Mariusz Wieczorek

[Adres e-mail]

Streszczenie

Czym jest sesja, pliki cookies, cache jak i kiedy wykorzystywać te elementy.  
Pola statyczne w kontrolerze.  
Jak tworzyć i odczytywać pliki cookies zarówno po stronie kontrolera w c#   
 jak i po stronie kodu html za pomocą jQuery.  
Implementacja potwierdzenia informacji o wykorzystywaniu przez stronę plików cookie

ASP.NET MVC

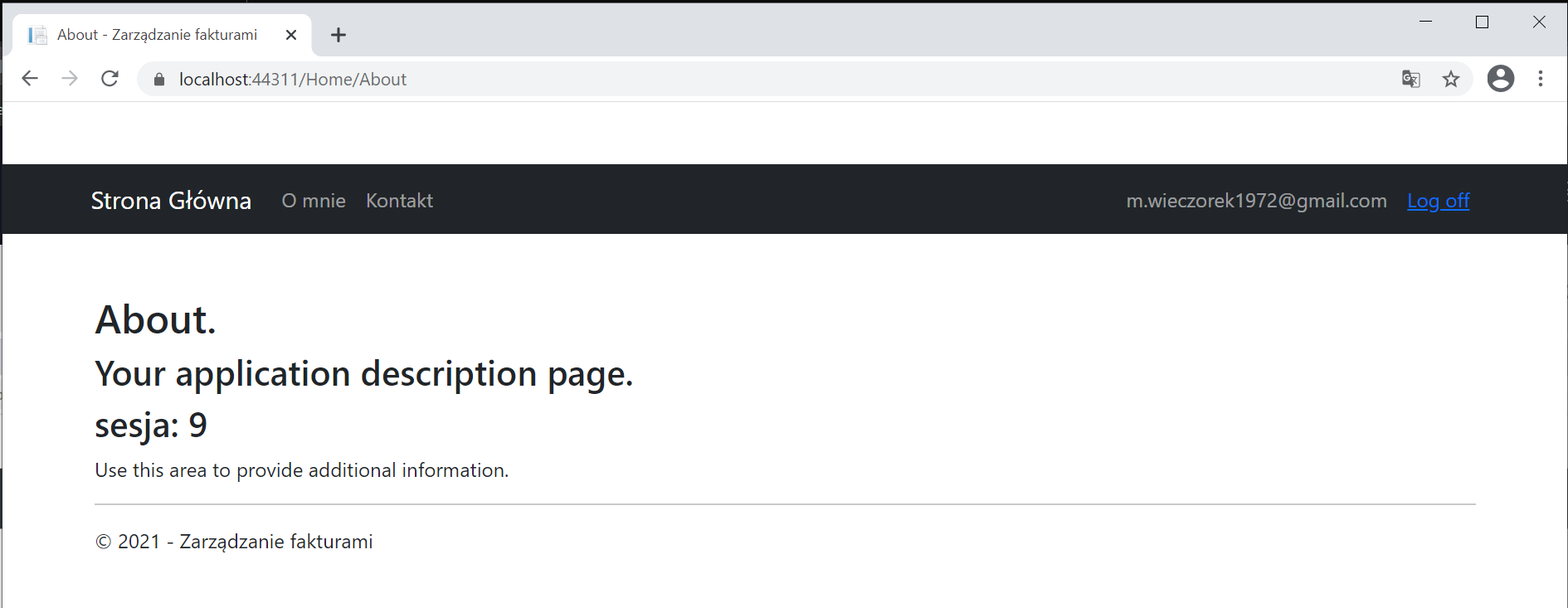
Sesje, pliki cookies, cache

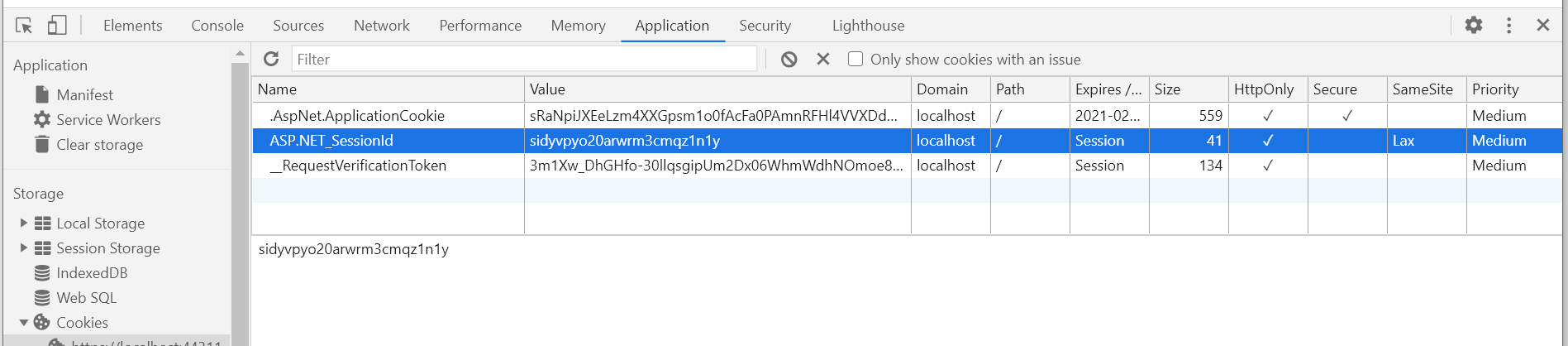
**Sesja**

Sesja jest **przechowywana po stronie serwera** i może zawierać informację  
per użytkownik a także per przeglądarka.  
Np. koszyk z zakupami w sklepie internetowym, który jest przechowywany na serwerze dla danego użytkownika przez pewien czas. Czas sesji możemy sami zdefiniować.

|  |
| --- |
| [AllowAnonymous]  public ActionResult About()  {  ViewBag.Message = "Your application description page.";  var valSession = GetSession();  valSession++;  UpdateSession(valSession);  ViewBag.ValSession = valSession;  return View();  }  private void UpdateSession(int i)  {  Session["klucz\_nr"] = i;  }  private int GetSession()  {  if (Session["klucz\_nr"] != null)  return (int)Session["klucz\_nr"];  return 0;  } |

Tworzymy w kontrolerze dwie prywatne metody UpdateSession i GetSession.  
Używamy ją w akcji About()  
Jako klucz możemy podać cokolwiek co potrzebujemy np. numer zamówienia.



