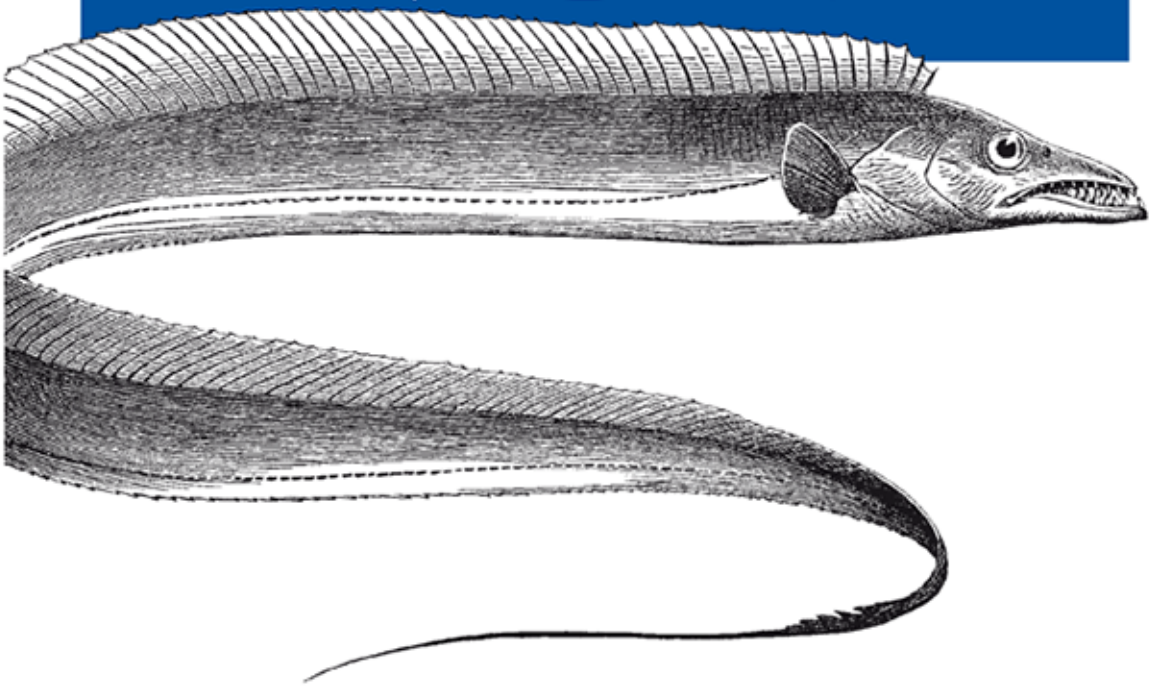
Wykorzystaj potencjał ASP.NET!

Programowanie

ASP.NET MVC 4



HELION

O'REILLY®

Jess Chadwick, Todd Snyder, Hrusikesh Panda

Tytuł oryginału: ASP.NET MVC 4. Programowanie

Tłumaczenie: Robert Górczyński

ISBN: 978-83-246-6644-7

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/aspm4p
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 10.

Programowanie na platformy mobilne

Sieć mobilna to oferujące bardzo duże możliwości medium dostarczania treści większej grupie użytkowników. Wraz ze zwiększającą się liczbą używanych smartfonów i rozwojem sieci mobilnej uwzględnienie urządzeń mobilnych podczas przygotowywania projektów staje się coraz ważniejsze.

Największy problem podczas przygotowywania aplikacji do jej używania w sieci mobilnej polega na tym, że nie wszystkie urządzenia mobilne są tworzone w taki sam sposób. Poszczególne urządzenia mają różne możliwości sprzętowe i zainstalowane przeglądarki internetowe, a także odmienne funkcje, np. w zakresie dotykowej obsługi urządzenia itd. Przystosowanie witryny internetowej do prawidłowego i spójnego działania w różnych urządzeniach mobilnych nie jest łatwym zadaniem.

W tym rozdziale dowiesz się, jak używać funkcji oferowanych przez platformę ASP.NET MVC — w szczególności nowych funkcji, które pojawiły się w wersji 4. platformy — w celu zapewnienia prawidłowego i spójnego działania witryny internetowej na maksymalnej liczbie urządzeń mobilnych. Przekonasz się również, co zrobić w sytuacji, gdy dla danego urządzenia mobilnego nie można zaoferować prawidłowej obsługi witryny internetowej, którą tworzysz.

Funkcje mobilne platformy ASP.NET MVC 4

Począwszy od wersji 3., platforma ASP.NET MVC oferuje zestaw funkcji ułatwiających tworzenie mobilnych wersji witryn internetowych. Funkcje te zostały usprawnione w wersji 4. platformy.

Poniższa lista przedstawia krótki opis funkcji pomagających w tworzeniu wersji mobilnej aplikacji sieciowej, wprowadzonych w ASP.NET MVC 4. W pozostałej części rozdziału przekonasz się, jak wykorzystać te funkcje w tworzonej aplikacji sieciowej.

Szablon aplikacji mobilnej w ASP.NET MVC 4
: Jeżeli już od samego początku chcesz utworzyć jedynie mobilną wersję aplikacji sieciowej, platforma ASP.NET MVC 4 oferuje szablon *Mobile Application*, który pozwala na natychmiastowe przystąpienie do tworzenia wersji mobilnej aplikacji sieciowej. Podobnie jak w przypadku szablonów zwykłych aplikacji MVC, w szablonie mobilnym platforma umieszcza już pewien kod odpowiedzialny za wygenerowanie widoków dla urządzeń mobilnych, przeprowadza konfigurację pakietów jQuery Mobile MVC NuGet oraz tworzy szkielet aplikacji. Dokładne omówienie szablonu aplikacji mobilnej znajdziesz w dalszej części rozdziału, w podrozdziale zatytułowanym „Szablon aplikacji mobilnej w ASP.NET MVC 4”.

Tryby wyświetlania

Aby można było jeszcze łatwiej dostosować aplikację sieciową do różnych urządzeń, platforma ASP.NET MVC 4 oferuje tak zwane *tryby wyświetlania* — to funkcja odpowiedzialna za wykrycie i przygotowanie odpowiedniego widoku dla poszczególnych urządzeń.

Poszczególne urządzenia mobilne mają różne rozdzielczości ekranu, inne zachowanie przeglądarek internetowych, a nawet odmienne funkcje, które mogą być wykorzystane przez aplikacje sieciowe. Zamiast próbować dopasować dany widok do maksymalnej liczby różnych urządzeń, lepszym rozwiązaniem jest umieszczenie poszczególnych zachowań i funkcji w oddzielnych widokach przeznaczonych dla konkretnych urządzeń.

Załóżmy, że masz przygotowany widok o nazwie *Index.cshtml*, przeznaczony dla zwykłych urządzeń biurkowych. Otrzymujesz zadanie przygotowania mobilnej wersji tego widoku przeznaczonej do wyświetlania na smartfonach i tabletach. Dzięki użyciu trybów wyświetlania możesz przygotować odpowiednie wersje widoku dla różnych urządzeń, np. *Index.iPhone.cshtml* i *Index.iPad.cshtml*, a następnie podczas uruchamiania aplikacji zarejestrować je za pomocą dostawcy widoku na platformie ASP.NET MVC 4. Opierając się na kryteriach filtrowania, platforma ASP.NET MVC 4 może automatycznie wyszukać widok zawierający odpowiedni przyrostek (*iPhone* lub *iPad*) i wyświetlić ten widok zamiast standardowego, przeznaczonego dla urządzeń biurkowych. (Zwróć uwagę, jak dla widoków alternatywnych platforma ASP.NET MVC stosuje prostą konwencję nadawania nazw plikom w postaci *nazwa_widoku.urządzenie.rozszerzenie*). W znajdującym się w dalszej części rozdziału podrozdziale zatytułowanym „Widoki dla konkretnych przeglądarek internetowych" przekonasz się, jak użyć tej funkcji w celu zapewnienia obsługi różnych urządzeń.

