|  |
| --- |
| **Katolicki Uniwersytet Lubelski Jana Pawła II**  **Wydział Matematyki, Informatyki i Architektury Krajobrazu**  **Instytut Matematyki i Informatyki** |
| **Informatyka, studia stacjonarne I stopnia** |
| **Małgorzata Leszczuk**  Nr albumu 134459 |
| **Tytuł pracy Walidacja danych wprowadzanych przez użytkownika z wykorzystaniem wzorca MVC na platformie ASP.NET** |
| Praca licencjacka napisana na seminarium  Bazy danychpod kierunkiem **dr Marcina Płonkowskiego** |
| **Lublin 2016** |

**Spis treści**

[Wstęp 3](#_Toc453268703)

[Generowanie stron internetowych 4](#_Toc453268704)

[1.1. Historia stron internetowych 4](#_Toc453268705)

[1.2. Technologie przydatne w tworzeniu stron internetowych 5](#_Toc453268706)

[1.3. Historia JavaScript 8](#_Toc453268707)

[1.4. Komunikacja pomiędzy klientem a serwerem 9](#_Toc453268708)

[1.5. Podsumowanie 10](#_Toc453268709)

[ASP.NET MVC 11](#_Toc453268710)

[2.1. Platforma .NET 11](#_Toc453268711)

[2.2. Entity Framework 12](#_Toc453268712)

[2.3. MODEL-WIDOK-KONTROLER 13](#_Toc453268713)

[2.4. MVC jako wzorzec złożony 16](#_Toc453268714)

[2.5. Cykl życia aplikacji w ASP.NET MVC 17](#_Toc453268715)

[2.6. Rozwój ASP.NET na przestrzeni lat 19](#_Toc453268716)

[2.7. Podsumowanie 23](#_Toc453268717)

[Kontrola poprawności wprowadzanych danych 24](#_Toc453268718)

[3.1. Walidacja danych 24](#_Toc453268719)

[3.2. Funkcje walidacyjne z których korzysta .NET 24](#_Toc453268720)

[3.3. Wyrażenia regularne 25](#_Toc453268721)

[3.4. Autoryzacja jako ważne narzędzie przy korzystaniu z użytkowników. 27](#_Toc453268722)

[3.5. Najczęściej spotykane ataki. 28](#_Toc453268723)

[3.6. Podsumowanie 29](#_Toc453268724)

[Projekt aplikacji i testowanie 31](#_Toc453268725)

[4.1. Opis aplikacji 31](#_Toc453268726)

[4.2. Testowanie 32](#_Toc453268727)

[4.3. Podsumowanie 36](#_Toc453268728)

[Bibliografia 37](#_Toc453268729)

# Wstęp

W dzisiejszych czasach nie wyobrażamy sobie świata bez dostępu do Internetu. Ciągły rozwój sieci komputerowych oraz najnowsze technologię prezentują szereg możliwości dla każdego użytkownika. Nowe pomysły i rozwiązania niosą za sobą atrakcyjne korzyści. Niestety każdy medal ma także drugą stronę, za którą kryję się bezpieczeństwo treści udostępnianych w Internecie. Niepowołany dostęp osób trzecich godzi w nasze poczucie bezpieczeństwa oraz może spowodować nieodwracalne skutki. Walidacja danych, która jest głównym tematem tej pracy licencjackiej to bardzo ważne zagadnienie zarówno dla programistów, jak i dla zwykłych użytkowników korzystających ze strony.

W rozdziale pierwszym zostaną omówione wszystkie możliwości jak można stworzyć swoją stronę internetową. Poznamy sposób programowania po stronie serwera i klienta. Będziemy mogli odróżnić programowanie dynamiczne od statystycznego oraz jakie składniki się w nim znajdują.

W rozdziale drugim skupimy się na technologii stworzonej przez firmę Microsoft. Poznamy strukturę .NET która udostępnia wiele przydatnych bibliotek oraz rozwiązań ułatwiających czytelność kodu. Model-Widok-Kontroler ułatwi nam oddzielanie aplikacji na 3 bardzo ściśle współpracujące ze sobą warstwy. Dowiemy się jak połączyć projekt z bazą danych oraz w jaki sposób można się z nią komunikować. Najważniejszym zagadnieniem w tym rozdziale będzie cykl życia aplikacji, który pozwoli nam zrozumieć działanie ASP.NET.

W rozdziale trzecim poznamy najczęściej występujące zagrożenia dla aplikacji webowych oraz sposób w jaki możemy się przed nimi ustrzec. Po zapoznaniu się w tym rozdziałem będziemy wiedzieli jak działa walidacja danych oraz jak stworzyć wyrażenie regularne. Aby aplikacja webowa utworzona na potrzeby pracy licencjackiej w łatwy sposób współpracowała z użytkownikami dostępnymi na stronie omówimy sposób autoryzacji użytkowników na przykładzie filtrów autoryzacji.

Aplikacja, która zostanie omówiona w ostatnim rozdziale udostępnia mechanizm do zalogowania się oraz wiele innych funkcji dostępnych tylko dla zarejestrowanych użytkowników. W czwartym rozdziale przyjrzymy się szczegółowo temu w jaki sposób możemy w niej pracować.

Rozdział I

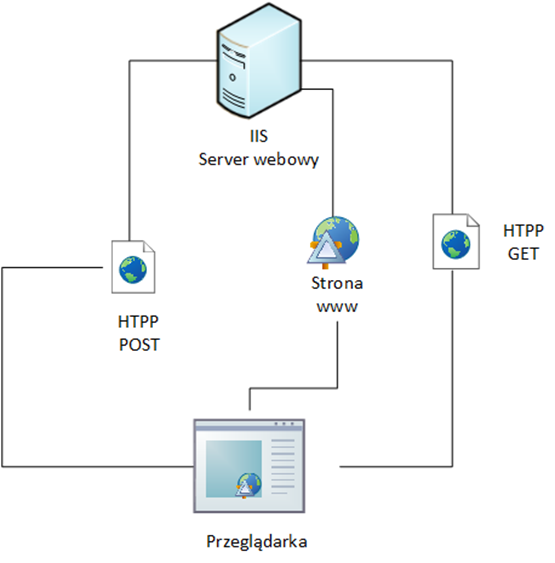
# Generowanie stron internetowych

## Historia stron internetowych

Przy początkowym generowaniu stron serwer sieci web był odpowiedzialny za przekazywanie treści, zaś przeglądarka służyła jako klient, który je udostępnia. Przesyłane pliki nagłówkowe [16]:

* HTTP GET – wywołuję żądanie danych z określonego zasobu. Tworząc w ten sposób ciąg znaków, które zostają podpięte do adresu strony. Wszystkie polecenia są zapisywane w pamięci przeglądarki, używa ich się tylko i wyłącznie do pobierania danych, które nie są poufne. Ich długość jest ograniczona.
* Server webowy w odpowiedzi przekazuję odpowiednio wygenerowaną stronę
* HTTP POST – przekazuję dane w celu przetworzenia do odpowiedniego zasobu. Jest przeciwieństwem HTTP GET.

Wszelkie zależności oraz sposób działania przedstawiono na poniższym rysunku.



Rys. 1.1 Komunikacja przeglądarki z serwerem webowym.

Wyróżniamy dwa sposoby tworzenia stron [16]:

* statyczne - kolejne odświeżenie strony nie zmienia jej wyglądu lub też zawartości. Zmiany na stronie są zainicjowane przez programistę, który ręcznie wprowadza modyfikację w kodzie, a co za tym idzie wyglądzie strony. Forma witryny internetowej znana jest już w fazie planu i projektu.
* dynamiczne - odświeżenie strony jest aktualizowane przez serwer HTTP, który modyfikuję zawartość w oparciu o parametry lub zmienne przekazane przez przeglądarkę.

Jeśli zamiarem programisty tworzącego stronę jest tworzenie złożonych i skomplikowanych witryn internetowych to bardzo często korzysta z obydwu metod równocześnie.

## Technologie przydatne w tworzeniu stron internetowych

Każda strona internetowa powinna być stworzona zgodnie z trzema podstawowymi warstwami:

* warstwa struktury lub treści – HTML. Hipertekstowy język znaczników jest odpowiedzialny za treść witryny.
* warstwa stylu lub prezentacji – CSS. Kaskadowe arkusze stylów - przedstawiają wygląd dokumentu.
* warstwa zachowania – JavaScript. Odpowiedzialna za interakcję z użytkownikiem, dynamiczne aktualizowanie strony oraz jej zmianę w oparciu o stworzony przez programistę zestaw warunków.

**HTML: Hypertext Markup Language** – *pol. hipertekstowy język znaczników*. Jest podstawowym i najprostszym sposobem dostarczania stworzonych stron do przeglądarki internetowej. W jego skład wchodzą znaczniki pozwalające na formatowanie wyświetlanego tekstu. Udostępnia także możliwość umieszczania obrazków oraz tworzenia sekcji i tabel. [8]

Na przestrzeni lat język HTML przeszedł wiele zabiegów mających na celu poprawę czytelności kodu. Rozwój stron WWW przyczynił się w dużej mierze do łączenia hipertekstowego języka z innymi językami usprawniającymi wygląd. Programiści tworzący wspomniany język postanowili poprawić zagnieżdżanie znaczników, tworząc substytut w postaci języka xHTML. Przez zdecydowaną większość czasu używano na przemian języka HTML posiłkując się xHTMLem.

Cechy wspólne obydwu języków:

* dane są przedstawione w postaci znaczników i tekstu,
* znaczniki mają możliwość korzystania z atrybutów, które zawierają wartości (zaprezentowane jako cudzysłów).
* wygląd znaczników rozpoczęcia oraz zakończenia.

Rekomendowanym językiem przez organizacje standardu pisania oraz przekazywania stron (W3C) jest XML. Hipertekstowy język znaczników dodatkowo można spotkać również w wersji dynamicznej (DHTML). [16] Udostępnia możliwość zmiany elementów na stronie bez wracania do serwera sieci Web. W jego skład wchodzą takie właściwości jak DOM (ang. *Document Object Model*), skrypty ( napisane w języku JavaScript), CSS (ang. *Cascading Style Sheets*), xHTML.

