ASP.NET MVC

ZOSTAŃ PROGRAMISTĄ .NET

.NET

ZOSTAŃ
PROGRAMISTĄ
.NET

ASP.NET MVC jest to platforma do budowy aplikacji Web.

Dzięki zastosowaniu wzorca **MVC** (Model View Controller) możemy odseparować od siebie 3 warstwy model, widok i kontroler. MVC jest to wzorzec polecany nie tylko do pisania aplikacji webowych w C#, ale również innych językach. Model jest to nasza logika aplikacji, widok napisany w razor'ze reprezentuje interfejs użytkownika, a kontroler przyjmuje żądania od użytkownika, obsługuje je za pomocą modelu i zwraca najczęściej odpowiedni widok. Miedzy innymi dzięki MVC kod może tworzyć jednocześnie wielu programistów, którzy mogą mieć różne umiejętności. Za tworzenie widoków, czyli za tak zwany frontend aplikacji może być odpowiedzialna 1 osoba, która nie zna C#. Z kolei za backend może być odpowiedzialna osoba, która nie ma wiedzy frontendowej. Dzięki MVC nasza aplikacja ma dobrą strukturę, wszystkie warstwy mogą być odseparowane, pogrupowane, dzięki temu aplikacja staje się łatwo rozszerzalna, oraz umożliwia między innymi pisanie testów jednostkowych.

Abyś mógł pisać aplikacje webowe w ASP.NET MVC, to znaczy kompletne aplikacje, czyli być fullstack developerem, musisz znać przede wszystkim C#. Następnie jeżeli chcesz pisać aplikacje z ładnym frontendem, to musisz znać podstawy HTMLa, CSSa oraz bootstrapa, który Ci sporo ułatwi. Jeżeli chcesz tworzyć bardziej zaawansowane mechanizmy to również musisz znać JavaScript oraz jQuery. Musisz wiedzieć jak stosować wspomniany wcześniej wzorzec MVC, a także silnik Razor.

HTML to znaczy HyperText Markup Language, czyli hipertekstowy język znaczników jest to kod używany do tworzenia struktury i zawartości strony internetowej. Budując taką stronę w HTMLu korzystamy z różnych znaczników i atrybutów, dzięki którym budujemy całą strukturę strony internetowej.

CSS to znaczy Cascading Style Sheet, czyli kaskadowe arkusze stylów. Dzięki nim możemy w łatwiejszy sposób określać wygląd naszej strony w HTMLu.

JS to znaczy **JavaScript** jest skryptowym językiem programowania. Jest to obecnie najbardziej popularny język programowania na świecie, ale głównie z tego względu, że musi on być zawsze używany we wszystkich aplikacjach webowych. Dzięki JavaScript możemy budowac interaktywne strony internetowe. Możemy w nim pisać całe aplikacje webowe. Pamiętaj, że Java oraz JavaScript to 2 zupełnie nie powiązane ze sobą języki. JavaScript pozwala nam na bezpośrednią interakcję z DOM, czyli Document Object Model, w celu dodawania, zmieniania czy usuwania treści. DOM jest to odzwierciedlenie naszej strony. Przeglądarka analizując HTML i CSS konwertuje go właśnie na DOM.

Korzystanie z JavaScriptu może nam ułatwić biblioteka **jQuery**. Dzięki niej możemy jeszcze łatwiej zmieniać elementy i zdarzenia w dokumencie HTML. Biblioteka jQuery nie dość, że ułatwia i przyśpiesza kodowanie, to jeszcze zwiększa kompatybilność kodu pod różne przeglądarki. jQuery posiada gotowe funkcje, które możemy używać.

Bootstrap jest to darmowa biblioteka CSS, dzięki której możemy w łatwy sposób używać przygotowanych rowiązań CSS. Jeżeli chcesz poznać wszystkie możliwości, funkcjonalności Bootstapa, to odsyłam Cię do ich dokumentacji, w której znajdziesz opisany zestawy różnych klas, które możesz używać w swoim projekcie. Aby użyć jakiegoś gotowego stylu z biblioteki Bootstrap wystarczy tylko dodać w kodzie HTML odpowiednią nazwę klasy lub klas. Bootstrap bardzo nam ułatwia tworzyć aplikacje responsywne, czyli taki które będą się ładnie wyświetlać na różnych urządzeniach (desktopowych/mobilnych).