Nadpisanie zwykłych widoków ich wersjami mobilnymi

Platforma ASP.NET MVC 4 wprowadziła prosty mechanizm pozwalający na nadpisanie dowolnego widoku (włączając w to układy graficzne i widoki częściowe) dla poszczególnych przeglądarek internetowych, w tym także dla przeglądarek internetowych używanych w urządzeniach mobilnych. Aby dostarczyć widok przeznaczony dla urządzenia mobilnego, konieczne jest utworzenie pliku widoku zawierającego w nazwie człon *.Mobile*. Na przykład przygotowanie mobilnej wersji widoku *Index* wymaga utworzenia jego kopii wraz z wymienionym wcześniej członem w nazwie (*Views\Home\Index.Mobile.cshtml*); platforma ASP.NET automatycznie wygeneruje ten (zamiast biurkowego) widok w mobilnej wersji przeglądarki internetowej. Warto dodać, że wprawdzie funkcja trybów wyświetlania pozwala na wskazanie widoku dla konkretnej przeglądarki internetowej w urządzeniu mobilnym, ale odbywa się to na ogólniejszym poziomie. Takie rozwiązanie będzie użyteczne, jeśli widoki są wystarczająco ogólne, aby mogły być prawidłowo wyświetlane w różnych mobilnych wersjach przeglądarek internetowych. Tę funkcję możesz również stosować, jeśli używasz platformy takiej jak jQuery Mobile, która zapewnia spójne działanie aplikacji na większości platform mobilnych.

Platforma jQuery Mobile

Platforma jQuery Mobile przynosi aplikacjom sieciowym całe bogactwo i użyteczność bibliotek jQuery oraz jQuery UI. Zamiast zmagać się z niespójnościami działania przeglądarek internetowych w różnych urządzeniach mobilnych, możesz po prostu utworzyć jedną aplikację sieciową, która będzie działać we wszystkich urządzeniach mobilnych. Platforma ta oferuje korzyści wynikające z możliwości stosowania technik progresywnych usprawnień i zapewnia elastyczny projekt, dzięki któremu starsze urządzenia nadal

otrzymują funkcjonalną (choć niekoniecznie ładną i bogatą w funkcje) aplikację, podczas gdy nowe urządzenia korzystają z pełni możliwości dostarczanych przez funkcje języka HTML5. Platforma jQuery Mobile doskonale obsługuje motywy, co pozwala na utworzenie niepowtarzalnej witryny bez konieczności rezygnowania z zalet wynikających z progresywnych uaktualnień. W rozdziale tym przekonasz się, jak dzięki platformie jQuery Mobile można przenieść aplikację na kolejny poziom.

Większa przyjazność aplikacji mobilnej

Tematyka związana z programowaniem na platformy mobilne jest obszerna i obejmuje wiele aspektów, które twórca witryny internetowej musi uwzględnić, aby utworzyć tego rodzaju witrynę. Prawdopodobnie najważniejszym aspektem jest najlepszy sposób dostarczenia informacji użytkownikom oraz interakcja z użytkownikami.

Weź pod uwagę aplikację sieciową dla urządzenia biurkowego, w którym okno przeglądarki internetowej jest wyświetlone na dużym ekranie, a użytkownik ma do dyspozycji szybkie i niezawodne połączenie internetowe, natomiast interakcję z aplikacją sieciową prowadzi za pomocą klawiatury i myszy. Z kolei aplikacje mobilne z reguły są wyświetlane na niewielkich ekranach, internet nie zawsze jest szybki i niezawodny, natomiast do wprowadzania danych służy najczęściej piórko bądź kilka palców.

Wymienione ograniczenia niewątpliwie prowadzą do zmniejszenia ilości dostarczanej treści oraz liczby oferowanych funkcji w porównaniu z aplikacjami sieciowymi przeznaczonymi dla przeglądarek internetowych działających w urządzeniach biurkowych. Jednak aplikacje mobilne dostarczają pewnych możliwości, które zazwyczaj nie istnieją w tradycyjnych aplikacjach sieciowych; są to np. obsługa danych dotyczących geograficznego położenia urządzenia mobilnego, większe możliwości komunikacji, obsługa komunikacji audio i wideo itd.

Trafne określenie wymagań użytkowników aplikacji to pierwszy krok podczas przygotowywania strategii tworzenia aplikacji mobilnej. Przeanalizuj przedstawione poniżej przykłady sposobów wykorzystania urządzeń mobilnych.

- Spacer po ulicy i jednoczesne sprawdzanie poczty e-mail (od czasu do czasu podnoszenie przy tym głowy, aby nie uderzyć w latarnię).
- Podróż komunikacją miejską i jednoczesne czytanie najnowszych wiadomości.
- Popijanie kawy w kawiarni — użytkownik jedną ręką trzyma filiżankę z kawą, natomiast drugą urządzenie mobilne, w którym sprawdza wysokość salda na koncie.

Sytuacje te mają jedną wspólną cechę — użytkownik dzieli swoją uwagę, próbując jak najszybciej wykonać pewne zadania i przejść do kolejnych w dniu wypełnionym po brzegi różnymi zajęciami.

Dla twórcy witryny internetowej oznacza to, że powinna ona koncentrować się na dostarczaniu użytkownikowi treści w żądany przez niego sposób, szybko, zrozumiale i odpowiednio do wykonywanego zadania.

Tworzenie widoku mobilnego dla aukcji

Podczas tworzenia mobilnej aplikacji sieciowej pracę rozpoczynasz od dodania do istniejącej aplikacji widoków przeznaczonych dla urządzeń mobilnych lub od utworzenia zupełnie od początku nowej aplikacji mobilnej. Ten wybór zależy od wielu czynników; oba podejścia mają swoje wady i zalety. Jak się przekonasz, platforma ASP.NET MVC oferuje narzędzia pomagające w pracy w obu przypadkach.

Skoncentrujemy się teraz na dodaniu widoku mobilnego do istniejącego widoku dla urządzenia biurkowego, a następnie będziemy powoli usprawniać widok mobilny kolejnymi funkcjami oferowanymi przez platformę ASP.NET MVC 4.

Rozpoczynamy pracę od utworzenia kopii widoku *Auctions.cshtml* i nadajemy jej nazwę *Auctions.Mobile.cshtml*, co wskazuje, że widok jest przeznaczony dla urządzenia mobilnego.

Aby zróżnicować generowane widoki, zmieniamy także nagłówek `<h1>` w widoku mobilnym na *Aukcje mobilne*.

Utworzenie widoku mobilnego można potwierdzić poprzez uruchomienie aplikacji sieciowej i wyświetlenie strony aukcji w przeglądarce internetowej dla urządzenia mobilnego. Wynik tej operacji pokazano na rysunku 10.1.