Funkcje DHTML:

* pozycjonowanie strony w czasie rzeczywistym,
* możliwość zmiany znaczników i właściwości,
* powiązania danych.

HTML udostępnia swoich pomocników tzw. HTML HELPERS, którzy gwarantują wiele możliwości ułatwiających czytelność kodu i prostotę. [14] Aby rozpocząć formularz można skorzystać z polecenia BeginForm, BeginRouteForm. Zakończenie formularza jest możliwe przy użyciu EndForm. Funkcje obsługujące pola formularza:

* Checkbox,CheckboxFor – odpowiednik pola wyboru, w nawiasach podajemy wartość ciągu znaków jakie mogą być wybrane lub odznaczone. Najlepiej aby klasa je opisująca była typu prawda/fałsz.
* Hidden,HiddenFor – pole ukryte.
* Password – pole hasła.
* RadioButton,RadioButtonFor – pole radia.
* TextBox, TextBoxFor – pole tekstowe formularza.
* Label,LabelFor – etykieta.

Inne przydatne funkcje:

* TextArea, TextAreaFor – pole tekstowe.
* ListBox,ListBoxFor – lista.
* DropDownList, DropDownListFor – lista rozwijana.
* EnumDropDownListFor – jest rozszerzoną funkcją listy rozwijalnej, obsługuję dodatkowo klasy typu wyliczeniowego.

Funkcje służące do przekierowania:

* ActionLink przenosi użytkownika do odpowiedniej podstrony.

Wykorzystajmy zdobytą wiedzę w praktyce na przykładzie modelu MVC, o którym dowiemy się więcej w drugim rozdziale. Chcielibyśmy uzyskać dostęp do strony głównej podstrony o nazwie Schronisko. Ścieżka dostępu będzie wyglądać następująco: */Schronisko/Index*

Zaś kod w postaci odnośnika:

@Html.ActionLink("Wykaz schronisk", "Index", "Schronisko")

Przykład 1.1 Przykładowe użycie ActionLink

* Trasowanie odnośników obsługiwane jest poprzez RouteLink, który działa analogicznie do poprzedniej funkcji.

Funkcje walidacyjne:

* ValidationSummary – wyświetla zbiorowy komunikat o walidacji.
* ValidationMesseage – w przeciwieństwie do poprzedniego pomocnika informuję o jednym elemencie.

Do walidacji danych wrócimy w rozdziale trzecim.

W dzisiejszych czasach nie wyobrażamy sobie strony internetowej, która nie jest w oprawiona grafiką lub nie ma przyjaznego interfejsu. Firmy lub instytucję chcąc zachęcić potencjalnych kontrahentów do współpracy tworzą aplikacje webowe umożliwiające załatwienie spraw bez wychodzenia z domu. Gdybyśmy mieli zająć się stroną tylko przy użyciu znaczników HTML, powtarzając niepotrzebnie kod tworzenie strony internetowej adekwatnej do obecnie obowiązujących trendów zajęłoby całe wieki. Chcąc tego uniknąć wprowadzono CSS.

**CSS: Cascading Style Sheets** (*pol. kaskadowe arkusze tekstowe*) umożliwiają generowanie witryny internetowej od strony wizualnej poprzez zmianę koloru czcionki, tła czy też marginesów. Wystarczy raz zdefiniować określony styl, który jest potem wielokrotnie używany.

Bardzo dużą popularność zyskuję framework Bootstrap bazujący na CSS.W łatwy i prosty sposób można wygenerować interfejs graficzny do stron lub aplikacji webowych.

Struktura dokumentu jest również ważnym elementem naszej witryny internetowej. Zajmuję się nią opisany model **Document Object Model** (*pol. obiektowy model dokumentu)* w skrócie DOM pozwalający na ułożenie struktury dokumentu.

Nie wyobrażamy sobie strony w której nie będziemy mogli skorzystać ze skryptów dostępnych po stronie klienta jak i serwera. **ECMAScript** zajmuje się definicją składni języka służącego do tworzenia skryptów. Jego przedstawicielem jest JavaScript.

Strony internetowe są dzielone na dwie bardzo istotne gałęzie ściśle współpracujące ze sobą. Możemy je podzielić ze względu na oprogramowanie na[16]:

* front-end - odbiera dane od osoby korzystającej ze strony, informując back-end o zmianach. Przykładowe technologie: CSS, HTML , JavaScript.
* back-end - jest to zbiór skryptów dostępnych po stronie serwera. Korzystają z niego wszystkie języki oparte na CGI - interfejsie, który umożliwia porozumiewanie się z oprogramowaniem będącym częścią serwera, również innymi które się na nim znajdują. Przykładami są chociażby PHP i ASP.

## Historia JavaScript

W 1995 producent przeglądarki Netscape postanowił tworzyć język, który pozwoli na używanie Javy w przeglądarce. Początkową nazwą był LiveScript. Zostały wtedy dołączone aplety języka Java napisane w JavaScript. Język odniósł wielki sukces z powodu łatwości użycia i możliwości szybkiej implementacji wymaganych funkcji. [6]

Po wydaniu kilku wersji skupiono się na bezpieczeństwie i zablokowaniu niebezpiecznych operacji. Przez jakiś czas przeglądarka Internet Explorer nie obsługiwała języka JavaScript tworząc jego substytut w postaci Jscript. Programiści chcąc tworzyć swoje strony niezależne od używanej przeglądarki musieli używać obydwu języków. Niebawem podjęto próbę stworzenia standaryzacji i połączenia ich w jeden - tak powstał EMCAScript. Później narodził się DOM - obiektowy model dokumentu, który spowodował zmiany w separacji trzech struktur.

W 1999 roku Microsoft opracował pierwszą wersję AJAX w formacie ActiveX pod nazwą XMLHTTP, co umożliwiło aktualizowanie zawartości dokumentu HTML bez odświeżania okna. Szybka popularyzacja tego trendu spowodowała nazwanie jej XMLHttpRequest. W 2005 zaczęto używać terminu Ajax.Najbardziej popularną biblioteką JavaScript jest jQuery stworzona przez J. Resiga. Umożliwia dokonanie zmian w modelu DOM.

Technologia AJAX (ang. Asynchronous JavaScript And XML) łączy w sposób asynchroniczny dwa języki - Javascript oraz XML. Służy do generowania aplikacji internetowych, interakcja między użytkownikiem odbywa się w sposób asynchroniczny - wszystkie dokumenty nie muszą być ładowane od nowa.[16].W skład technologii AJAX wchodzą następujące języki:

* XMLHttpRequest - asynchroniczne przekazywanie informacji.
* JavaScript- język skryptowy po stronie użytkownika.
* XML- język znaczników, w którym są zaprogramowane otrzymane informacje.

W praktyce można dodatkowo korzystać ze skryptów umożliwiających komunikację z bazą danych.

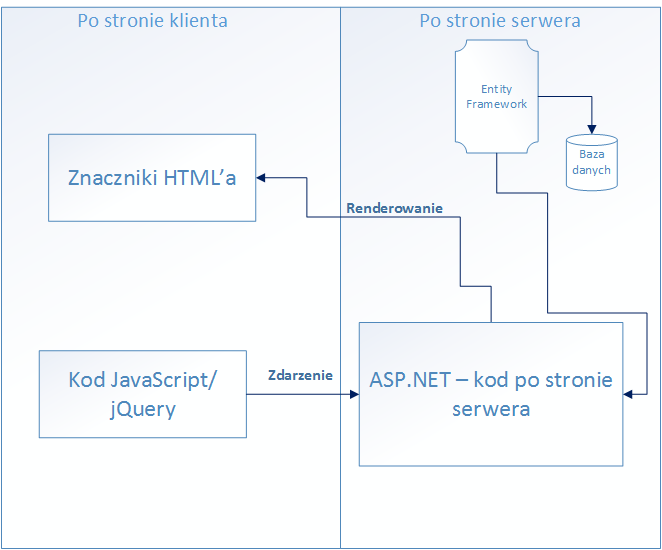
## Komunikacja pomiędzy klientem a serwerem

Platforma ASP.NET MVC pełni rolę serwera. Przesyła dane w postaci znaczników HTML do przeglądarki, która tworzy zaprogramowany wcześniej interfejs dla użytkownika oraz pozwala na korzystanie z strony WWW. JQUERY umożliwia przesłanie danych z formularza w postaci kodu, a następnie serwer umieszcza je w bazie danych.

ASP.NET stara się zabezpieczyć witrynę przed trzema problemami, takimi jak:

* Izolacja danych – korzystając z aplikacji po stronie klienta, nie mamy dostępu do wartości przeznaczonych dla serwera. To rozwiązanie pozwala na zabezpieczenie bazy danych.
* Bezpieczeństwo – zapewniając kod jedynie po stronie klienta, pomijając serwer – dajemy możliwość łatwiejszego zrozumienia i manipulowania aplikacją w sposób nieautoryzowany. Aplikacja webowa korzystająca z obydwu sposobów jest mniej narażona na ataki.
* Różne urządzenia – utworzona witryna może być przeglądana na telefonie, tablecie oraz komputerze. Elastyczność ASP.NET pozwala na stworzenie dobrze zaprojektowanej aplikacji na wszystkich wspominanych sprzętach. [12]

Szczegółowy schemat działania przedstawia Rysunek 1.2.



Rys. 1.2 Komunikacja pomiędzy klientem a serwerem.

W dzisiejszych czasach nikt nie tworzy aplikacji działających tylko po stronie klienta. W zdecydowanej większości aplikacje są tworzone po stronie klienta a następnie łączą się serwerem.

## Podsumowanie

W tym rozdziale poznaliśmy metody tworzenia stron oraz ich historię. Potrafimy odróżnić języki programowania działające po stronie serwera od tych działających po stronie klienta. W kolejnym rozdziale skupimy się na technologii ASP.NET MVC.