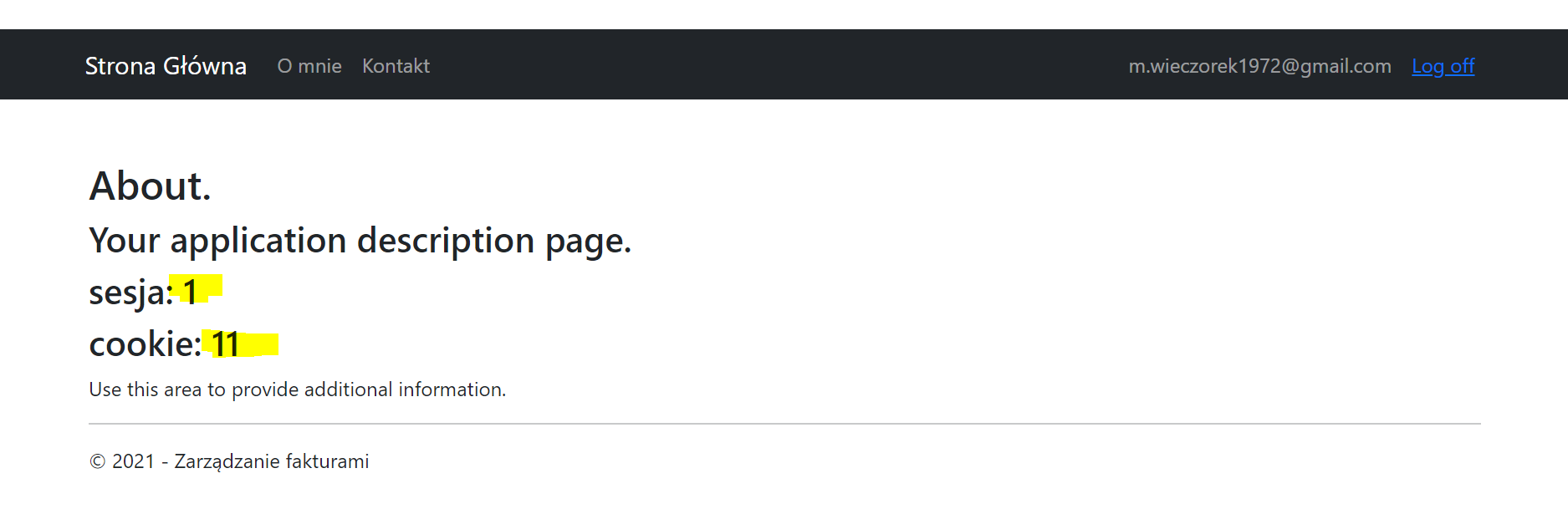


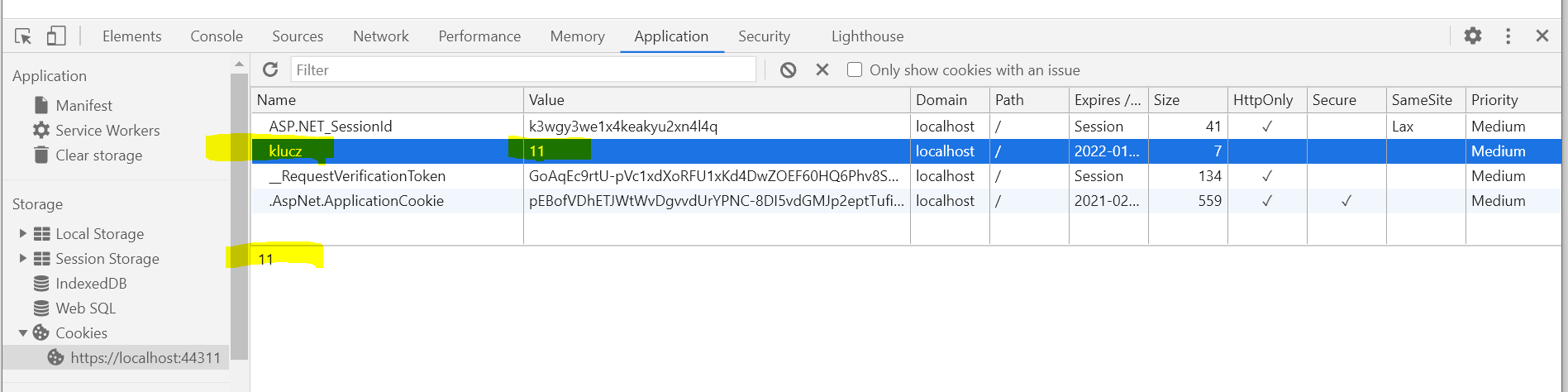
Id sesji jest trzymane w plikach cookies. Jak usuniemy plik cookie to licznik będzie liczył od nowa.

**Pliki Cookies**

Pliki Cookies **trzymane są po stronie przeglądarki** w odróżnieniu od sesji.  
Mogą być zapisywane na określony czas i mogą mieć maksymalnie do 4 kB.  
Można w nich zapamiętywać jakieś wcześniejsze wybory użytkownika.

|  |
| --- |
| [AllowAnonymous]  public ActionResult About()  {  ViewBag.Message = "Your application description page.";    var valSession = GetSession();  valSession++;  UpdateSession(valSession);  ViewBag.ValSession = valSession;  var valCookie = GetCookie();  valCookie++;  UpdateCookie(valCookie);  ViewBag.ValCookie = valCookie;  return View();  }  private void UpdateCookie(int i)  {  var cookie = new HttpCookie("klucz",i.ToString());  cookie.Expires = DateTime.Now.AddDays(365);  Response.SetCookie(cookie);  }  private int GetCookie()  {  if (Request.Cookies["klucz"] != null)  return int.Parse(Request.Cookies["klucz"].Value);  return 0;  } |