Rysunek 10.1. Platforma ASP.NET MVC potrafi wykryć i automatycznie wygenerować widok dla urządzenia mobilnego

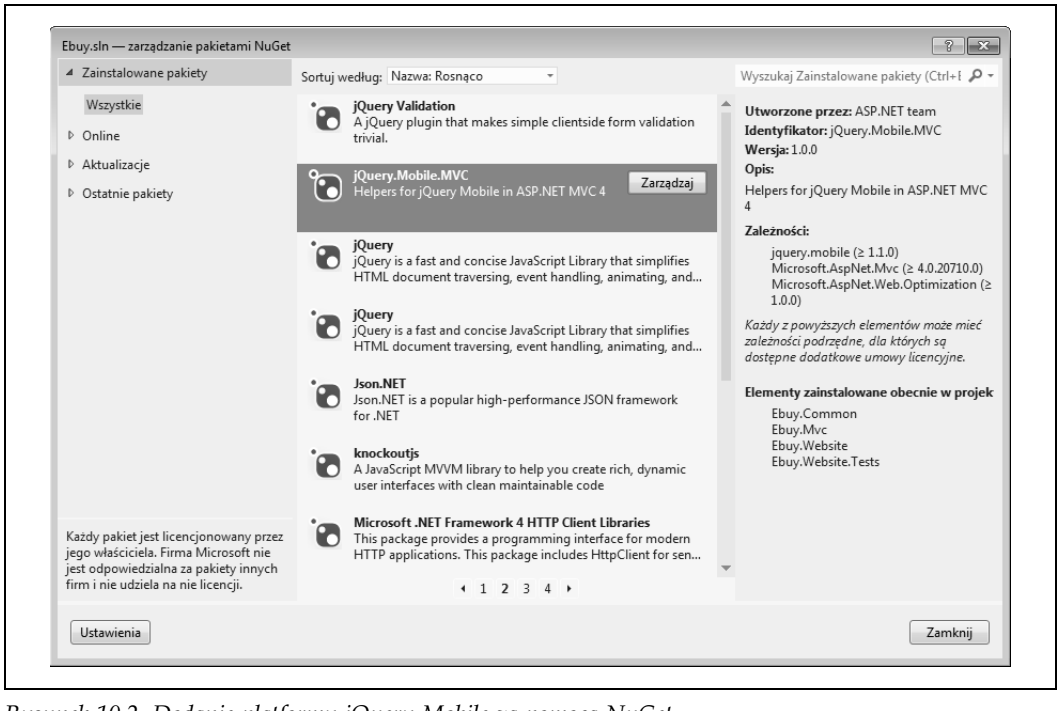
Jak możesz zobaczyć na rysunku, nagłówek *Aukcje mobilne* potwierdza wyświetlenie widoku mobilnego (przejście na stronę aukcji w biurkowej wersji przeglądarki internetowej powoduje wyświetlenie zwykłego widoku wraz z nagłówkiem *Aukcje*). Tryby wyświetlania to funkcja platformy pozwalająca na wykrycie przeglądarki internetowej używanej przez klienta i wygenerowanie dla niej odpowiedniego widoku.

Kiedy żądanie przychodzi z urządzenia mobilnego, platforma ASP.NET MVC 4 nie ogranicza się jedynie do automatycznego wczytania widoku mobilnego. W rzeczywistości wczytuje także mobilne wersje układów graficznych i widoków częściowych — pakiet *jQuery.Mobile.MVC* jest wykorzystywany do utworzenia opartego na jQuery Mobile układu graficznego zoptymalizowanego dla urządzeń mobilnych.

Rozpoczęcie pracy z jQuery Mobile

jQuery Mobile pozwala na szybkie usprawnienie istniejącego widoku i utworzenie na jego podstawie widoku dla urządzeń mobilnych, który będzie miał rodzimy dla nich wygląd i rodzime zachowanie. Możliwe jest także użycie motywu w aplikacji; technika uaktualnień progresywnych powoduje, że starsze wersje przeglądarek internetowych otrzymują nieco zredukowaną (niezbyt atrakcyjną wizualnie), ale funkcjonującą i użyteczną stronę.

Aby móc używać jQuery Mobile, konieczne jest zainstalowanie pakietu *jQuery.Mobile.MVC* z galerii pakietów NuGet (rysunek 10.2).



Rysunek 10.2. Dodanie platformy jQuery Mobile za pomocą NuGet

Pakiet ten powoduje dodanie następujących plików:

jQuery Mobile Framework
: Zestaw plików JavaScript (*jQuery.mobile-1.1.0.js*) i CSS (*jQuery.mobile-1.1.0.css*) wraz z ich zminimalizowanymi wersjami oraz obrazami pomocniczymi.

/Content/Site.Mobile.css
: Nowy arkusz stylów przeznaczony dla urządzeń mobilnych.

Views/Shared/_Layout.Mobile.cshtml
: Układ graficzny zoptymalizowany dla urządzeń mobilnych i odwołujący się do plików platformy jQuery Mobile (JavaScript i CSS). Platforma ASP.NET MVC będzie automatycznie wczytywała ten układ dla widoków mobilnych.

Komponent przełączania widoku
: Składa się z widoku częściowego *Views/Shared/_ViewSwitcher.cshtml* oraz kontrolera *ViewSwitcherController.cs*. Ten komponent powoduje wyświetlenie w mobilnej przeglądarce internetowej łącza pozwalającego użytkownikowi na przejście do strony w wersji dla urządzenia biurkowego. Sposób działania tego komponentu zostanie omówiony w dalszej części tego rozdziału, w podrozdziale „Przełączanie pomiędzy widokami mobilnym i zwykłym".

Platforma jQuery Mobile nadal jest w fazie opracowywania, więc po jej zainstalowaniu może się okazać, że jej pliki są w nowszych wersjach niż wymienione w książce.

Aby umożliwić platformie jQuery Mobile nadanie stronie odpowiedniego stylu, przejdź do pliku *Views/Shared/_Layout.Mobile.cshtml* i zmodyfikuj jego treść w taki sposób, aby odpowiadała poniższemu fragmentowi kodu:

```
<body>
   <div data-role="page" data-theme="b">
      <header data-role="header">
         <h1>@Html.ActionLink("EBuy: Witryna demonstracyjna ASP.NET MVC",
            "Index", "Home")</h1>
      </header>
      <div id="body" data-role="content">
         @RenderBody()
      </div>
   </div>
</body>
```

Następnie zmodyfikuj plik *Auctions.Mobile.cshtml* i zoptymalizuj go dla układu graficznego przeznaczonego dla urządzeń mobilnych:

```
@model IEnumerable<AuctionViewModel>
<link href="@Url.Content("~/Content/product.css")" rel="stylesheet" type="text/css" />
@{
   ViewBag.Title = "Auctions";
}

<header>
   <h3>Aukcje mobilne</h3>
</header>

<ul id="auctions">
```

```
    @foreach (var auction in Model)
    {
        <li>
            @Html.Partial("_AuctionTile", auction);
        </li>
    }
</ul>
```

Po wprowadzeniu wymienionych zmian skompiluj i uruchom aplikację, a następnie wyświetl stronę główną aplikacji w mobilnej wersji przeglądarki internetowej. Zwróć uwagę na zmiany w stosunku do wersji biurkowej. Na ekranie powinieneś zobaczyć stronę podobną do pokazanej na rysunku 10.3.

Rysunek 10.3. Aplikacja EBuy zoptymalizowana dla układu graficznego przeznaczonego dla urządzeń mobilnych

Możesz zobaczyć, jak widok *Auctions* został zmieniony w celu jego dostosowania do przeglądarki internetowej urządzenia mobilnego. Wprawdzie wygląd strony nie jest perfekcyjny, ale pakiet jQuery Mobile dostarcza podstaw, na bazie których możesz szybko i łatwo tworzyć widoki mobilne.

Usprawnianie widoku za pomocą jQuery Mobile

Pakiet *jQuery.Mobile.MVC* wykonuje większą część pracy za programistę, ale interfejs użytkownika witryny internetowej nadal nie przypomina rodzimej aplikacji dla urządzenia mobilnego. Na szczęście platforma jQuery Mobile dostarcza wielu komponentów i stylów, dzięki którym aplikacja otrzyma widok naprawdę przypominający aplikację mobilną.