Rozdział II

# ASP.NET MVC

## Platforma .NET

W tym rozdziale poznamy technologię ASP.NET MVC[10]. Stworzona przez firmę Microsoft platforma programistyczna, która pozwala tworzyć aplikacje dzieląc się na dwa moduły:

* CLR (ang. Common Language Runtime) – wspólne środowisko rozruchowe, które pełni rolę silnika i odpowiada za eksploatacje uruchomionych aplikacji.
* FLC (.NET Framework) - biblioteki klas, które pozwalają na dodawanie wielu przydatnych funkcji w programie.

.NET Framework pozwala tworzyć oprogramowanie w wielu językach programowania. Przy tworzeniu .NET Framework firma Microsoft bazowała na wspomnianym wyżej CLR odpowiedzialnym za generowanie kodu i kompilacje przygotowanych przez programistę aplikacji opartych na wspólnym języku niskiego poziomu CIL (ang. *Common Intermediate Language*) - poprzednia nazwa MSIL ( ang. *Microsoft Intermediate Language*).

W składnikami środowiska jest:

* Standardowy zestaw typów danych (CLR), który gwarantuję bezpieczeństwo aplikacji oraz udostępnia możliwość pakowania kodu zarządzanego.
* CTS (ang. *Common Type System*) umożliwiający korzystanie z różnorodnych zestawów typów danych

.NET Framework daje możliwość realizowania uniwersalnych zagadnień programistycznych, takich jak dostęp do pliku, składowanie danych, podłączenie się pod utworzoną bazę danych. Dodatkowo wspomniana wyżej biblioteka posiada typy dające możliwość tworzenia różnych ścieżek rozwoju naszej aplikacji.

Platforma .NET pozwala użytkownikowi na generowanie dynamicznych stron przy użyciu:

* ADO (ang. *ActiveX Data Objects*) - daje dostęp do bazy danych.
* ADO.NET (ang. *ActiveX Data Objects for .NET* ) - pozwalający na swobodną komunikację z dowolną bazą danych.
* ASP (ang. *Active Server Pages*) – tworzenie stron
* ASP.NET - dodatkowo daje możliwość korzystania z XML(ang. *eXtensible Markup Language*), pozwala na standardowy dostęp do programu oraz jego danych, dowolność języka programowania.

## Entity Framework

Biblioteka domyślnie zainstalowana wraz z frameworkiem ASP.NET, która jest pośrednikiem w dostępie do danych. Głównym celem jej stosowania jest zmiana z struktury obiektowej na bazę danych. Taki zabieg nazywa się ORM – ang. *Object Relational Mapping.* [1].

Jej działanie opiera się na modelu związków encji – ERM – ang. *Entity Relationship Model*. Od wersji 4.1 istnieje możliwość tworzenia projektu według trzech sposobów:

* Database First – jest to najstarsze podejście polegające na podłączeniu projektu pod istniejąca już bazę danych, przy użyciu narzędzia Entity Data Model Wizard. Wszelkie zmiany z nią związane zachodzą w Entity Designer Data Model.
* Code First – upoważnia do zmiany modelu w postaci klas na relacyjną bazę danych. Tworzą one związki przy użyciu referencji, adnotacje przy tworzeniu obiektów klas pozwalają na określenie czy utworzony obiekt jest kluczem i elementem wymaganym. Kolejnym krokiem jest stworzenie bazy danych, która musi być odświeżana po każdej zmianie w modelu i zainicjowana od nowa przy użyciu narzędzia migracji. Visual Studio po nazwie obiektu wybiera jego przeznaczenie. Dla przykładu nazwanie go ID na początku lub pod koniec automatycznie kojarzy go jako klucz główny.
* Model First – Visual Studio pozwala na skorzystanie z graficznego interfejsu w którym tworzy się model. Kolejnym krokiem jest uzupełnienie go o encje i utworzenie między nimi relacji(związków). Po wykonaniu wspomnianych wyżej czynności następuję produkowanie struktury bazy danych napisanej w języku DDL (ang. *Data Definition Language*). Zmiany w modelu wymuszają odwołanie się do widoku, a następnie ponowne wygenerowanie modelu fizycznego. Każda zmiana w strukturze bazy egzekwuję rozpoczęcie tworzenia tabel od nowa poprzez pozbycie się obecnej wersji i stworzenie nowej od zera.

CRUD jest skrótem angielskich słów Create Read Update Delete. Służy do automatycznego stworzenia kontrolera w technologii ASP.NET MVC. Możemy zarządzać danymi usuwać je edytować, aktualizować oraz tworzyć. Istnieje rozbudowana wersja w postaci SCRUD, dająca dodatkową możliwość przeszukiwania bazy danych.

Baza danych mająca tylko jedną tabelę staję bezużyteczna oraz nadmiarowa. Tworząc kolejne tabele, musimy je połączyć. Bardzo ważnym etapem tworzenia konceptualnego modelu jest dodanie relacji zachodzących pomiędzy klasami i tabelami – przy użyciu asocjacji.

Możemy tworzyć związki na kilka sposobów:

* 1-1(jeden-do-jednego).
* 1-N (jeden-do-wielu).
* N-M (wiele-do-wielu).

Istotnym zagadnieniem tworząc bazę danych jest podłączenie jej z naszym projektem przy użyciu narzędzia *connection string*. Zainicjowanie odpowiednich parametrów będących jego częścią pozwala na komunikację z utworzoną bazą danych.

Jego składnikami są[17]:

* Data Source – w tym miejscu określamy instancję bazy danych.
* AttachDbFileName – nazwa pliku wraz z ścieżką dostępu do niego
* ID – autoryzacja do bazy danych poprzez login
* Initial Catalog – nazwa bazy
* Integrated Security – wyraża w jaki sposób należy się logować do instancji.
* False – logujemy się przy użyciu parametrów ID i Password
* True – możliwość zalogowania się tylko za pośrednictwem systemu
* SSPI – potwierdza logowanie przez system operacyjny, gdy brak ID i Password
* MultipleActiveResultSets – w skrócie MARS, pozwala na wykorzystywanie wielu instancji w jednym połączeniu.
* Password – określenie hasła.
* User Instance – używa się tego składnika tylko i wyłącznie w przypadku SQL Server Express określonej jako TRUE – generując instancję pozwalającą użytkownikowi być administratorem.

Wszelkie zmiany zachodzą w pliku Web.config.

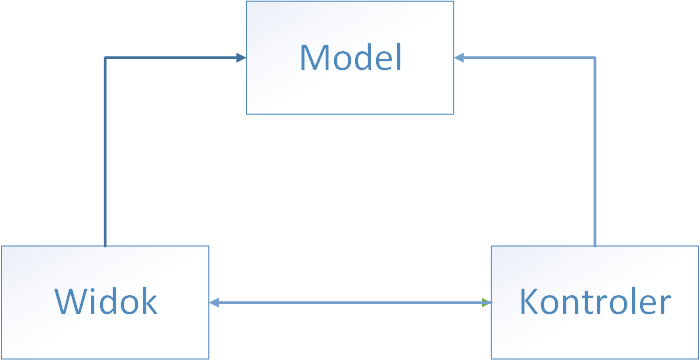
## MODEL-WIDOK-KONTROLER

MVC(ang. *Model-View-Controller*) od wielu lat jest bardzo popularnym architektonicznym wzorcem projektowym. W roku 1979 pierwotna nazwa tego wzorca brzmiała Thing-Model-View-Editor. Wspomniane trzy warstwy ułatwiają projektowanie aplikacji webowych. Wzorzec MVC możemy spotkać w wielu platformach m.in..:PHP, Java i ASP.NET. Korzysta z niego coraz więcej różnych bibliotek i języków programowania. [1]

Architektoniczny wzorzec projektowy dzieli interfejs użytkownika na trzy najważniejsze aspekty:

* Model - zestaw klas opisujący dane wraz z instrukcją do ich zarządzania.
* Widok - opisuję wygląd interfejsu użytkownika dla konkretnej aplikacji
* Kontroler - odpowiedzialny za przepływ informacji pomiędzy widokiem a modelem.

Na powyższym rysunku przedstawiono zależności w model-widok-kontroler.

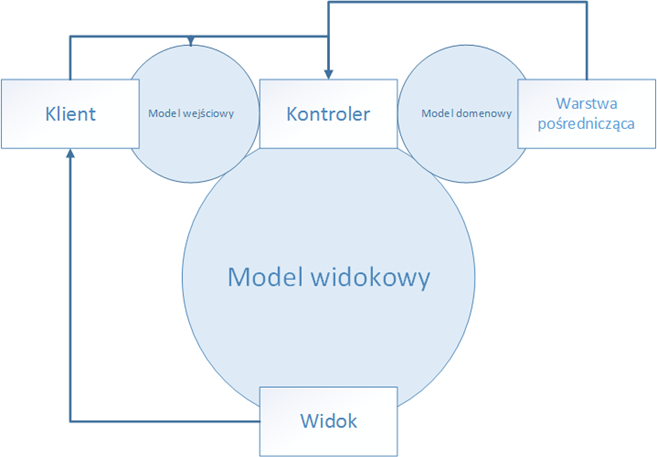


Rys. 2.1 MVC

Wiemy w jaki sposób następuję komunikacja w postaci ogólnej. Przyjrzymy się teraz na MVC ze względu na platformę ASP.NET.

**Model** posiada klasy reprezentujące domeny, których obiekty poprzez hermetyzację danych są elementami bazy danych. Rozróżniamy trzy rodzaje modeli. Domenowy znajduję się pomiędzy kontrolerem a warstwą pośredniczącą, której reprezentantem jest np. baza danych. Opisuję wybrany fragment rzeczywistości. Widokowy przetwarza dane przesyłane do klienta, jest w sąsiedztwie widoku i kontrolera. Wejściowy – sporządza dane, które będą możliwe do wprowadzenia poprzez URL lub formularz. [1]

Na poniższym rysunku przedstawiono zależności.