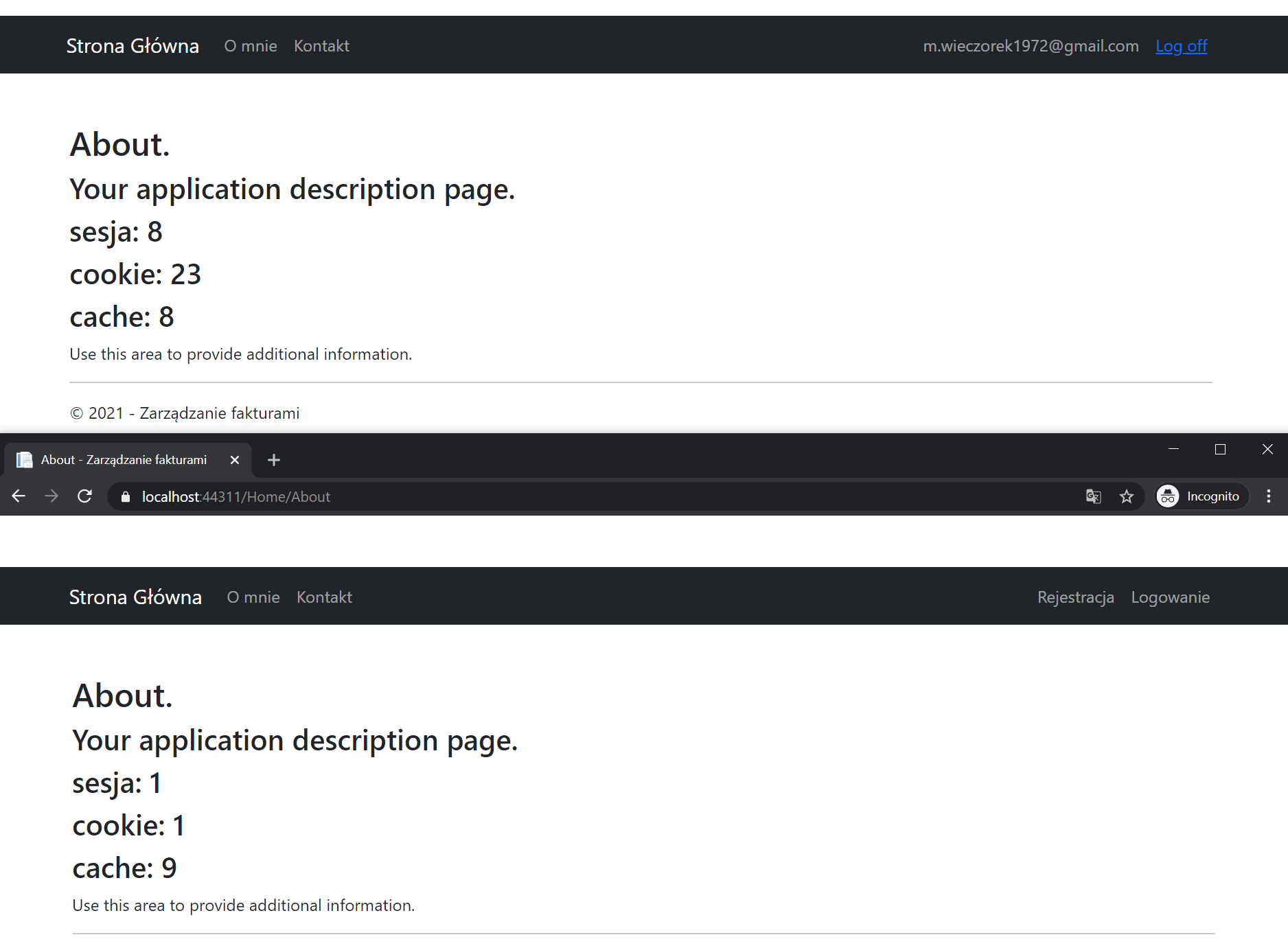
Przy kolejnym starcie aplikacji wartość cookie jest taka jaka była przy poprzednim uruchomieniu.  


Cookie jest widoczne pod nazwą klucza

**Cache**

Jest przechowywany po stronie serwera w celu wymiany danych między użytkownikami.  
Jest używany w celu odciążenia serwera

|  |
| --- |
| [AllowAnonymous]  public ActionResult About()  {  ViewBag.Message = "Your application description page.";    var valCache = GetCache();  valCache++;  UpdateCache(valCache);  ViewBag.ValCache = valCache;  return View();  }    private void UpdateCache(int i)  {  HttpRuntime.Cache["klucz"] = i;  }  private int GetCache()  {  if (HttpRuntime.Cache["klucz"] != null)  return (int)HttpRuntime.Cache["klucz"];  return 0;  } |



Cache jest trzymany na serwerze, tak więc gdy otworzymy nowe okno incognito licznik będzie kontynuował. Używamy aby uniknąć odczytywania z bazy danych rzadko aktualizowanych danych.