Usprawnianie listy aukcji za pomocą komponentu listview platformy jQuery Mobile

Pracę rozpoczynamy od usprawnienia listy aukcji poprzez użycie komponentu `listview` platformy jQuery Mobile. W celu przeprowadzenia większości transformacji mobilnych platforma jQuery Mobile operuje na atrybutach `data-role`. Aby zatem wygenerować listę aukcji jako `listview`, koniecznie należy dodać atrybut `data-role="listview"` do znacznika `<ul>` aukcji:

```
<ul id="auctions" data-role="listview">
   @foreach (var auction in Model.Auctions)
   {
      <li>
         @Html.Partial("_AuctionTileMobile", auction);
      </li>
   }
</ul>
```

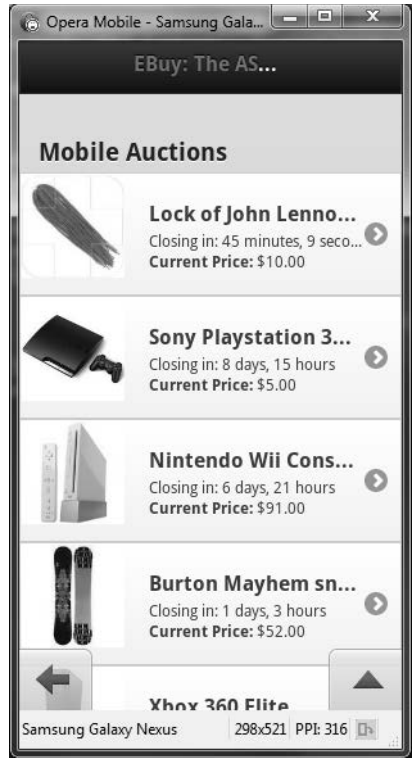
oraz w następujący sposób zmodyfikować widok częściowy *_AuctionTileMobile*:

```
@model AuctionViewModel
@{
   var auctionUrl = Url.Auction(Model);
}

<a href="@auctionUrl">
   @Html.Thumbnail(Model.Image, Model.Title)
   <h3>@Model.Title</h3>
   <p>
      <span>Koniec aukcji za: </span>
      <span class="time-remaining" title="@Model.EndTimeDisplay">↵
      @Model.RemainingTimeDisplay</span>
   </p>
   <p>
      <strong>Aktualna cena: </strong>
      <span class="current-bid-amount">@Model.CurrentPrice</span>
      <span class="current-bidder">@Model.WinningBidUsername</span>
   </p>
</a>
```

Po wyświetleniu widoku *Auctions* w mobilnej wersji przeglądarki internetowej otrzymasz teraz znacznie ładniejszą listę aukcji (rysunek 10.4).

Biorąc pod uwagę to, że element `<ul>` to w rzeczywistości lista, możesz uznać za zbyteczne dodawanie roli `listview`. Element `<ul>` wyświetla listę, ale obszar łącza będzie zbyt mały, aby można było kliknąć go w urządzeniu mobilnym wyposażonym w niewielki ekran. Zadaniem atrybutu `data-role="listview"` jest umożliwienie powiększania obszaru łącza i ułatwienie użytkownikom naciskania elementów listy.



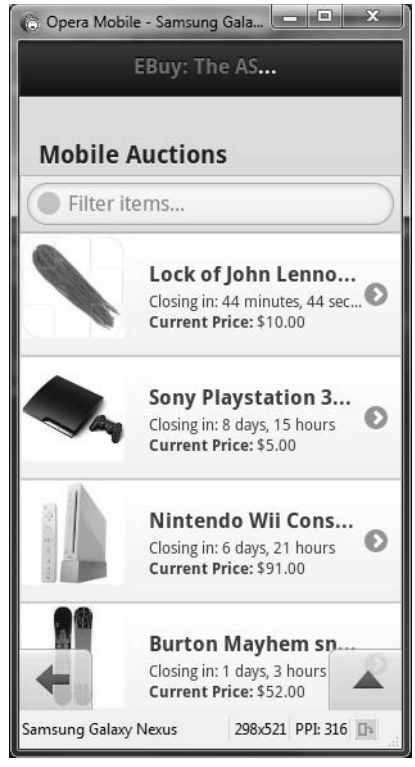
Rysunek 10.4. Zawartość znacznika <ul> wygenerowana przez platformę jQuery Mobile jako listview

Umożliwianie przeszukiwania listy aukcji dzięki komponentowi „data filter" platformy jQuery Mobile

Kolejnym krokiem jest zwiększenie przyjazności widoku poprzez dodanie użytecznego pola wyszukiwania, pozwalającego użytkownikowi na szybkie filtrowanie listy aukcji. Dzięki platformie jQuery Mobile to zadanie jest bardzo łatwe — znacznikowi `<ul>` listy aukcji wystarczy nadać atrybut `data-filter="true"`:

```
<ul id="auctions" data-role="listview" data-filter="true">
   @foreach (var auction in Model.Auctions)
   {
      <li class="listitem">
         @Html.Partial("_AuctionTileMobile", auction);
      </li>
   }
</ul>
```

Po odświeżeniu strony w mobilnej wersji przeglądarki internetowej powinieneś zobaczyć na górze strony pole wyszukiwania (rysunek 10.5).



Rysunek 10.5. Dzięki platformie jQuery Mobile listę aukcji można przeszukiwać

Wprowadź tekst w polu wyszukiwania i zobacz, jak platforma jQuery Mobile automatycznie filtruje listę, wyświetlając jedynie te aukcje, które zostały dopasowane do wprowadzonego przez Ciebie tekstu (rysunek 10.6).

Przekonałeś się, jak platforma jQuery Mobile ułatwia transformację dowolnej strony, aby wyświetlona w urządzeniu docelowym miała dla niego rodzimy wygląd i sposób działania. Oprócz wymienionych funkcji jQuery Mobile oferuje także wiele innych użytecznych komponentów, które możesz wykorzystać w widokach witryny internetowej, czyniąc je tym samym bardziej dostosowanymi dla użytkowników urządzeń mobilnych. Pełną listę odpowiednich atrybutów znajdziesz na stronie dokumentacji API platformy jQuery Mobile pod adresem *http://jquerymobile.com/test/docs/api/data-attributes.html*.

Przełączanie pomiędzy widokami mobilnym i zwykłym

Kiedykolwiek dostarczasz wersję mobilną witryny internetowej, ogólnie rzecz biorąc, dobrym rozwiązaniem jest automatyczne przekierowanie użytkowników urządzeń mobilnych do odpowiedniej wersji witryny. Warto jednak zapewnić im możliwość przejścia do pełnej wersji witryny internetowej przeznaczonej dla urządzeń biurkowych.

Rysunek 10.6. Platforma jQuery Mobile automatycznie filtruje listę aukcji na podstawie tekstu wprowadzonego w polu wyszukiwania

Zwróć uwagę, że na górze widoku mobilnego standardowo dostarczanego przez szablon aplikacji mobilnej ASP.NET MVC znajduje się łącze pozwalające użytkownikom na przejście do „widoku zwykłego". To jest widget `ViewSwitcher` zainstalowany jako część pakietu NuGet *jQuery.Mobile.MVC*.

Aby dowiedzieć się, jaki jest sposób działania tego widgetu, konieczne będzie zagłębienie się w komponenty platformy jQuery Mobile.

Spójrz na nowy widok częściowy o nazwie *_ViewSwitcher.cshtml*, w którym znajdziesz następujący kod znaczników:

```
@if (Request.Browser.IsMobileDevice && Request.HttpMethod == "GET")
{
   <div class="view-switcher ui-bar-a">
      @if (ViewContext.HttpContext.GetOverriddenBrowser().IsMobileDevice)
      {
         @: Wyświetlenie widoku mobilnego.
         @Html.ActionLink("Widok zwykły", "SwitchView", "ViewSwitcher",
         new { mobile = false, returnUrl = Request.Url.PathAndQuery },
         new { rel = "external" })
      }
      else
      {
```

```
            @: Wyświetlenie widoku zwykłego.
            @Html.ActionLink("Widok mobilny", "SwitchView", "ViewSwitcher",
            new { mobile = true, returnUrl = Request.Url.PathAndQuery },
            new { rel = "external" })
        }
    </div>
}
```

Wartością zwrotną metody `GetOverridenBrowser()` jest obiekt `HttpBrowserCapabilities` zawierający możliwości oferowane przez nadpisywaną przeglądarkę internetową lub aktualnie używaną, jeśli nie jest nadpisywana. W ten sposób możesz sprawdzić, czy urządzenie, które wykonało żądanie, jest urządzeniem mobilnym. Następnie widget określa widok, który powinien zostać wygenerowany, oraz tworzy odpowiednie łącza pozwalające na przełączanie pomiędzy widokami mobilnym i zwykłym.