Rys. 2.2 Zależności pomiędzy modelami.

**Widok** nakreśla schemat do dynamicznego generowania HTML. Tworząc widoki na platformie ASP.NET MVC mamy możliwość skorzystania z gotowych rozwiązań takich jak:

* CREATE – daje możliwość utworzenia widoku generującego rekord w bazie danych
* EDIT – edycja rekordu w bazie
* DELETE – usunięcie rekordu
* DETAILS – wyświetla szczegóły o wybranym rekordzie.
* EMPTY – pusty widok w pełni do dyspozycji.

Aby ułatwić sobie pracę można skorzystać z automatycznego tworzenia 4 pierwszych sposobów przy pomocy Scaffold Item - MVC 5 Controller with views, using Entity Framework

**Kontroler** jest zestawem odpowiednich klas służących do zarządzania i wymiany informacji pomiędzy dwoma wyżej wymienionymi elementami. Reaguję na interakcje ze strony użytkownika, wprowadzając odpowiednie zmiany w widoku projektu.

Wykorzystajmy wiedzę w praktyce. Chcielibyśmy przejść do zakładki About. Kontroler obsługujący nasze żądanie prezentuję się w sposób następujący:

public class HomeController : Controller

{

public ActionResult About()

{

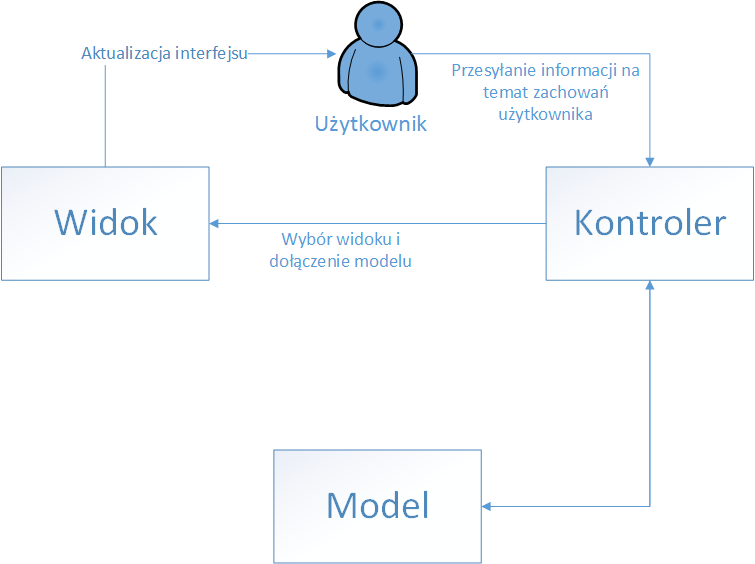
return View();

}

}

Przykład 2.1 Kontroler zwracający widok podstrony About

Najważniejszą rolę w naszej aplikacji będzie pełnił użytkownik, który będzie miał realny wpływ na jej funkcjonowanie. Na poniższym rysunku przedstawiono interakcję użytkownika ze wszystkimi warstwami MVC.



Przykład 2.2 Rola użytkownika w modelu MVC.

Popularność MVC spowodowała stworzenie jego rozmaitych wariantów, takich jak: MVP (Model-Widok-Prezenter), PAC (Prezentacja-Abstrakcja-Kontrola), MVVM (Model-Widok-Widok Modelu).

## MVC jako wzorzec złożony

Nie zapominajmy, że wzorce projektowe są nieodłącznym elementem życia każdego programisty. Ich znajomość może ułatwić, lecz na początku sprawić też wiele problemów. Warto dowiedzieć się dokładnej genezy powstania samego MVC. Czynniki wpływające na jego obecny wygląd oraz poznać inne wzorce działające na podobnej zasadzie. Na początku przybliżymy sobie wzorzec czynnościowy pod nazwą Obserwator.

**Obserwator** opisuję relację jeden do wielu pomiędzy obiektami. W przypadku kiedy stan jednego z nich ulegnie zmianie to inne z nim powiązane są aktualizowane i na bieżąco poinformowane o jego statusie. Można spotkać ten wzorzec projektowy także pod nazwą Dependents (pol. *Zależni*) lub Publish-Subscribe (pol. *Publikuj-Prenumeruj*) Posiada obiekty: obserwowany oraz obserwator. Ilość obserwatorów związanych z jednym obserwowanym jest nieograniczona. Kiedy obserwowany modyfikuję swój stan - jego obserwatorzy są o tym informowani. W celu synchronizacji po obydwu stronach obserwator odpowiada zapytaniem o status obserwowanego. [4]

**Strategia** podobnie jak obserwator należy do rodziny obiektowych wzorców czynnościowych. Spotykany także jako Policy (pol. *Polityka*). Zajmuję się tworzeniem klas algorytmów. Dodatkowo daje możliwość zmiany używanych algorytmów, która jest niezależna od obiektów korzystających. [4]

**Kompozyt** jest przykładem obiektowego wzorca strukturalnego. Zestawia obiekty w drzewa hierarchiczne typu część-całość (relacja pomiędzy dwoma terminami, jeden z nich przedstawia całość obiektu, zaś drugi opisuję część całości). Pozwala użytkownikom identycznie odnosić się do pojedynczych jak i złożonych obiektów.

MVC korzysta także z innych wzorców projektowych. Takich jak Metoda Wytwórcza oraz Dekorator. Jednak najważniejszymi są opisane 3 wymienione wyżej wzorce - Obserwator, Strategia i Kompozyt.

## Cykl życia aplikacji w ASP.NET MVC

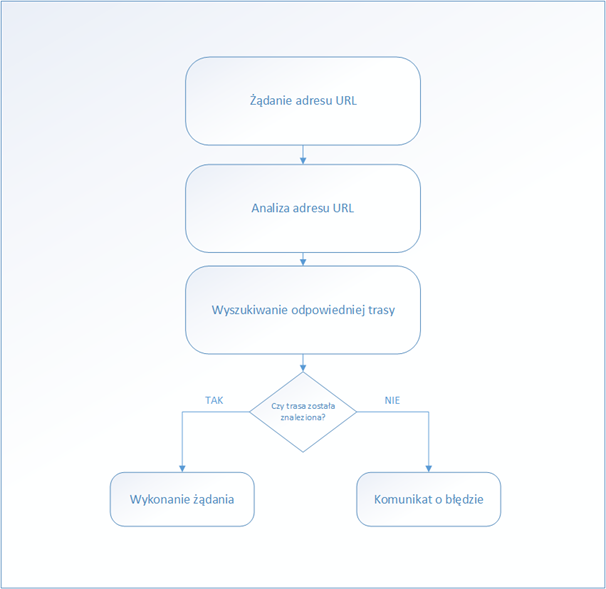
Na chwilę zastanówmy się jak działa aplikacja stworzona w technologii Microsoft. Na samym początku następuję żądanie – aktywowane poprzez kliknięcie w odnośnik na stronie. W tym momencie rozpoczyna się analizowanie ze strony serwera. Jako odpowiedź otrzymywana jest wygenerowana strona dopasowana do odpowiedniej reguły. Wraz z momentem uruchomienia linku poprzez jego kliknięcie lub wpisanie do przeglądarki następuję włączenie HTTP Request, które jest zgłoszeniem do serwera.[17]

Jako pierwszy etap określamy Trasowanie(ang. *Routing*), czyli przyporządkowanie żądania pod określone działanie umieszczone w pliku RouteConfig.cs.

Zakładamy, że domyślnym szablonem będzie Default przyjmujący dwie wartości odpowiedzialne za kontroler i akcję. Jeśli odwiedzamy stronę główną to automatycznie jest ona kojarzona z plikiem /Home/Index.

Warto zauważyć, że kontroler obsługujący dodanie rekordu w bazie danych w tabeli Schronisko – to zostaję interpretowany po stronie serwera jako Schronisko/Create.

Przebieg routingu przedstawiono na poniższym rysunku:



Rys. 2.3 Mechanizm trasowania

Następnie zostaję uruchomiony tak zwany MVCHandler – jego głównym zadaniem jest dobranie odpowiedniego kontrolera pod wywołaną akcję. Cały proces dzieli się na kroki takie jak: Tworzenie kontrolera, Autoryzacja i Uwierzytelnienie, Wiązanie modelu, Wezwanie akcji kontrolera, Przekazanie klas, które dziedziczą po ActionResult. Pierwszy etap tworzenie kontrolera został opisany w podpunkcie 2.3.

**Autoryzacja i Uwierzytelnianie:**

Najważniejszym zadaniem tego filtru jest sprawdzenie, czy użytkownik może korzystać z podstrony do jakiej próbuje uzyskać dostęp. Więcej o filtrach autoryzacji dowiemy się w Rozdziale III.

**Wiązanie modelu:**

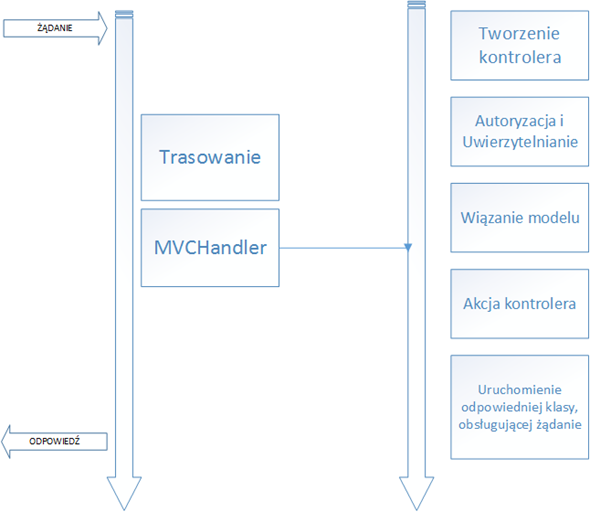
Jest mechanizmem który odwzorowuję dane żądania HTTP na właściwości, takie jak modele i akcje. Skojarzone dane są zwracane jako odpowiedź.