Uwierzytelnianie, na które programiści często błędnie mówią autentykacja od słowa authentication, jest to proces polegający na potwierdzeniu swojej tożsamości, to znaczy najczęściej poprzez podanie swojego loginu, lub maili oraz hasła. Jest to prostu logowanie do aplikacji, dzięki czemu jesteśmy wstanie zidentyfikować dany podmiot, danego użytkownika.

Autoryzacja jest to proces, w którym potwierdza się czy dany podmiot, użytkownik ma dostęp do danego zasobu. Czyli najpierw użytkownik musi zostać uwierzytelniony, zalogowany do aplikacji i następnie za pomocą autoryzacji, sprawdzamy czy ma on dostęp do konkretnych zasobów.

W C# dzięki przestrzeni nazw **DataAnnotation**, możesz używać atrybutów dla definiowania metadanych kontrolek.

Protokół **HTTP** zawiera mnóstwo typów metod, takich jak GET, POST, PUT, HEAD, DELETE, PATCH, OPTIONS. Najczęście używane to przede wszystkim metody GET oraz POST. Domyślnie wszystkie akcje kontrolerów są metodami GET. Jeżeli chcemy określić daną akcję jako POST to musimy dodać właśnie taki atrybut nad jej nazwą. Metody **GET** zazwyczaj używamy gdy chcemy otrzymać jakieś dane z określonego zasobu i wtedy zazwyczaj, jeżeli mamy jakieś dodatkowe parametry to przekazujemy je w adresie utl po znaku ? i oddzielone &.
Przykładowy URL:
https://www.modestprogrammer.pl/Home/Search?s=szkolenie&test=csharp

Metode **POST** zazwyczaj używana jest wtedy gdy chcemy wysłać jakieś dane na serwer w bardzie bezpieczny sposób by utworzyć lub zaktualizować jakiś zasób. W metodzie POST dane przesyłamy w ciele żądania HTTP, czyli nie są one przesyłane jawnie i widoczne w adresie URL, tak jak ma to miejsce w metodzie GET.

Razor jest to silnik renderujący HTML, dzięki któremu możemy pisać widoki w ASP.NET MVC. Łączy on se sobą HTML i C#. Widoki w aplikacji ASP.NET MVC są pisanie właśnie w Razorze i mają rozszerzenie cshtml. Składnie Razora jest bardzo intuicyjna, piszesz kod w HTML, a jeżeli chcesz przełączyć się na język C# to musisz użyć znaku @.

Kontrolery zazwyczaj powinny być krótkie, nie powinny one zawierać zbyt dużo logiki. Powinny one pobrać dane od użytkownika, wykonać walidacje, wywołać logikę i zwrócić najczęściej widok. Oprócz widoku mogą zwrócić wszystkie obiekty, klas dziedziczących po ActionResult. Międz innymi PartialView, ContentResult, FileResult, JsonResult, RedirectToActionResult itp.

Dane z kontrolera do widoku możesz przekazać poprzez model, ale także jeżeli masz do przekazania więcej danych, to wtedy powinno się używać view modeli. Możesz również przekazać dane za pomocą ViewBag, ViewData oraz TempData.

AJAX czyli Asynchronous JavaScript And Xml, to znaczy asynchroniczny JavaScript i Xml. Dzięki AJAXowi użytkownik aplikacji może wysłać asynchroniczne żądanie do serwera bez przeładowywania całego dokumentu. To znaczy możesz wywołać daną akcję w kontrolerze bez przeładowywania całej strony, bez odświeżania tej strony.

Niezwykle ważnym elementem każdej aplikacji jest **walidowanie** wprowadzanych przez użytkownika danych. W ASP.NET MVC możesz to robić po stronie serwera w C#, a także po stronie klienta za pomocą JavaScript. Błędy walidacyjne możesz zweryfikować po stronie serwera dzięki ModelState.IsValid. Możesz wprowadzać różne warunki walidacji np. poprzez zastosowanie atrybutów z DataAnnotation. Możesz również wprowadzać własne warunki.