Bonusem jest ustawienie właściwości `mobile` w słowniku `RouteValue`, wskazującej aktualnie aktywny widok.

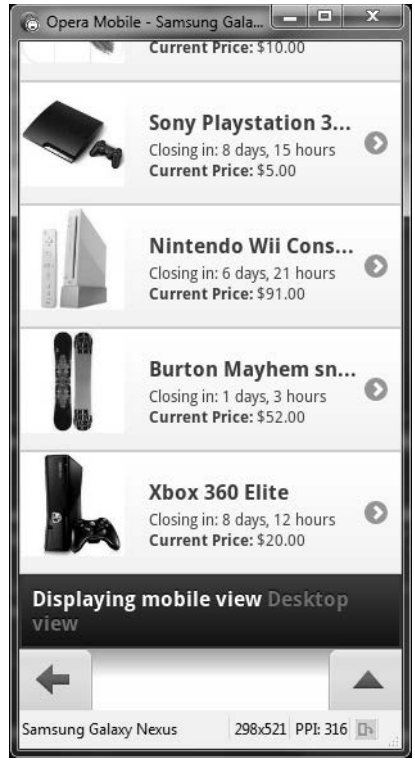
Spójrz teraz na klasę `ViewSwitcherController` zawierającą logikę odpowiedzialną za przełączanie widoków:

```
public class ViewSwitcherController : Controller
{
    public RedirectResult SwitchView(bool mobile, string returnUrl)
    {
        if (Request.Browser.IsMobileDevice == mobile)
        {
            HttpContext.ClearOverriddenBrowser();
        }
        else
        {
            HttpContext.SetOverriddenBrowser(mobile ? BrowserOverride.Mobile
                : BrowserOverride.Desktop);
        }
        return Redirect(returnUrl);
    }
}
```

W zależności od tego, czy żądanie pochodzi z urządzenia mobilnego (na co wskazuje wartość właściwości `Request.Browser.IsMobileDevice`), kontroler używa metod `ClearOverriddenBrowser()` i `SetOverriddenBrowser()` w celu poinformowania platformy ASP.NET MVC o sposobie traktowania żądania. Następnie platforma wyświetla odpowiednią wersję witryny, mobilną dla urządzenia mobilnego lub pełną dla urządzenia biurkowego.

Przedstawiony poniżej fragment kodu umieść przez zamykającym znacznikiem `<body>` w pliku *Layout.Mobile.cshtml* w celu wygenerowania widoku częściowego `ViewSwitcher` jako stopki (rysunek 10.7):

```
<div data-role="footer">
    @Html.Partial("_ViewSwitcher")
</div>
```



Rysunek 10.7. Przełącznik widoku wyświetlony w stopce strony

Po kliknięciu łącza *Widok zwykły* nastąpi wyświetlenie zwykłej wersji widoku *Auctions*. Zwróć jednak uwagę, że widok zwykły nie zawiera łącza pozwalającego na przejście do widoku mobilnego. Aby to naprawić, przejdź do widoku współdzielonego *_Layout.cshtml* i umieść w nim poniższy wiersz kodu:

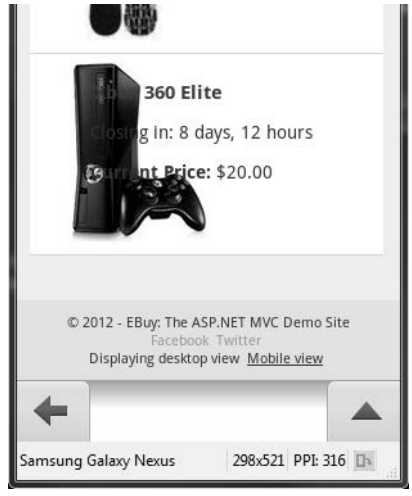
```
@Html.Partial("_ViewSwitcher")
```

Uruchom aplikację; w mobilnej wersji przeglądarki internetowej przejdź do dowolnej strony. Przekonasz się, że przełącznik widoków generuje łącza pozwalające na przełączanie pomiędzy widokami zwykłym i mobilnym (rysunek 10.8).

Unikanie widoków biurkowych w witrynie mobilnej

Na pewno zauważysz, że w przypadku braku mobilnej wersji widoku platforma ASP.NET MVC powoduje wygenerowanie wersji biurkowej widoku w mobilnym układzie graficznym strony.

Trzymanie się standardów podczas tworzenia kodu znaczników pomaga w wyświetlaniu w miarę użytecznego widoku, ale mogą zdarzać się sytuacje, w których po prostu będziesz wolał uniknąć takiego zachowania platformy.



Rysunek 10.8. Przełącznik widoków wyświetlony w widoku biurkowym strony

W tym celu należy przypisać wartość `true` właściwości `RequireConsistentDisplayMode`:

```
@{
   Layout = "~/Views/Shared/_Layout.cshtml";
   DisplayModeProvider.Instance.RequireConsistentDisplayMode  = true;
}
```

Po wprowadzeniu powyższej zmiany żaden widok domyślny (to znaczy niemobilny) nie zostanie wygenerowany w mobilnym układzie graficznym strony. Ustawienie możesz wprowadzić także globalnie dla wszystkich widoków poprzez przypisanie wartości `true` wymienionej właściwości w pliku */Views/_ViewStart.cshtml*.

Usprawnianie wersji mobilnej witryny

Mobilne wersje przeglądarek internetowych mają możliwość wyświetlania stron HTML, przynajmniej do pewnego stopnia. Jednak poleganie w całości na przeglądarce internetowej podczas wyświetlania strony wcale nie gwarantuje uzyskania najlepszych wyników. Wynika to z faktu, że przeglądarki internetowe mają jedynie ograniczone możliwości w zakresie zmiany wielkości strony i obrazów. Tylko autor może zadecydować, które elementy są najważniejsze i tym samym powinny być wyróżnione na małym ekranie oraz które są mniej ważne, przez co można je po prostu pominąć. Dlatego też Twoim obowiązkiem jest zapewnienie witrynie ładnego wyglądu i umożliwienie jej funkcjonalnego działania w różnych przeglądarkach internetowych.

Na szczęście możesz wykorzystać techniki *Adaptive Rendering* i usprawnienia progresywne w celu poprawienia sposobu wyświetlania witryny. Dzięki platformom ASP.NET MVC 4 i jQuery Mobile jest to dość łatwe zadanie, o czym się przekonasz w poniższych podrozdziałach.

Technika Adaptive Rendering

Adaptive Rendering to technika pozwalająca widokowi na „zaadaptowanie" możliwości oferowanych przez przeglądarkę internetową. Przyjmujemy założenie, że na stronie znajduje się mnóstwo kart, a kliknięta karta powoduje wykonanie żądania AJAX w celu pobrania treści i jej wyświetlenia. W przypadku wyłączonej obsługi języka JavaScript karta nie będzie w stanie wyświetlić treści żądanej przez użytkownika. Jednak gdy będzie stosowana technika Adaptive Rendering, karta będzie po prostu zawierała adres URL prowadzący do zasobu z treścią, a tym samym użytkownik uzyska dostęp do żądanej treści.