**Wezwanie akcji i przekazywanie klas:**

Po wiązaniu modelu, następuję skojarzenie kontrolera z odpowiednią akcją a następnie jej wywołanie.

Po wszystkich powyższych krokach nasze żądanie zostaję obsłużone, a następnie odesłana odpowiedź zwrotna w jaki sposób aplikacja powinna zareagować.

Na poniższym rysunku przedstawiono cykl życia aplikacji:



**Rys. 2.4.** Cykl życia aplikacji ASP.NET

## Rozwój ASP.NET na przestrzeni lat

W 2002 roku przy tworzeniu pierwszej wersji ASP.NET dominującym nurtem było połączenie metodyki .NET wraz z Web Forms. [7]

ASP.NET od zawsze wspierał abstrakcyjne warstwy:

* System.Web.UI - opisuję metody, zdarzenia oraz definicję używane przez wszystkie kontrolki serwera ASP.NET.
* System.Web - zawiera klasy oraz metody umożliwiające komunikację z przeglądarką. Dodatkowo dzieli się na klasy:
  + HttpRequest - odpowiedzialną za dostarczenie informacji o bieżącym stanie żądania http.
  + HttpResponse - sprawuję kontrolę nad wynikiem ze strony serwera HTTP do klienta.
  + HttpServerUtility - zapewnia zestaw narzędzi dla serwera i procesów.
* Obsługa ciasteczek i wyjątków:
  + Transfer plików.
  + Zarządzanie pamięcią podręczną danych wyjściowych.

Pierwsza wersja:

Scott Guthrie zaprezentował pomysł stworzenia ASP.NET MVC w 2007 roku na konferencji w Redmond. Jego idea została pozytywnie oceniona przez zespół programistów zainteresowany dalszym rozwijaniem przedstawionej technologii. Bazował na licencji open source - co pozwalało na otwarty dostęp do kodu źródłowego.

Po wielu poprawkach i dyskusjach został oficjalnie opublikowany 13 marca 2009 roku.[7]

Druga wersja:

Po zaledwie roku został zmieniony i wyposażony w dodatkowe funkcje takie jak:

* Ulepszone narzędzia w Visual Studio.
* Wprowadzenie pomocników HTML.
* Walidacja danych po stronie serwera i klienta.
* Umieszczenie UL. Helpers odpowiedzialnych za automatyczne połączenie z szablonami.
* Usprawnienie API - możliwość korzystania z nowych funkcji oraz narzędzi.
* Wspiera asynchroniczne kontrolery, dzielenie aplikacji na obszary oraz Html.RenderAction(służy do przetwarzania sekcji na stronie).

Chcąc tworzyć oprogramowanie warto porównać bardzo zbliżone do siebie technologie mianowicie ASP.NET Web Forms z ASP.NET MVC.

ASP.NET MVC:

* Programista ma dostęp do generowanego kodu przez znaczniki HTML.
* Łatwość testowania oraz zmian w aplikacji.
* Podział aplikacji na 3 warstwy.

ASP.NET Web Forms:

* Obsługa zdarzeń i środowiska RAD.
* Automatyczne generowanie strony oraz sprawowanie nad nią kontroli.
* Krótszy kod niż w MVC, możliwość korzystania z gotowych kontrolek.

Trzecia wersja:

Po 10 miesiącach mamy możliwość zapoznania się usprawnieniami jakie narzuca nam najnowsza wersja. Wprowadzono do niej następujące funkcjonalności:

* Poprawienie walidacji.
* Dodanie silnika Razor (w poprzednich wersjach używano silnika Web Forms).
* Używanie NuGet - który pozwala na zarządzanie całą platformą oraz dostarcza potrzebne oprogramowanie.
* Wsparcie dla platformy .NET 4.
* Korzystniejsza współpraca z JavaScript.
* Lepsza kontrola i elastyczność oraz rozwiązywanie zależności ze wsparciem dla filtrów akcji.

Czwarta wersja:

Nie musieliśmy czekać zbyt długo na kolejne możliwości jakie zapewnia nam ASP.NET MVC 4. Usprawniono ją o dodatkowe korzyści, takie jak:

* Dołączenie ASP.NET Web API.
* Dodatki do domyślnych szablonów projektu.
* Możliwość tworzenia mobilnych aplikacji webowych przy użyciu jQuery Mobile.

Nowe możliwości w postaci ASP.NET Web API dają nam szereg usprawnień, ułatwiających programowanie w tej technologii. Takich jak:

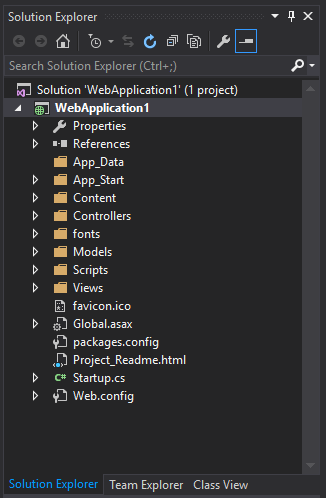
* Korzystanie z tego samego mapowania tras, co akcje kontrolera.
* Automatycznie odwzorowanie znaczenia żądania HTTP do modelu.
* MVC korzysta z filtrów, umożliwiających korzystanie z atrybutów pozwalających na dodawanie reakcji w aplikacji.

Piąta wersja:

Tworząc MVC 5 skupiono się także na rozbudowaniu środowiska programistycznego, jakim jest Visual Studio 2013. Całość opublikowano w październiku i wyposażono o dodatkowe właściwości:

* Ujednolicenie wersji ASP.NET.
* Routing - metoda określenia trasy poprzez umieszczenie adnotacji lub metody działania w kontrolerze klasy.
* Filtry uwierzytelniania.
* Zmiana wyglądu szablonu projektu.
* Generowanie projektu w oparciu o klasy modelu.

Nie zapominajmy, że łatwiejsze zarządzanie projektem jest możliwe poprzez automatycznie wygenerowane foldery obsługujące odpowiednie zadania. Wygląd projektu przedstawiono na poniższym rysunku.



Rys. 2.4 Widok projektu

Opiszmy co znajduję się w danych folderach i plikach w kolejności przedstawionej na rysunku 2.4

* App\_Data – w tym folderze mogą znajdować się pliki z rozszerzeniem .mdf, LocalDB oraz inne związane bezpośrednio z danymi. Mogą być używane tylko do odczytu lub odczytu i zapisu.
* App\_Start – przechowuję pliki konfiguracyjne np. mechanizmu trasowania.
* Content – pliki inne niż skrypty – np. z rozszerzeniem .css.
* Controllers – kontrolery używane przez nasz program.
* Fonts – czcionki.
* Models – modele, których metody publiczne będą używane w przyszłości do manipulowania danymi oraz ich przechowywania.
* Scripts – wszystkie skrypty z jakich korzysta projekt.
* Views – widoki, które będą podzielone na różne funkcje np. dodanie nowego elementu do bazy w postaci atrakcyjnego widoku HTML dla przyszłego użytkownika.

Pliki:

* Global.asax – w tym pliku obsługuję się zdarzenia i w jaki sposób program powinien zareagować np. na początek sesji.
* Packages.config – pliki obsługiwane przez NuGet PacketManagera.
* Startup.cs – zbiór funkcji, które zostaną wywołane wraz z uruchomieniem aplikacji.
* Web.config – konfiguracja poziomu aplikacji.

Dodatkowo w zależności od potrzeb można utworzyć inne przydatne foldery takie jak np. DAL – w którym przechowujemy gotowe rekordy do bazy danych najpierw w pliku obsługującym kontekst bazodanowy, a następnie w pliku tworzącym rekordy. Przy korzystaniu z takiego działania warto wygenerować mechanizm Migracji, który umożliwia dokumentowanie zmian jakie zaszły w związku z aktualizacją bazy danych. Aby nie martwić się o ten proces warto edytować plik Configuration.cs – w którym dopisać AutomaticMigrationsEnabled = true;

## Podsumowanie

W tym rozdziale poznaliśmy technologię ASP.NET MVC. Wiemy w jaki sposób działa architektoniczny wzorzec projektowy. Nasza aplikacja powoli zaczyna układać się w spójną całość. Pozostaję jednak niepokój, czy została ona w odpowiedni sposób zabezpieczona. Tą rozterkę postaramy się rozwiązać w kolejnym rozdziale związanym z walidacją i bezpieczeństwem danych.

Rozdział III

# Kontrola poprawności wprowadzanych danych

## Walidacja danych

Walidacja danych - polega na sprawdzeniu poprawności wprowadzanych przez użytkownika danych. Dodatkowo weryfikuję, czy wymagane pola są wypełnione oraz wyświetla odpowiedni komunikat w przypadku umieszczenia nieodpowiednich informacji.

Wszystkie funkcje związane z walidacją i napisane w nawiasach [] będą realizowane w klasach danej bazy danych.

## Funkcje walidacyjne z których korzysta .NET

Zasady DRY (ang.do not repreat yourself) oraz DIE (ang.duplication is evil) – są obowiązkowym składnikiem MVC – ich głównym założeniem jest nie powielanie kodu, zaś stosowanie funkcji, obsługujących dany problem. [2]

Najczęściej spotykane komponenty związane walidacją danych:

* Required – pole wymagane.
* StringLength – długość znaków
* RegularExpression – wyrażenie regularne
* MinLength – najmniejsza dopuszczalna długość
* MaxLength – największa dopuszczalna długość
* ErrorMessage – komunikat o błędzie.
* Range – przedział w jakim ma być podana wartość

Po zapoznaniu się z powyższymi komponentami możemy spróbować użyć ich w praktyce. Chcielibyśmy ograniczyć wartość pola imię do 50 znaków oraz poinformować użytkownika o źle wprowadzonych danych.

Odnosząc się do początku obecnego rozdziału wszystkie poznane dotychczas możliwości wpisujemy do klasy, w jakiej chcielibyśmy je używać. W naszym przypadku będzie to Imie. Kod jaki musimy dopisać prezentuję się na poniższym przykładzie.