Błędy walidacji możesz wyświetlić w widoku dzięki helperowi Html.ValidationMessageFor(x => x.Title) oraz Html.ValidationSummary().
Aby włączyć walidacje po stronie klienta musisz ustawić odpowiednie wartości w web.config:
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
a także dodać do widoku skrypty:
jquery.validate.js
jquery.validate.unobtrusive.js

Do generowania plików PDF na podstawie widoku HTML możesz użyć biblioteki **Rotativa**.

Aby Twoja aplikacji była **bezpieczna** to przede wszystkim zabezpieczyć się dzięki uwierzytelnianiu i autoryzacji. Musisz pamiętać, żeby aktualizować dane na serwerze poprzez metodę POST. Koniecznie walidować dane wprowadzane przez użytkownika. Zabezpieczyć się przed atakim XSS, czyli Cross Site Scripting oraz CSRF (XSRF), czyli Cross Site Request Forgery.

Aby zabezpieczyć się przed atakim CSRF, użyj po stronie widoku helpera Html.AntiForgeryToken(), a po stronie kontrolera atrybuty ValidateAntiForgeryToken.

Pamiętaj, że wartości pól **statycznych** są udostępniany pomiędzy wszystkimi użytkownikami aplikacji.

Sesja (Session) jest zapisywana po stronie serwera i może przechowywać informacje per użytkownik, a nawet per przeglądarka. Przykładem sesji może być koszyk z zakupami w sklepie internetowym, który jest zapamiętywany na serwerze dla konkretnego użytkownika przez pewien czas, dopóki sesja nie wygaśnie. Możemy sami ten czas dla sesji zdefiniować.

```
Session["nr"] = i;
return (int)Session["nr"];
```

Ciasteczka (Cookies) są przechowywane po stronie przeglądarki, mogą być zapisywane na określony czas i mieć maksymalnie do 4KB. Wartości możesz podejrzeć w przeglądarce w narzędziach developerskich. Jakieś małe informacje np. zapamiętywanie jakichś wcześniejszych wyborów użytkownika.

```
var cookie = new HttpCookie("nr",
i.ToString());
cookie.Expires = DateTime.Now.AddDays(365);
Response.SetCookie(cookie);
return
int.Parse(Request.Cookies["nr"].Value);
```

Cache jest udostępniany pomiędzy użytkownikami aplikacji. Przechowywanie danych po stronie serwera w celu odciążenia serwera. Możemy na przykład pobrać jakieś dane z bazy danych, które są rzadko aktualizowane, zapisać je w cache i następnie wczytać z cache'a. Dzięki temu odciążymy serwer.

```
HttpRuntime.Cache["nr"] = i;
return (int)HttpRuntime.Cache["nr"];
```

Możesz również oznaczyć całą akcję w kontrolerze atrybutem OutputCache i ustawić przez jaki czas ten dane będą pobierane z cache'a.

```
[OutputCache(Duration = 10)]
public ActionResult Index()
{
      //pobieranie jakichś danych
       return View();
}
```

IIS, czyli Internet Information Services, jest to zbiór usług internetowych, które pełnią funkcję serwera dla aplikacji webowych pisanych w ASP.NET. Jeżeli chcesz hostować swoją aplikację webową, to możesz właśnie to zrobi dzięki IISowi. Możesz to zrobić lokalnie na swoim komputerze lub na serwerze.

Przy wdrażaniu aplikacji powinieneś ustawić osobny widok dla wyświetlania błędów użytkownikom, tak żeby nie widzieli całej ścieżki wystąpionych błedów.

```
<customErrors mode="On"
defaultRedirect="~/Shared/Error">
</customErrors>
```

Pamiętaj także, o ustawieniu copilation debug na false:

```
<compilation
debug="false" targetFramework="4.7.2" />
```

Możesz publikować Twoje aplikacjie na zewnętrznym hostingu np. Webio.pl. Co ważne musi to być hosting na systemie windows.

Akcje są to metody klas kontrolerów, który zazwyczaj zwracają widok.

Routing jest mechanizm, który dopasowuje żądanie, a konkretnie adres URL do odpowiedniej akcji kontrolera.