Innym przykładem może być pasek nawigacyjny zawierający poziomą listę łączy. Wprawdzie przedstawia się on dobrze w urządzeniu biurkowym, ale na małym ekranie urządzenia mobilnego może być przytłaczający. Dzięki technice Adaptive Rendering pasek nawigacyjny będzie mógł zostać wyświetlony jako rozwijane menu, co oznacza dostosowanie jego funkcjonalności do urządzenia o mniejszym ekranie.

Zaletą omawianej techniki jest możliwość wyświetlenia „funkcjonalnej" lub inaczej „użytecznej" wersji witryny w różnych przeglądarkach internetowych i urządzeniach. Sam poziom usługi może być zależny od możliwości oferowanych przez poszczególne urządzenia, ale witryna mimo wszystko nadal pozostanie użyteczna.

Jeżeli chcesz, aby użytkownicy nieustannie powracali na Twoją witrynę, musisz zagwarantować im przyjemność z korzystania z niej niezależnie od tego, jakich używają urządzeń.

Platforma ASP.NET MVC 4 zapewnia techniki adaptacyjne poprzez jQuery Mobile.

Znacznik viewport

W grafice komputerowej pojęcie *viewport* oznacza wyświetlany prostokątny obszar. W przypadku przeglądarki internetowej jest to jej okno, w którym następuje wyświetlanie dokumentu HTML. Innymi słowy, to wyimaginowana konstrukcja zawierająca znacznik `<html>`, który z kolei jest elementem głównym dla całego kodu znaczników strony.

Co się dzieje w przypadku powiększania lub pomniejszania (zmiana stopnia przybliżenia) okna przeglądarki internetowej? A co się stanie po zmianie orientacji urządzenia? Czy viewport również ulegnie zmianie?

W świecie urządzeń mobilnych udzielenie odpowiedzi na powyższe pytania jest nieco utrudnione, ponieważ w rzeczywistości nie istnieje jeden, ale są dwa viewporty — jeden dla „układu graficznego" oraz jeden „wizualny".

Viewport układu graficznego nigdy nie ulega zmianie — to jest wyimaginowana konstrukcja stanowiąca ogranicznik dla znacznika `<html>` strony. Podczas zmiany poziomu przybliżenia strony lub orientacji urządzenia zmienia się viewport wizualny, co wpływa na elementy wyświetlane na ekranie urządzenia.

Rolę viewportu powinieneś postrzegać jako sposób zapewnienia użytkownikom funkcjonalnej i użytecznej witryny internetowej. Kiedy strona jest generowana w urządzeniu mobilnym, jej szerokość nie powinna być ani zbyt duża, ani zbyt mała — powinna być idealnie dopasowana do ekranu. Ponadto strona dopasowana do szerokości ekranu nie powinna być wyświetlana jako mikroskopijna, zmniejszona wersja pełnej strony; powinna to być wersja odpowiednio dostosowana, czytelna.

W nowoczesnych przeglądarkach internetowych to nie CSS, ale znacznik `<meta name="viewport">` pozwala na zdefiniowanie wymiarów wizualnego viewportu:

```
<meta name="viewport" content="width=device-width" />
```

Użyta w powyższym kodzie wartość `"width=device-width"` jest wartością specjalną i oznacza, że szerokość viewportu odpowiada rzeczywistej szerokości ekranu urządzenia. To najbardziej elastyczna i najczęściej używana wartość.

Właściwości `content` można przypisać konkretną wartość liczbową, o ile treść będzie zaadaptowana w ten sposób:

```
<meta name="viewport" content="width=320px" />
```

Po użyciu powyższego kodu niezależnie od szerokości ekranu urządzenia treść zawsze będzie miała szerokość 320 pikseli. Oznacza to, że użytkownicy większych ekranów będą musieli ją powiększać, natomiast mniejszych — zmniejszać.

Znacznik `<meta name="viewport">` jest standardem przemysłowym, ale w rzeczywistości nie jest częścią standardu W3C.

Wymieniona funkcja została po raz pierwszy zaimplementowana w przeglądarce internetowej iPhone'a i bardzo szybko — ze względu na dużą popularność tego smartfona — zaczęła być obsługiwana także przez innych producentów urządzeń mobilnych.

Wykrywanie funkcji mobilnych

Ponieważ każde urządzenie mobilne obsługuje odmienny zestaw funkcji, nigdy nie możesz przyjmować założenia, że konkretna funkcja będzie dostępna w każdej przeglądarce internetowej.

Przyjmujemy założenie, że aplikacja używa funkcji Web Storage HTML5 obsługiwanej wprawdzie przez wiele smartfonów (np. iPhone, Android, Blackberry i Windows Phone), ale jednocześnie nieobsługiwanej przez pozostałe.

Programiści stosowali zazwyczaj takie techniki jak wykrywanie rodzaju przeglądarki internetowej, aby sprawdzić, czy aplikacja może być uruchomiona w danej przeglądarce.

Zamiast więc sprawdzać, czy funkcja Web Storage jest obsługiwana, w podejściu klasycznym sprawdza się, czy używaną przeglądarką internetową jest Opera Mini.

Tego rodzaju podejście wiąże się jednak z pewnymi wadami; niektóre z nich podano poniżej.

- Potencjalne wykluczenie przeglądarek internetowych, które nie zostały wykluczone wyraźnie, ale obsługują daną funkcję.
- Niebezpieczeństwo nieprawidłowego działania witryny internetowej, jeśli użytkownik wyświetli ją z poziomu innego urządzenia.

Oto przykład omówionego podejścia.

```
// Ostrzeżenie: nie stosuj takiego kodu!
if (document.all) {
   // Internet Explorer 4+
   document.write('<link rel="stylesheet" type="text/css" src="style-ie.css">');
}
```

```
else if (document.layers) {
   // Navigator 4
   document.write('<link rel="stylesheet" type="text/css" src="style-nn.css">');
}
```

Zwróć uwagę, że powyższy fragment kodu dostarcza arkusze stylów jedynie dla przeglądarek Internet Explorer i Netscape Navigator 4 pomimo tego, że przeglądarka internetowa musi mieć włączoną obsługę JavaScript. Oznacza to, że w innych przeglądarkach internetowych, takich jak Netscape 6, Netscape 7, CompuServe 7, Mozilla i Opera, dana witryna internetowa może nie być wyświetlona prawidłowo.

Nawet jeśli zapewnisz wyraźną obsługę większości przeglądarek internetowych, nadal możesz pominąć nowe wersje przeglądarek zawierające już obsługę funkcji wymaganych do prawidłowego działania witryny.

Inny potencjalny problem wiąże się z błędną identyfikacją przeglądarki internetowej.

Ponieważ mechanizm wykrywania przeglądarki internetowej w dużej mierze opiera się na zgadywaniu na podstawie ciągu tekstowego określającego agenta użytkownika i na pewnych właściwościach, to istnieje niebezpieczeństwo błędnej identyfikacji pewnych przeglądarek internetowych.

```
// Ostrzeżenie: nie używaj tego kodu!
if (document.all) {
   // Internet Explorer 4+
   elm = document.all['menu'];
}
else {
   // Przyjmujemy założenie, że przeglądarka to Navigator 4.
   elm = document.layers['menu'];
}
```

Zwróć uwagę na stosowany przez powyższy kod podział przeglądarek internetowych. Jeśli przeglądarka nie zostanie rozpoznana jako Internet Explorer, to kod uznaje ją za przeglądarkę Netscape Navigator 4 i następuje próba użycia warstw.

To często spotykane źródło problemów podczas używania przeglądarek Opera i przeglądarek opartych na silniku Gecko.

Wynika z tego, że najlepszym rozwiązaniem jest wyraźne sprawdzenie istnienia danej funkcji zamiast przyjmowanie założenia o jej obsłudze bądź braku obsługi przez konkretne wersje poszczególnych przeglądarek.