[Required]

[StringLength(50,ErrorMessage = "Wpisane imie jest zbyt długie")]

public string Imie { get; set; }

Przykład 3.1 Pole wymagane i komunikat o błędzie

Nie zapominajmy, że przy użyciu wspominanych właściwości warto jest dołączyć do klasy niezbędne przestrzenie nazw, takie jak: *using System.ComponentModel.DataAnnotations.*

Inne funkcje poprawiające wygląd danych:

* Display – wyświetla tekst widoczny dla użytkownika.
* DateType – odpowiednio formatuję wprowadzane dane w zależności od potrzeb.
* DisplayFormat – pozwala na zdefiniowanie danych w odpowiednim ciągu.

Aplikacja jaką tworzymy powinna w sposób przyjazny dla użytkownika prezentować dane. Możemy na wiele sposobów zmieniać format daty. Chcielibyśmy wyświetlić ją w postaci: dzień, miesiąc, rok. Rozwiązanie naszego problemu jest dostępne w poniższym przykładzie:

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:dd-mm-YYYY}",

ApplyFormatInEditMode = true)]

[Display(Name ="Data dodania do schroniska ")]

public DateTime DataWplywu { get; set; }

Przykład 3.2 Ułożenie daty

## Wyrażenia regularne

Jest to zbiór reguł obejmujący wygląd składni oraz jakie dane mogą przejść przez tak umieszczony formularz. Posiada odpowiednie atrybuty pozwalające zidentyfikować dopuszczalne znaki, ich ilość oraz wiele innych przydatnych elementów. W technologii ASP.NET MVC zajmuję się nimi klasa RegularExpressionAttribute. Aby ułatwić życie programiście zapewniono dodatkowe klasy np. Email, w których nie trzeba wymieniać wzorca.

Przedstawmy teraz różnicę pomiędzy wyrażeniem regularnym pisanym przez programistę, korzystając ze wzorców, a gotowej właściwości z której możemy skorzystać w każdym momencie. Chcielibyśmy zabezpieczyć pole, w którym będziemy wpisywać adres e-mail. Jeśli skorzystamy z wzorców będzie on prezentował się w sposób:

[RegularExpression(@"[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4})]

public string Email { get; set; }

Przykład 3.3 Wyrażenie regularne dla adresu e-mail, przy użyciu RegularExpression

Możemy zrezygnować z użycia RegularExpression i napisać po prostu EMAIL. Przykład dostał przedstawiony na listingu:

[Email]

public string Email { get; set; }

Przykład 3.4 Wyrażenie regularne dla adresu email, korzystając z atrybutu.

Jeśli nasze wyrażenie nie jest opisane przez gotową właściwość powinniśmy zaprzyjaźnić się z poniższą tabela. Aby upewnić się czy Microsoft nie stworzył gotowego rozwiązania musimy przejrzeć dokumentację, a dokładniej opisywaną już wcześniej przestrzeń nazw *System.ComponentModel.DataAnnotations*.

Tabela wyrażeń regularnych:

|  |  |
| --- | --- |
| Wzór | Charakterystyka: |
| [] | Dostosowuję się do wielkości podanych znaków |
| [^] | Znajduję znaki, których nie ujęto w nawiasach. |
| [ - ] | Sprawdza, czy wszystkie znaki są z zakresu np. [a-z] |
| . | Dowolny znak, wyjątkiem jest nowa linia |
| \w | Cyfra lub litera |
| \W | Dowolne wyrażenie nie będące ani literą ani cyfrą |
| \s | Białe znaki |
| \S | Literały, które nie są białymi znakami |
| \d | Dowolna liczba |
| \D | Znaki nie będące liczbą. |
| ^ | Znajduję się na początku linii lub łańcucha |
| $ | Jest na końcu linii lub łańcucha |
| \A | Pojawia się na starcie literału |
| \Z | Przed znakiem nowej linii lub na zakończeniu literału |
| \z | Koniec łańcucha |
| \G | Zaczyna się tam, gdzie kończy poprzedni |
| \b | Jest pomiędzy literą lub cyfrą i wartością nie będąca ani cyfrą ani liczbą |
| \B | Przeciwieństwo \b |
| \n | Nowa linia |
| \t | Tabulator |
| ? | Element będący poprzednikiem występuję jeden raz lub zero. |
| {n} | Poprzednik pojawia się systematycznie n razy |
| {n.} | Poprzednik ukazuję się minimum n razy |
| {n.m} | Poprzednik trafia się minimum n razy, maksimum m razy |
| \* | Poprzednik zsynchronizuję się od zera do więcej razy. |
| + | Poprzednik występuję przynajmniej jeden lub więcej razy |

Tabela 3.1 Tabela wyrażeń regularnych

## Autoryzacja jako ważne narzędzie przy korzystaniu z użytkowników.

Tworząc projekt na platformie ASP.NET mamy możliwość skorzystania z szeregu różnych rozwiązań związanych z autoryzacją użytkownika. Możemy również zrezygnować z autoryzacji, co jest raczej nie wskazane wybierając opcje No Authentication, która pozwala na dostęp do podstron dla wszystkich.

Najprostszym i najszybszym sposobem zalecanym do stworzenia kont dla przyszłych użytkowników jest wybranie opcji Individual User Accounts, w której jest stworzony użytkownik posiadający nazwę oraz hasło. Można również skorzystać z połączenia przez portal społecznościowy np. Facebook. Wszelkie informacje na temat osoby będącej użytkownikiem są przechowywane jako Member.

Jeśli nie chcemy podpinać naszej aplikacji pod portal społecznościowy lub nie czujemy potrzeby tworzenia konta od zera. Warto skorzystać z opcji Organizational Accounts gdzie logujemy się za pomocą Active Directory poprzez użycie np konta Microsoft.

Gdy z naszego serwera korzysta wielu użytkowników najlepszym sposobem będzie wybór Windows Authentication, w którym Uwierzytelnienie następuję przy pomocy IIS(ang***.*** *Internet Information Services).*

Mając utworzonych użytkowników można skorzystać z filtrów autoryzacji, które pozwalają na wyświetlenie odpowiedniej strony po zalogowaniu. Kryteria z jakich podstron może skorzystać uwierzytelniony użytkownik ustalane są w kontrolerze. Filtry autoryzacji można stosować wobec wszystkich zalogowanych, danego użytkownika lub roli. Nad prawidłowym działaniem wspomnianego zagadnienia czuwa klasa AuthorizeAttribute. Aby wybrać jakie właściwości ma mieć dany użytkownik piszemy słowo [Authorize]. Będzie ono wpisane w kontrolerze danej klasy, w metodzie GET.

Przećwiczmy zdobytą wiedzę i spróbujmy coś zaprogramować. Chcielibyśmy uniemożliwić, dodawanie rekordów do bazy danych zwykłym użytkownikom. Ustalamy, że administrator będzie mógł zmieniać naszą bazę danych.

[Authorize=(Roles=”Admin”)]

public ActionResult Create() {

.. }

Przykład 3.5 Zastosowanie filtru autoryzacji

Pokrewnymi filtrami do wyżej wspomnianych są filtry uwierzytelnienia, które identyfikują jaki użytkownik jest obecnie zalogowany lub próbuję się zalogować do systemu.

## Najczęściej spotykane ataki.

Największym zagrożeniem dla aplikacji webowych są ataki. W tym podrozdziale przyjrzymy się atakom, które są bardzo często spotykane i dowiemy się w jaki sposób można im przeciwdziałać.[15] Pierwszym z nich będzie atak na pliki Cookies.

Cross-Site-Scripting – w skrócie XSS – jest atakiem na klienta korzystającego z witryny. Polega na zaszczepieniu kodu JavaScript, który może być uruchomiony w przeglądarce. Zostają wykradzione sesje ciasteczek, umożliwiające pełną swobodę na zalogowanych sesjach użytkownika zaatakowanego. Istnieję możliwość zmiany wyglądu przeglądanej strony na stronę stworzoną przez Hakera. Dodatkowo atak pozwala na uruchomienie keyloggera, który sprawdza klikanie przez użytkownika klawisze i może dzięki temu przejąć hasła dostępu.

Istnieją 2 najczęściej spotykane odmiany tego ataku:

* Persistent/Stored – polega na dodaniu kodu napisanego w JavaScript w formularzu. Odpowiednio wygenerowany kod pozwala podpiąć się pod sesję użytkownika, który zobaczył wstrzyknięty kod.
* Reflected XSS – atak polega na kliknięciu w link – zazwyczaj wygenerowany przez atakującego – wysłany przez e-mail lub poprzez portal społecznościowy.   
  Wejście do niezaufanej witryny automatycznie wprowadza wirusa do naszego systemu, który ma wiele możliwości które zostały opisane wyżej.

Aby zabezpieczyć się przed tego typu złośliwą ingerencją można skorzystać z zamiany znaczników na encje. Ochronę ciasteczek zapewnia HttpOnly – który umożliwia dostęp jedynie przez protokół HTTP.

Dodatkowo można umieścić w aplikacji bibliotekę JavaScript :

<script src="~/lib/jquery/jquery.js"></script>

<script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>

Przykład 3.6 Biblioteka JavaScript zabezpieczająca przed wstrzykiwaniem kodu

Firma Microsoft nauczona poprzednimi doświadczeniami oraz w celu zwiększenia bezpieczeństwa stworzyła rozwiązania, które pozwalają ustrzec się przed atakami. Platforma .NET udostępnia gotowe rozwiązania, które zabezpieczają przed XSS pod nazwą - Request validation. Jeśli zostanie wykryty niebezpieczny ruch – witryna jest automatycznie wstrzymywana z komunikatem – „wykryto potencjalnie niebezpieczne wartości”. W wersji 4.5 wprowadzono bibliotekę AntiXss która zabezpiecza przed wstrzykiwaniem SQL oraz wspominanym wyżej atakiem XXS.

