser()  {  Invoices = new Collection<Invoice>();  Clients = new Collection<Client>();  }    [Required]  public string Name { get; set; }  public int AddressId { get; set; }  public Address Address { get; set; }    public ICollection<Invoice> Invoices { get; set; }  public ICollection<Client> Clients { get; set; }    public async Task<ClaimsIdentity>  GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)    {  // Note the authenticationType must match the one defined in  // CookieAuthenticationOptions.AuthenticationType  var userIdentity = await manager.CreateIdentityAsync(this,   DefaultAuthenticationTypes.ApplicationCookie);  // Add custom user claims here  return userIdentity;  }  }  } |

**Modyfikacja Modelu Widoku: RegisterViewModel**  
Dodajemy/ edytujemy dane użytkownika w formularzu register.  
Musimy więc dodać te dane do formularza na widoku.  
Widok korzysta z modelu widoku **RegisterViewModel**

|  |
| --- |
| @model InvoiceManager.Models.**RegisterViewModel** |

Musimy również go zmodyfikować

|  |
| --- |
| namespace InvoiceManager.Models  {  public class RegisterViewModel  {  [Required]  [EmailAddress]  [Display(Name = "Email")]  public string Email { get; set; }  [Required]  [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.",  MinimumLength = 6)]  [DataType(DataType.Password)]  [Display(Name = "Password")]  public string Password { get; set; }  [DataType(DataType.Password)]  [Display(Name = "Confirm password")]  [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]  public string ConfirmPassword { get; set; } |
| public string Name { get; set; }  public Address Address { get; set; } |
| }  } |

Uwaga właściwość **Display** odpowiada za treść wyświetlaną w labelu

|  |
| --- |
| public class RegisterViewModel  {  [Required]  [EmailAddress]  [Display(Name = "Email")]  public string Email { get; set; } |

**Zmiany w widoku**

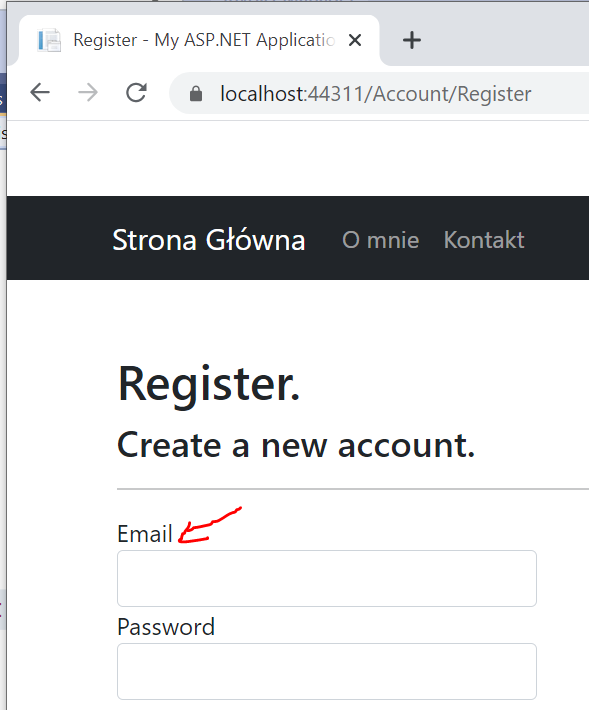
W widoku dodajemy

|  |
| --- |
| <div class="form-group">  @Html.LabelFor(m => m.Name, new { @class = "col-md-2 control-label" })  <div class="col-md-10">  @Html.TextBoxFor(m => m.Name, new { @class = "form-control" })  </div>  </div>  <div class="form-group">  @Html.LabelFor(m => m.Address.City, new { @class = "col-md-2 control-label" })  <div class="col-md-10">  @Html.TextBoxFor(m => m.Address.City, new { @class = "form-control" })  </div>  </div>  <div class="form-group">  @Html.LabelFor(m => m.Address.PostalCode, new { @class = "col-md-2 control-label" })  <div class="col-md-10">  @Html.TextBoxFor(m => m.Address.PostalCode, new { @class = "form-control" })  </div>  </div>  <div class="form-group">  @Html.LabelFor(m => m.Address.Street, new { @class = "col-md-2 control-label" })  <div class="col-md-10">  @Html.TextBoxFor(m => m.Address.Street, new { @class = "form-control" })  </div>  </div>  <div class="form-group">  @Html.LabelFor(m => m.Address.Number, new { @class = "col-md-2 control-label" })  <div class="col-md-10">  @Html.TextBoxFor(m => m.Address.Number, new { @class = "form-control" })  </div>  </div> |

**Zmiany w kontrolerze**Aby móc zapisać informacje z nowo dodanych pól, zmian musimy dokonać również w kontrolerze, Musimy uzupełnić dodatkowe pole w obiekcie przekazywanym do metody tworzącej konto użytkownika.

|  |
| --- |
| // POST: /Account/Register  [HttpPost]  [AllowAnonymous]  [ValidateAntiForgeryToken]  public async Task<ActionResult> Register(RegisterViewModel model)  {  if (ModelState.IsValid)  {  var user = new ApplicationUser  { UserName = model.Email,  Email = model.Email,  Name = model.Name,  Address = model.Address,  };  var result = await UserManager.CreateAsync(user, model.Password);  if (result.Succeeded)  {  await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);    // For more information on how to enable account confirmation and password reset please visit https://go.microsoft.com/fwlink/?LinkID=320771  // Send an email with this link  // string code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);  // var callbackUrl = Url.Action("ConfirmEmail", "Account", new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);  // await UserManager.SendEmailAsync(user.Id, "Confirm your account", "Please confirm your account by clicking <a href=\"" + callbackUrl + "\">here</a>");  return RedirectToAction("Index", "Home");  }  AddErrors(result);  } |

**Zmiana opisów pól.**



Możemy to zrobić bezpośrednio w **modelu widoku**.  
Ustawiamy atrybut Display i wartość atrybutu Name

|  |
| --- |
| [DataType(DataType.Password)]  [Display(Name = "Potwierdź Hasło")]  [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]  public string ConfirmPassword { get; set; }  [Display(Name = "Nazwa Użytkownika")]  public string Name { get; set; }  public Address Address { get; set; } |

Natomiast opis do pól adresowych uzupełniamy w **klasie Address**.

|  |
| --- |
| public class Address  {  public Address()  {  Clients = new Collection<Client>();  ApplicationUsers = new Collection<ApplicationUser>();  }  public int Id { get; set; }  [Required]  [Display(Name = "Ulica")]  public string Street { get; set; }  [Required]  [Display(Name = "Numer Domu")]  public string Number { get; set; }  [Required]  [Display(Name = "Miasto")]  public string City { get; set; }  [Required]  [Display(Name = "Kod Pocztowy")]  public string PostalCode { get; set; } |