Mariusz Wieczorek

mariusz.wieczorek@kabat.pl

Streszczenie

Czym jest uwierzytelnianie, a czym autoryzacja  
HttpGet, HttpPost

Atrybuty: [Authorize], [AllowAnonymous]

ASP.NET MVC

uwierzytelnianie oraz autentykacja

**Uwierzytelnianie i Autoryzacja**

|  |
| --- |
| **Uwierzytelnianie** (ang. authentication) – proces potwierdzenia twojej tożsamości. Podanie loginu, maila, hasła, po prostu logowanie do aplikacji. |

|  |
| --- |
| **Autoryzacja** (ang. authorization )– proces, w którym potwierdza się, czy dany podmiot ma dostęp do danego zasobu, funkcji, widoku |

Dzięki szablonowi ASP.NET MVC proces uwierzytelniania mamy już gotowy

W AcountController – jest tu mnóstwo logiki związanej z logowaniem i rejestracją użytkownika.  
Potwierdzenie maila, przypomnienie hasła

W ManageController – zarządzanie kontem, zmiana hasła, podwójna autoryzacja itd.

**HttpPost**

Protokół http posiada kilka typów metod takich jak Get, Post, Delete, Put, Options  
Domyślnie wszystkie akcje są typu Get  
Jeżeli akcja ma być typu Post musimy nadać jej atrybut [HttpPost]

**Get** - gdy pozyskujemy jakieś dane to przekazujemy parametry w adresie Url po znaku zapytania oddzielone znakami &.  
**Post** – Gdy chcemy wysłać dane na serwer w bardziej bezpieczny sposób. Dane przesyłamy w ciele żądania http, nie są one przesyłane jawnie i widoczne w adresie Url.

W kontrolerze używamy Atrybutu: [Authorize] aby zaznaczyć akcje, czyli metody, które abyś mógł wykonać musisz być zalogowany. Można tym atrybutem również zaznaczyć cały kontroler.  
Może on przyjmować parametry np. [Authorize(Roles = „Admin”)] lub  
[Authorize(Users= „Admin, Jan, Adam”)]

Takim atrybutem oznaczymy kontroler Home

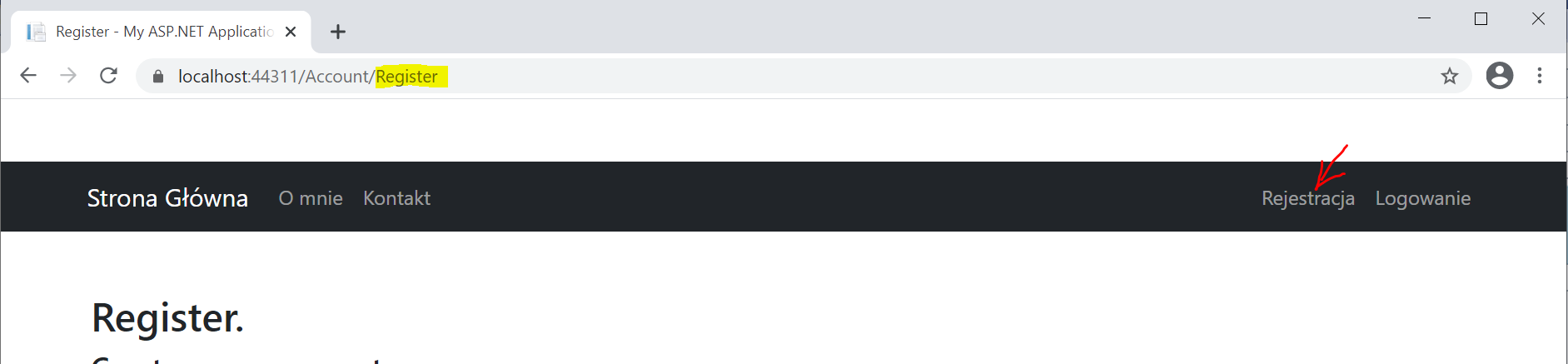
|  |
| --- |
| [Authorize]  public class HomeController : Controller  {  public ActionResult Index()  {  return View();  } |

Atrybut [AllowAnonymous] pozwala aby akcja w obrębie klasy oznaczonej jako Authorize mogła być wywołana przez osobę niezalogowaną.

W naszym kontrolerze Home oznaczymy tak niektóre akcje

|  |
| --- |
| [AllowAnonymous]  public ActionResult About()  {  ViewBag.Message = "Your application description page.";  var tekst = "121212 1212121 1212121";  return View((object)tekst);  }    [AllowAnonymous]  public ActionResult Contact()  {  ViewBag.Message = "Your contact page.";  return View();  } |

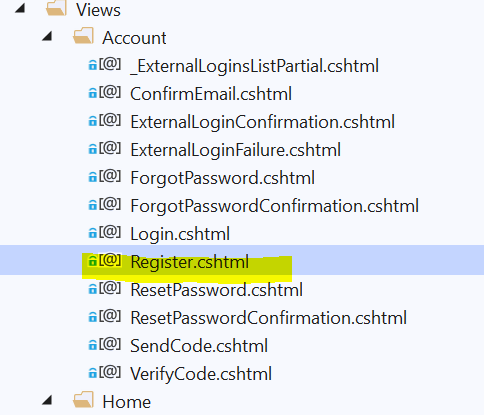
Analiza Formularza Rejestracji



Po kliknięci na przycisk Register zostaje wywołana Akcja z kontrolera Account o nazwie Register.  
Szukamy więc metody Register w metodzie AccountController.cs oznaczonej jako Get.

|  |
| --- |
| // GET: /Account/Register  [AllowAnonymous]  public ActionResult Register()  {  return View();  }  //  // POST: /Account/Register  [HttpPost]  [AllowAnonymous]  [ValidateAntiForgeryToken]  public async Task<ActionResult> Register(RegisterViewModel model)  {  if (ModelState.IsValid)  { |

Zostaje tam zwrócony widok, bez żadnego parametru więc szukamy widoku o nazwie Register.cshtml  
w folderze Account.



Jest to kod napisany w Razor.

|  |
| --- |
| @using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-horizontal", role = "form" }))  {  @Html.AntiForgeryToken() |
| <div class="form-group">  <div class="col-md-offset-2 col-md-10">  <input type="submit" class="btn btn-default" value="Register" />  </div>  </div>  } |

Po naciśnięciu przycisku oznaczonego jako submit z value = Register  
Wywoła w kontrolerze Account metodę Register oznaczoną jako HttpPost   
Ta akcja otrzymuje dane, które są walidowane a następnie zapisane do bazy

|  |
| --- |
| // 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 };  var result = await UserManager.CreateAsync(user, model.Password);  if (result.Succeeded)  {  await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false); |