**SQL Incjection** – jest zabiegiem, w którym haker wstrzykuję zapytanie napisane w języku SQL. Skutkuję to nieautoryzowanym dostępem do bazy danych, unika uwierzytelnienia, wykonuję wstrzykiwany kod(może pobrać dane do logowania użytkowników dostępnych w systemie). Najłatwiejszym zabezpieczeniem przed tego typu atakiem jest wprowadzenie walidacji, która uniemożliwi dodanie nieodpowiednich danych.

Bardzo podobnym atakiem podobnie działającym jest **XPATH injection** – w którym składnia jest napisana w XPATH, jednak wywoływane przez hakera komendy są analogicznymi do SQL Incjection. Uzyskujemy tu dostęp do bazy danych zapisanej w formacie .XML. Inną odmianą tego typu ataku jest **LDAP injection** – który polega na wstrzyknięciu jak największej ilości znaków do formularza.

## Podsumowanie

Ataki na witryny internetowe towarzyszą nam nie od dziś. Najlepszym sposobem zapobiegania przed takiego typu ingerencją jest dobrze zabezpieczona i zaprojektowana aplikacja po stronie serwera jak i klienta. Programista tworzący aplikację powinien mieć na uwadze wszelkie możliwości jakie mógłby wykonać potencjalny haker. Dokumentacja oraz własne doświadczenia są najlepszym przyjacielem każdego szanującego się webmastera. Warto skorzystać z gotowych bibliotek oraz samodzielnie starać się na etapie tworzenia wychwycić potencjalnie zagrożenia i przeciwdziałać im.

Rozdział IV

# Projekt aplikacji i testowanie

## Opis aplikacji

Aplikacja została napisana w języku C# przy użyciu technologii ASP.NET MVC 5. Użyto programu Visual Studio 2015 oraz .NET w wersji 4.5.Aplikacja Schronisko jest zbiorem informacji o schroniskach i psach w nich dostępnych.

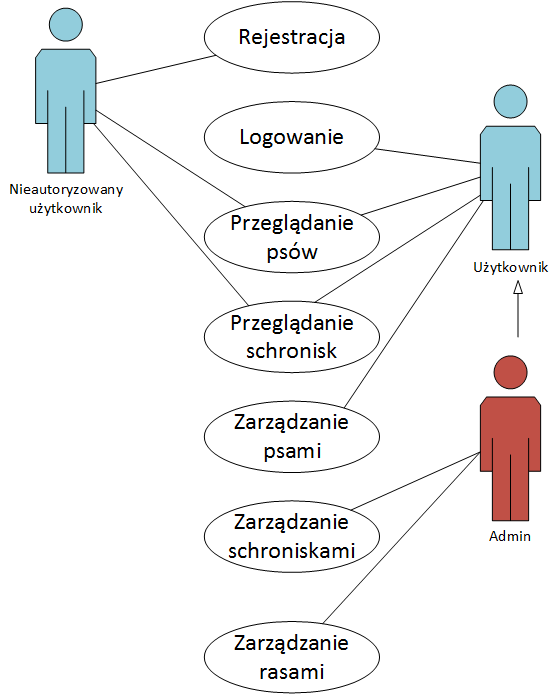
Zostały wyodrębnione następujące przypadki użycia:

* Logowanie do systemu – umożliwia zalogowanie się do aplikacji przy użyciu loginu będącego równocześnie adresem email i hasła. Po tym zabiegu jest możliwość wylogowania się.
* Przeglądanie psów i schronisk – pozwala każdemu użytkownikowi(w tym niezarejestrowanemu) na przeglądanie katalogu psów i schronisk zapisanych w bazie danych.
* Zarządzanie psami – użytkownik po zalogowaniu, może dodać, edytować i usunąć dany rekord z bazy danych.
* Zarządzanie schroniskami – użytkownik posiadający rolę administratora – ma szansę edytować, usuwać oraz dodawać dane do bazy schronisk.
* Rejestracja w systemie – jest możliwa dla każdego użytkownika. Polega na wprowadzeniu danych takich jak adres e-mail oraz hasło.
* Zarządzanie rasami – administrator może dodawać nowe rasy, usuwać je oraz edytować.

Wyróżniamy 3 typy użytkowników.

* Użytkownik nieautoryzowany – może przeglądać schroniska oraz psy. Nie może edytować, usuwać ani dodawać danych. Istnieję możliwość rejestracji i utworzenia konta.
* Użytkownik zarejestrowany – tak jak anonimowy swobodnie przegląda bazę danych, dodatkowo może się zalogować do systemu i uzupełnić bazę danych o nowe psy.
* Administrator – uzupełnia bazę danych o nowe schroniska. Jest ulepszoną wersją użytkownika zarejestrowanego.

Rolę użytkownika przyjmuję pracownik schroniska informujący o nowych psach do adopcji.



Rys. 4.1 Model przypadków użycia na podstawie 3 typów użytkowników

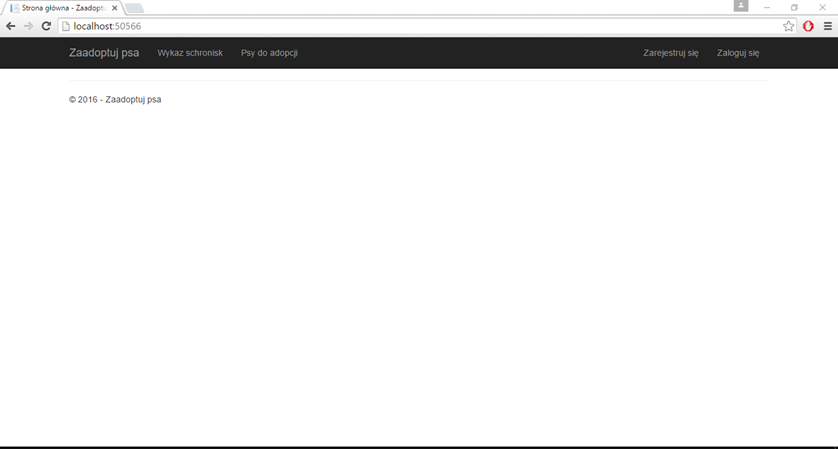
## Testowanie

Przetestujmy naszą aplikację, przy użyciu narzędzia uwierzytelnienia, a następnie sprawdźmy, czy filtry autoryzacji działają poprawnie. W poniższej tabeli przedstawiono rolę oraz dane do zalogowania się dla przykładowych użytkowników.

|  |  |  |
| --- | --- | --- |
| login | hasło | Typ użytkownika |
| pracownik@schronisko.com | Passw123@wp | Manage(zwykły użytkownik) |
| admin@admin.com | Password@12 | Administrator |

Tabela 4.1 Przykładowi użytkownicy

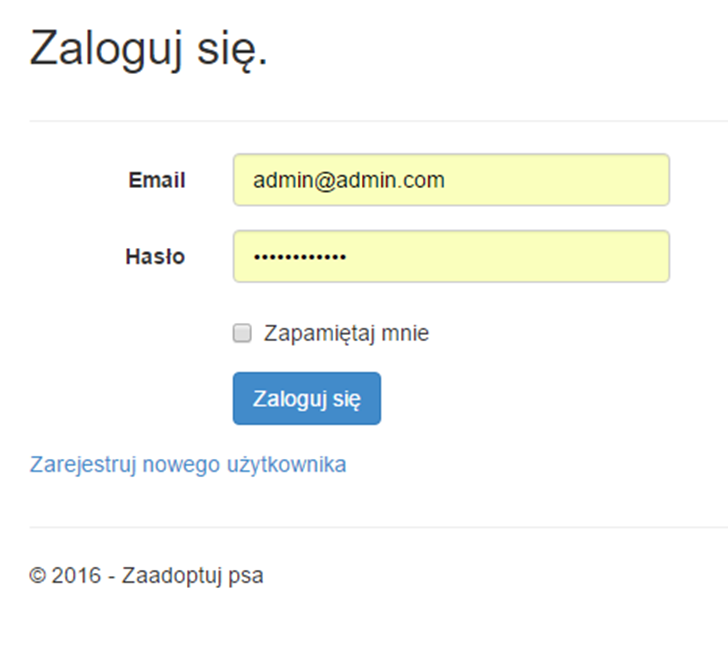
Aplikacja po uruchomieniu, dostępna dla nieautoryzowanego użytkownika prezentuję się w sposób przedstawiony na rysunku



.

Rys. 4.2 Wygląd aplikacji dla nieautoryzowanego użytkownika

Mając przykładowych użytkowników, spróbujmy się zalogować klikając w zakładkę Zaloguj się. Wygląd ekranu logowania przedstawiono na rysunku 4.3



Rys. 4.3 Ekran logowania

Po zalogowaniu mamy dostęp do listy rozwijalnej, która umożliwia nam dodawanie nowych rekordów do bazy danych. W tym przypadku jesteśmy zalogowani jako Adminstrator co daję nam nieograniczone możliwości.

Na Rysunku 4.4 przedstawiono aplikację po zalogowaniu się.