Poniżej przedstawiono wcześniejszy fragment kodu, ale zmodyfikowany w celu wykrywania obsługi konkretnych funkcji zamiast przeglądarek.

```
// Jeżeli obsługiwana jest funkcja localStorage, należy jej użyć.
if (('localStorage' in window) && window.localStorage !== null) {
   // Łatwa w użyciu właściwość obiektu.
   localStorage.wishlist = '["Unicorn","Narwhal","Deathbear"]';
} else {
   //  W przypadku braku sessionStorage konieczne jest użycie pliku cookie
   //  za pomocą API document.cookie.
   var date = new Date();
   date.setTime(date.getTime()+(365*24*60*60*1000));
   var expires = date.toGMTString();
   var cookiestr = 'wishlist=["Unicorn","Narwhal","Deathbear"];'+
      ' expires='+expires+'; path=/';
   document.cookie = cookiestr;
}
```

Powyższy fragment kodu nie tylko jest znacznie bardziej niezawodny, ale również będzie doskonale odgrywał swoją rolę w przyszłości — każda przeglądarka, w której zostanie dodana obsługa funkcji Web Storage, otrzyma dostęp do nowych funkcji aplikacji.

Zapytania mediów CSS

Zapytania CSS media to technika usprawnień progresywnych, która pozwala na zaadaptowanie i wyświetlanie alternatywnych stylów w zależności od przeglądarki internetowej.

Wersja druga specyfikacji CSS (znana jako „CSS2") pozwala na wskazanie stylu na podstawie medium, np. `screen` lub `print`.

Z kolei wersja trzecia specyfikacji CSS (znana jako „CSS3") wprowadza koncepcję tak zwanych *zapytań mediów*, czyli technikę rozszerzającą koncepcję pomagającą wykrywać w standardowy sposób funkcje oferowane przez przeglądarki internetowe.

Niestety, specyfikacja CSS3 nadal pozostaje w fazie „rekomendacji", co oznacza, że zapytania mediów — oraz oczywiście inne nowe funkcje wprowadzone w specyfikacji CSS3 — niekoniecznie są doskonale obsługiwane przez wszystkie przeglądarki internetowe.

Dlatego też bardzo ważne jest przygotowanie stylów domyślnych dostarczanych przeglądarkom internetowym, które nie obsługują nowych funkcji.

Wcześniej dowiedziałeś się, jak znacznik viewport pozwala na zdefiniowanie domyślnej szerokości na podstawie ekranu urządzenia. Wprawdzie viewport powoduje, że strona wygląda dobrze w standardowej wielkości, ale nie będzie pomocny, gdy użytkownik zechce powiększyć lub zmniejszyć stronę w urządzeniu.

Wraz ze zmianą układu graficznego konieczne jest znalezienie sposobu na poinformowanie przeglądarki internetowej, jak ograniczyć treść do konkretnej szerokości, aby była wyświetlana poprawnie w każdej sytuacji.

Spójrz na prosty przykład rozwiązania opartego na zapytaniu mediów CSS:

```
body {background-color:blue;}
@media only screen and (max-width: 800px) {
   body {background-color:red;}
}
```

Reguły CSS są wykonywane od początku pliku do końca, więc rozpoczynamy od umieszczenia ogólnej reguły definiującej niebieski kolor tła strony.

Następnie znajduje się reguła przeznaczona dla konkretnego urządzenia — wykorzystane zostaje zapytanie mediów CSS. Reguła powoduje zmianę koloru tła na czerwony w urządzeniach, których szerokość ekranu nie przekracza 800 pikseli.

W urządzeniach obsługujących zapytania mediów CSS i posiadających ekran o szerokości poniżej 800 pikseli tło będzie wyświetlone w kolorze czerwonym. Natomiast w pozostałych urządzeniach kolorem tła pozostaje niebieski. (Zauważ, że zmiana koloru tła podczas zmiany wielkości strony nie jest operacją wykonywaną w rzeczywistych aplikacjach. Zamiast tego w omówionym przykładzie koncentrujemy się na pokazaniu sposobu używania zapytania mediów w celu stosowania różnych stylów na podstawie określonych warunków).

Bardzo ważne jest umieszczenie ogólnej reguły na początku, a następnie jej usprawnianie poprzez dodanie obsługi zapytań mediów i wykrywania dostępności poszczególnych funkcji.

Dzięki temu witryna internetowa będzie wyświetlana w pełnej krasie w przeglądarkach internetowych oferujących obsługę najnowszych funkcji i jednocześnie będzie wyświetlana poprawnie (choć mniej atrakcyjnie pod względem wizualnym) w starszych wersjach przeglądarek internetowych.

Widoki dla konkretnych przeglądarek internetowych

Nowa funkcja platformy ASP.NET MVC 4, jaką jest tryb wyświetlania, pozwala na wczytywanie odmiennych widoków w zależności od zdefiniowanych warunków. Przykładem użycia tej funkcji może być utworzenie oddzielnych widoków przeznaczonych dla smartfonów (urządzeń o małych ekranach) i tabletów (urządzeń o ekranach większych niż w smartfonach, ale jednocześnie mniejszych niż w urządzeniach biurkowych). Przygotowanie widoków dla tych klas urządzeń pozwala na optymalne wykorzystanie dostępnej powierzchni ekranu. Ponadto stanowi przykład dostarczenia efektywnej i użytecznej aplikacji przystosowanej do możliwości oferowanych przez konkretne urządzenia.

Pierwszym krokiem podczas realizacji wymienionego podejścia jest rejestracja trybów wyświetlania przeprowadzana w trakcie uruchamiania aplikacji.

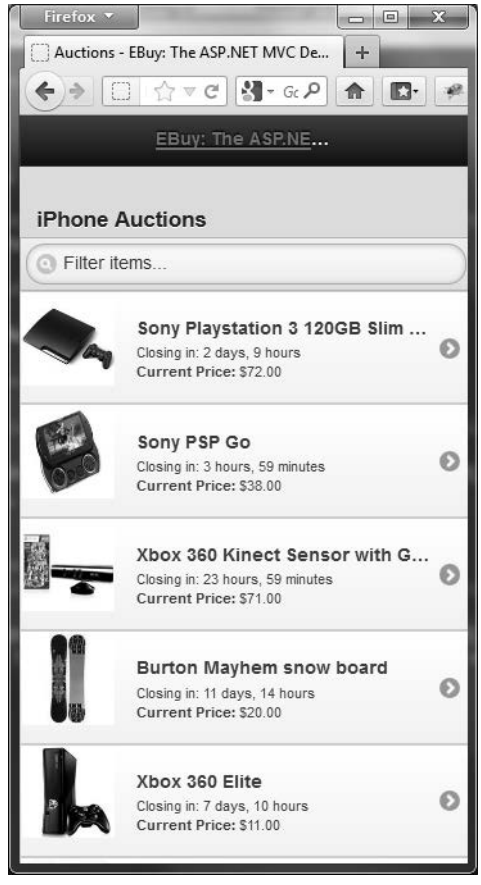
```
using System.Web.WebPages;

// Rejestracja widoku przeznaczonego dla iPhone'a.
DisplayModeProvider.Instance.Modes.Insert(0, new DefaultDisplayMode("iPhone")
{
   ContextCondition = (ctx => ctx.Request.UserAgent.IndexOf(
      "iPhone", StringComparison.OrdinalIgnoreCase) >= 0)
});

// Rejestracja widoku przeznaczonego dla Windows Phone.
DisplayModeProvider.Instance.Modes.Insert(0, new DefaultDisplayMode("WindowsPhone")
{
   ContextCondition = (ctx => ctx.Request.UserAgent.IndexOf(
      "Windows Phone", StringComparison.OrdinalIgnoreCase) >= 0)
});
```

Drugim krokiem jest przygotowanie odpowiednich widoków. Utwórz widok dla iPhone'a poprzez skopiowanie widoku *Auctions.mobile.cshtml* i zmianę jego nazwy na *Auctions.iPhone.cshtml*. Następnie nagłówek w nowym widoku zmień na *Aukcje iPhone'a*, aby go odróżnić od pozostałych. Aby zobaczyć tak przygotowaną stronę w działaniu (rysunek 10.9), uruchom aplikację za pomocą emulatora przeglądarki internetowej dla urządzenia mobilnego (pokazane tutaj przykłady używają rozszerzenia *User Agent Switcher* dla przeglądarki internetowej Firefox pozwalającego na emulację przeglądarki internetowej w iPhonie).