Możemy też całą akcję w kontrolerze oznaczyć jako **[OutputCache]** i ustawić czas przez jaki te dane będą przechowywane w kontrolerze.

|  |
| --- |
| public static int number = 0;  [AllowAnonymous]  [OutputCache(Duration = 10)]  public ActionResult Contact()  {  number++;  ViewBag.Message = "Your contact page." + number;  return View();  } |

Podczas odświeżania zadziała tylko odświeżenie, które będzie wykonane po upływie 10 sekundowej  
przerwie od ostatniego odświeżenia.  
Gdy usuniemy atrybut **[OutputCache]**, to każde odświeżenie spowoduje inkrementacje.

Uwaga:  
Wykorzystujemy pole statyczne, które jest tworzone tylko raz i jest wspólne dla wszystkich użytkowników. Dlatego należy zwrócić uwagę, aby nie przechowywać w niej jakiś danych wrażliwych, które mógłby zobaczyć inny użytkownik.

Gdyby pole nie było statyczne, za każdym odświeżeniem byłoby tworzone na nowo, czyli zawsze była by wyświetlana jedynka.

**Zastosowanie w naszej aplikacji**  
W aplikacji zastosujemy do powiadamiania o tym, że aplikacja wykorzystuje pliki cookies i poproszenie użytkownika, że zapoznał się z tą informacją.

Komunikat ten ma być wyświetlany na każdej stronie. Jak użytkownik wciśnie przycisk ok  
za pomocą jQuery usuniemy ten komunikat, oraz dodamy nowy plik cookie, który będzie informował o tym, że dany użytkownik zapoznał się z komunikatem.  
Dzięki temu przy kolejnym uruchomieniu nie będzie tego komunikatu już otrzymywał.

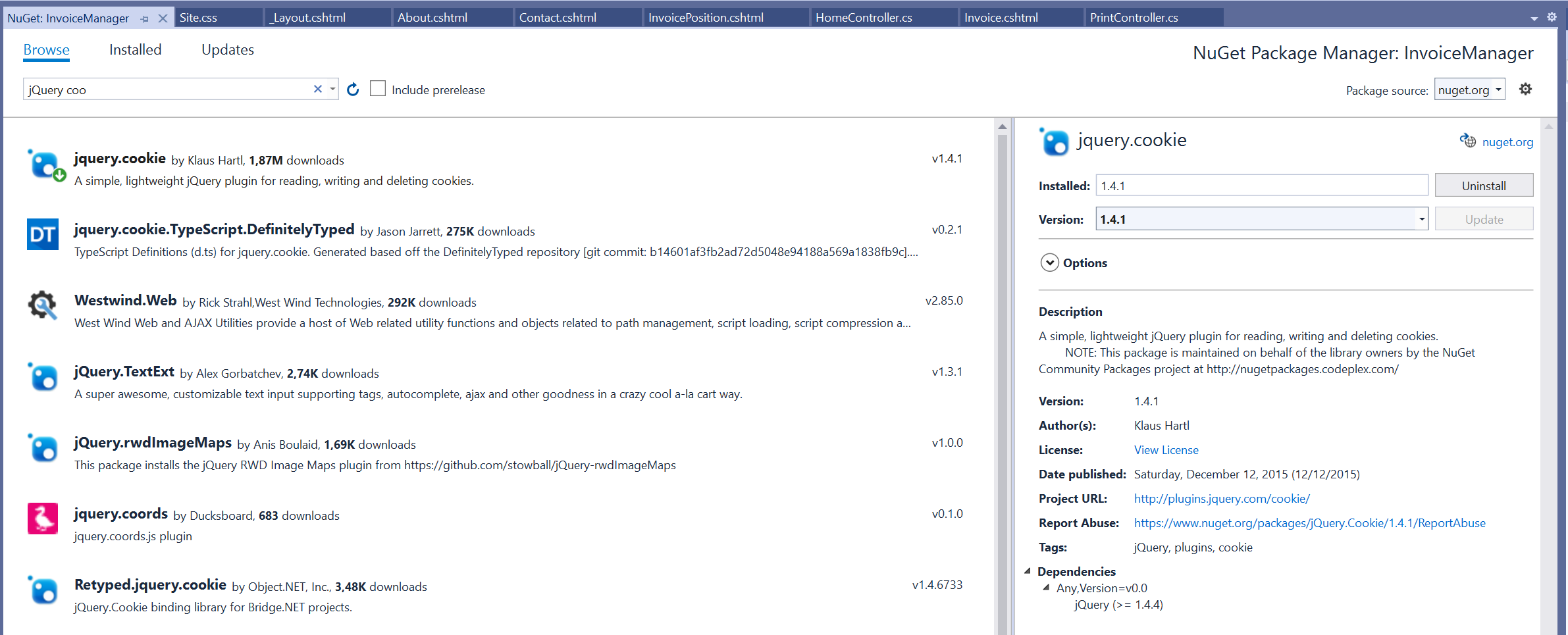
Dodajemy więc kod do pliku \_layout.cshtml.