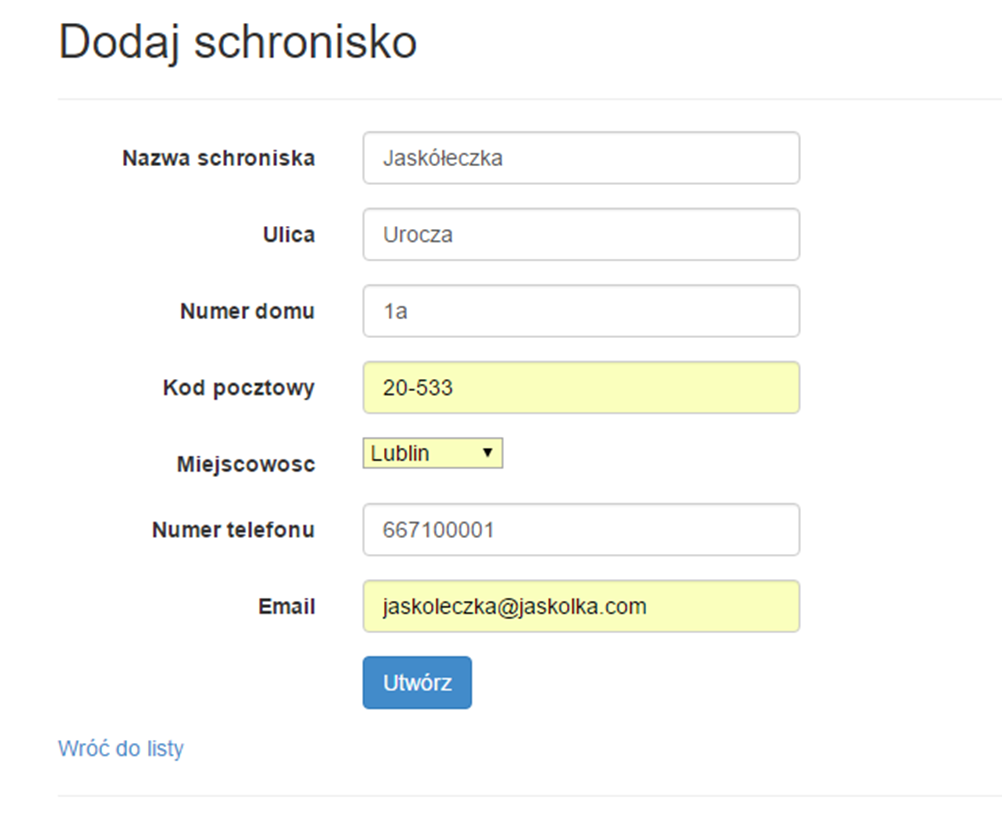


Rys. 4.4 Wygląd aplikacji po zalogowaniu się jako administrator

W związku z naszymi przywilejamy spróbujmy dodać przykładowe rekordy, takie jak:

* Schronisko – Jaskółeczka
* Rasa – Buldog
* Pies – Reksio

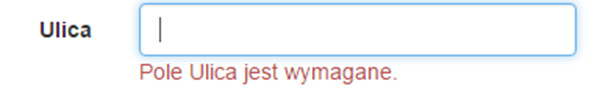
Na samym początku spróbujmy uzupełnić naszą bazę danych o nowe Schroniska, podając przykładowe dane pokazane na rysunku 4.5. Aby uzupełnić bazę schronisk wybieramy z listy rozwijalnej opcje Dodaj schronisko, a następnie wypełniamy wymagane pola w sposób następujący:



Rys. 4.5 Dodanie schroniska do bazy danych

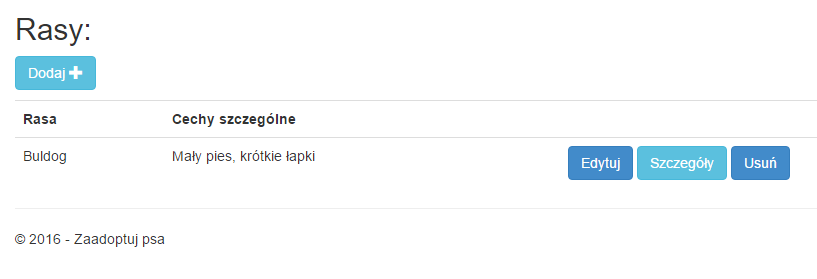
Mechanizm dodawania schroniska, a także innych rekordów do bazy danych jest zabezpieczony przed niebezpieczną ingerencją osób trzecich.

Dodatkowo przy użyciu odpowiednich mechanizmów dostarczonych przez platformę .NET – programista określił dane wymagane i odpowiedni komunikat uniemożliwiający przejście całego formularza bez ich spełnienia. Na rysunku przedstawiono komunikat informujący o konieczności wypełnienia pola.



Rys. 4.6 Sprawdzenie, czy pole Ulica jest wymagane

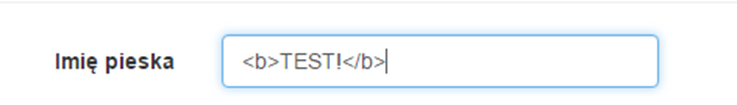
Dodajmy do naszej bazy danych rasę do której możemy później przypisać psa. Cała procedura wygląda identycznie jak w przypadku Schronisk, na poniższym rysunku została przedstawiona przykładowa Rasa, taka jak Buldog.



Rys. 4.7 Przykładowa rasa dodana do bazy danych.

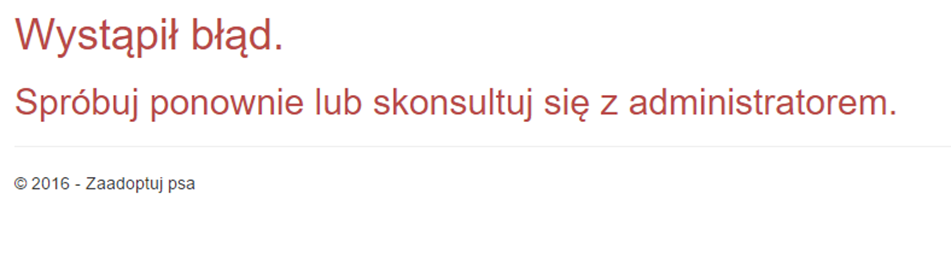
Nasza aplikacja ma pomóc znaleźć wymarzonego pupila, zależy nam na poinformowaniu zainteresowanych przyszłych właścicieli o możliwości przygarnięcia czworonoga. Dodajmy psa o imieniu Reksio, aby dzięki naszej aplikacji odnalazł kochający dom dla siebie. Wypełnijmy formularz dowolnymi danymi i przypiszmy naszego psa do rasy i schroniska

W przypadku, gdyby osoba niepowoływana chciała w dowolny sposób ingerować w strukturę aplikacji jest ona odpowiednio zabezpieczona. Spróbujmy to zweryfikować i wpiszmy do formularza znacznik HTML.



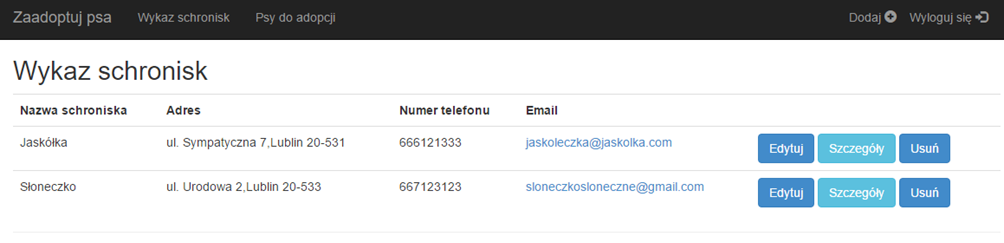
Rys. 4.8 Wstrzyknięcie kodu HTML do formularza

Nasza próba zostanie odrzucona, a następnie przeniesiemy się do podstrony informującej nas o wprowadzonym błędzie(rysunek 4.9). Baza danych nie zostanie zmieniona.



Rys. 4.9 Komunikat o błędzie

Dodajmy inne rekordy do bazy danych, aby rozwinąć możliwości aplikacji. Dodając w sposób analogiczny kolejne schronisko. Aplikacja z uzupełnionymi danymi prezentuję się w sposób następujący:



Rys. 4.10 Przykładowe schroniska

## Podsumowanie

Aplikacja Schronisko jest wyjątkowo użyteczną aplikacją webową oferująca użytkownikom możliwość dodania psów szukających domu. Po zalogowaniu możemy wybrać schronisko mające w opiece danego psa i wyświetlić jego dane aby w szybszy i łatwiejszy sposób odnaleźć wymarzonego czworonoga.

# Bibliografia

1. Z. Fryźlewicz, E. Bukowska, D. Nikończuk - *ASP.NET MVC 4. Programowanie aplikacji webowych,* Wydawnictwo Helion.
2. D. Borycki, M. Pakulski, M. Grabek, J. Matulewski - *ASP.NET MVC. Kompletny przewodnik dla programistów interaktywnych aplikacji internetowych w Visual Studio*,  Wydawnictwo Helion.
3. J. Chadwick, T. Snyder, H. Panda *Programowanie - ASP.NET MVC 4,* Wydawnictwo Helion.
4. E. Gamma , R. Helm, R. Johnson , J. Vlissides - *Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku,* Wydawnictwo Helion.
5. Jamie Munro - *ASP.NET MVC 5 with Bootstrap and Knockout.js* , WydawnictwoO’Reilly Media.
6. T.Wright - *Learning JavaScript. A Hands-On Guide to the Fundamentals of Modern JavaScript.,* Wydawnictwo Addison-Wesley.
7. J. Galloway, B.Wilson, K.Scott Allen, D. Matson - *Professional ASP.NET MVC*5,Wydawnictwo Wrox.
8. Mark J.Collns - *Pro HTML5 with Visual Studio 2015 ,* Wydawnictwo Apress.
9. B. Joshi- *HTML5 Programming for ASP.NET Developers,* Wydawnictwo Apress.
10. M. Wrzesień – *Aplikacje internetowe w ASP.NET* , Rzeszów : Wyższa Szkoła Informatyki i Zarządzania,
11. P. Chłosta- *Ćwiczenia z ASP.NET i kolekcje C#*, Wydawnictwo mikom.
12. M. MacDonald - *Beginning ASP.NET 4.5 in C#*, Wydawnictwo Apress.
13. Dokumentacja Microsoft: https://msdn.microsoft.com/en-us/library/
14. Tutorial na temat MVC : http://www.tutorialsteacher.com/mvc/asp.net-mvc-tutorials
15. Blog o bezpieczeństwie: http://sekurak.pl/
16. Wikipedia: https://pl.wikipedia.org/wiki/Wikipedia:Strona\_g%C5%82%C3%B3wna
17. Łukasz Kosiorowski – blog: http://lukaszkosiorowski.pl/category/programowanie/net/