|  |
| --- |
| @Scripts.Render("~/bundles/jquery")  @Scripts.Render("~/bundles/jqueryval")  @Scripts.Render("~/bundles/bootstrap")  <div class="privacyPoliceInfo">  Serwis wykorzystuje pliki cookies.Korzystając ze strony wyrażasz zgodę na wykorzystanie  plików cookies.  <button onclick="closePrivacyPolice()" type="button"   class="btn btn-light btn-sm mx-3 my-1 privacyPoliceBtn">Ok, rozumiem  </button>  </div>  @RenderSection("scripts", required: false) |

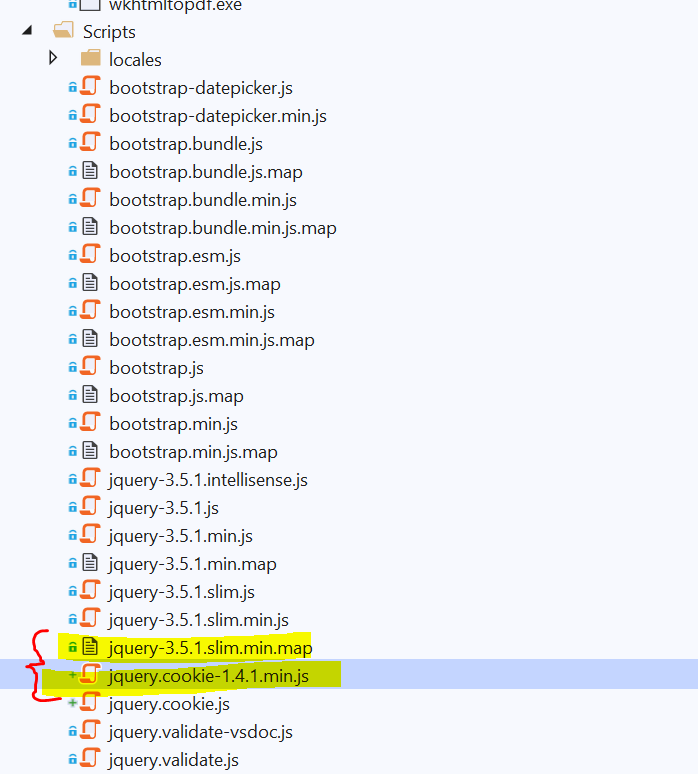
Do site.css

|  |
| --- |
| .privacyPoliceInfo{  font-size: 10px;  position: fixed;  bottom: 0;  background-color: #464646;  width: 100%;  color: lightblue;  text-align: center;  padding: 5px 0px;  }  .privacyPoliceBtn{  font-size: 12px;  } |

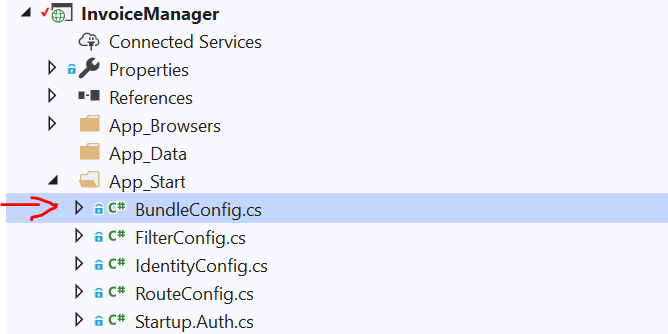
Aby dodać obsługę ciasteczek po stronie widoku za pomocą jQuery, musimy dodać odpowiednią bibliotekę **jQuery Cookie**.



Po zainstalowaniu pojawiły się nowe skrypty



Musimy je dodać do naszego bundle’a.



|  |
| --- |
| public static void RegisterBundles(BundleCollection bundles)  {  bundles.Add(new ScriptBundle("~/bundles/jquery").Include(  "~/Scripts/jquery.cookie.js",  "~/Scripts/jquery-{version}.js")); |

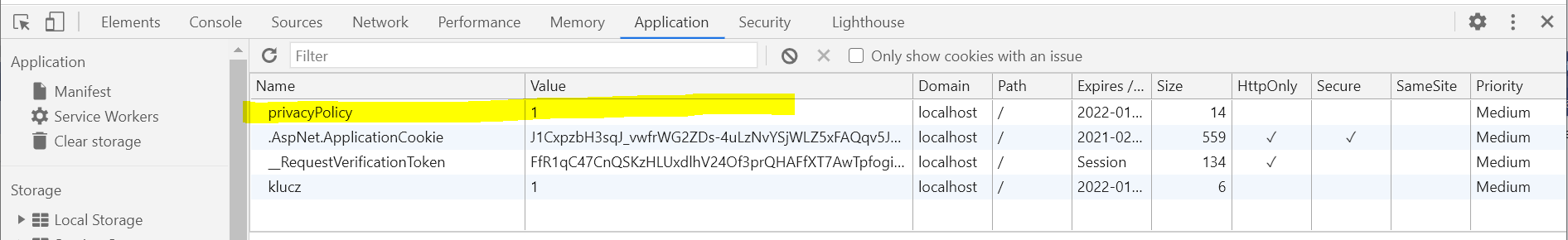
Dodajemy skrypt, a w nim funkcję jaka zostanie wywołana po kliknięciu w przycisk.  
Oraz będziemy potrzebowali metody jQuery, która będzie wywoływana po załadowaniu się dokumentu.

Będziemy mieli dedykowane cookie, które będzie informowało o tym, czy dany użytkownik kliknął już w ten przycisk.

W funkcji wywoływanej po załadowaniu dokumentu będziemy sprawdzali wartość w pliku cookies :  
if ($.cookie("privacyPolicy") !== "1")  
jeżeli jest tam 1 to będziemy kasować styl  
$('.privacyPoliceInfo').css("display", ""); Ważne aby między cudzysłowami nie było spacji  
czyli jeżeli było display: none to teraz nie będzie żadnego stylu czyli będzie widoczne

Natomiast w funkcji po kliknięciu będziemy ustawiać display na none – czyli ukrywać  
$('.privacyPoliceInfo').css("display", "none");  
oraz będziemy ustawiać 1 w naszym ciasteczku:  
$.cookie("privacyPolicy", "1", { path='/', expires = 365});  
oraz dodatkowo w kolejnym parametrze informujemy za pomocą path: ‘/’, że dotyczy to wszystkich stron oraz, że ciasteczko wygasa po 365 dniach.

|  |
| --- |
| <div class="privacyPoliceInfo" style="display: none;">  Serwis wykorzystuje pliki cookies. Korzystając ze strony wyrażasz zgodę na wykorzstanie  plików cookies.  <button onclick="closePrivacyPolice()" type="button"  class="btn btn-light btn-sm mx-3 my-1 privacyPoliceBtn">Ok, rozumiem  </button>  </div>  <script>  function closePrivacyPolice() {  $('.privacyPoliceInfo').css("display", "none");  $.cookie("privacyPolicy", "1", { path: '/', expires: 365 });  }  $(document).ready(function () {  if ($.cookie("privacyPolicy") !== "1")  $('.privacyPoliceInfo').css("display", "");  });  </script> |



int x = 1;

int y = (int)x;

int u = int.Parse(x); parameter musi być stringiem

string a = "7";

int z = int.Parse(a);

int v = (int)a; nie można rzutować stringa na int