Aby przygotować wersję widoku dla Windows Phone, utwórz kolejną kopię *Auctions.mobile.cshtml* i zmień jego nazwę na *Auctions.WindowsPhone.cshtml*. Następnie nagłówek w nowym widoku zmień na *Aukcje Windows Phone*, aby go odróżnić od pozostałych. Po uruchomieniu aplikacji w emulatorze przeglądarki internetowej na urządzeniu mobilnym (rysunek 10.10) będziesz mógł podziwiać przygotowaną stronę w działaniu.

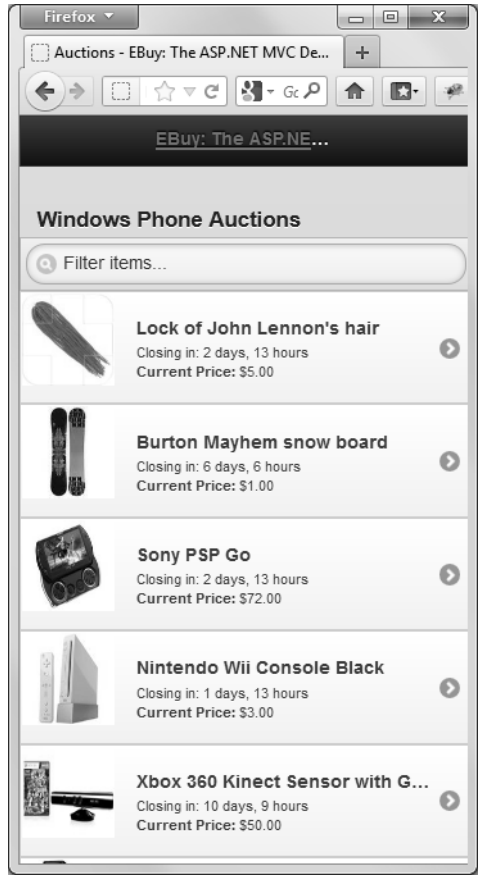


Rysunek 10.9. Widok utworzony dla iPhone'a

Aby sprawdzić, czy żądanie pochodzi z urządzenia mobilnego, platforma ASP.NET analizuje je i porównuje z zestawem doskonale znanych definicji przeglądarek internetowych dla urządzeń mobilnych.

Dzięki klasie `HttpBrowserCapabilities` (*http://msdn.microsoft.com/en-us/library/system.web.httpbrowsercapabilities.aspx*) platforma otrzymuje bardzo dużą ilość informacji dotyczących możliwości danej przeglądarki. Informacje te są dostępne poprzez właściwość `Request.Browser`.

Ewentualnie zamiast polegać na wbudowanych w platformę definicjach przeglądarek, możesz skorzystać z usługi takiej jak *51Degrees.mobi* (*http://51degrees.mobi/Support/Blogs/tabid/212/EntryId/26/51Degrees-mobi-and-MVC4.aspx*), która oferuje znacznie bardziej aktualną bazę informacji o różnych urządzeniach mobilnych.



Rysunek 10.10. Widok utworzony dla Windows Phone

Tworzenie nowej aplikacji mobilnej zupełnie od początku

Platforma ASP.NET MVC 4 bardzo ułatwia dodawanie widoków mobilnych do istniejących aplikacji. Równie łatwo możesz jednak zupełnie od początku tworzyć aplikacje mobilne. To użyteczne rozwiązanie, jeśli nie masz jeszcze aplikacji, którą mógłbyś wykorzystać jako punkt wyjścia, bądź gdy z jakiegoś powodu nie chcesz łączyć witryn w wersji zwykłej i mobilnej.

Na platformie ASP.NET MVC 4 znajdziesz szablon *Aplikacja dla urządzeń przenośnych*, dzięki któremu szybko będziesz mógł rozpocząć tworzenie aplikacji dla urządzeń mobilnych. Szablon ten w dużej mierze opiera się na platformie jQuery Mobile. Aby zatem rozpocząć tworzenie efektywnych aplikacji, w pierwszej kolejności musisz poznać jQuery Mobile.

Platforma jQuery Mobile

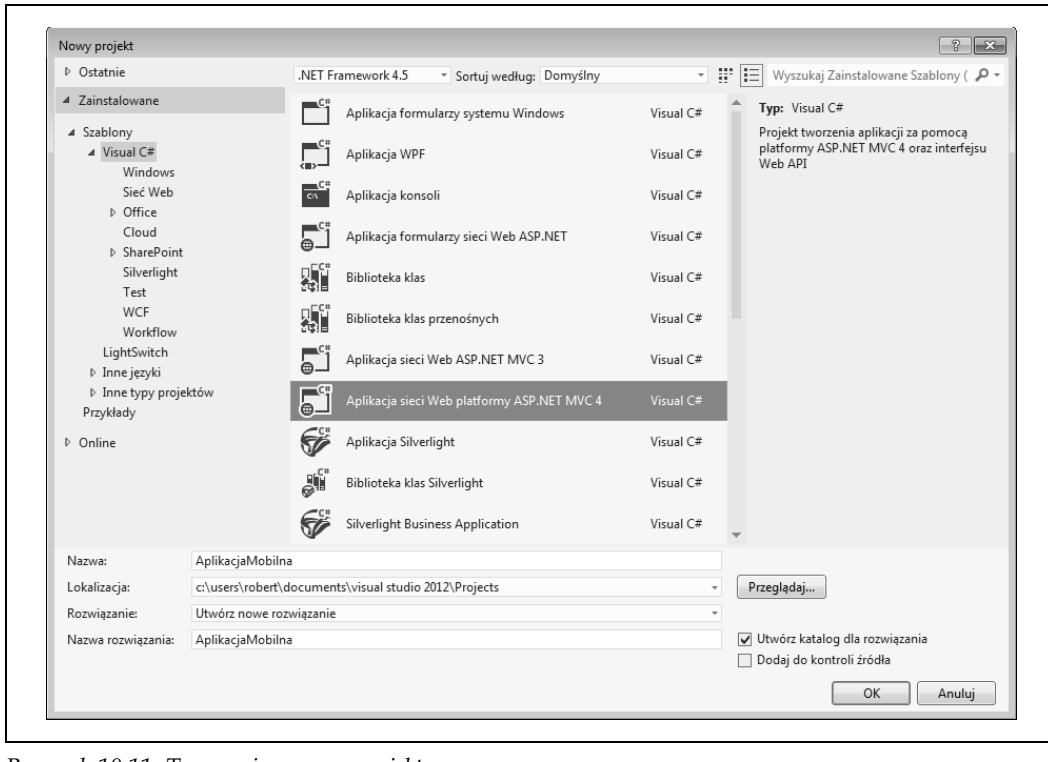
Podczas pracy z jQuery Mobile bardzo istotne jest pojęcie strony. W tradycyjnym programowaniu sieciowym strona oznacza dokument HTML, stronę *.aspx* w ASP.NET Web Forms lub widok *.cshtml* na platformie ASP.NET MVC. Pliki te zawierają kod znaczników oraz logikę pozwalające na wygenerowanie strony w przeglądarce internetowej.

Jednak na platformie jQuery Mobile jeden plik może zawierać wiele „stron" dla urządzeń mobilnych. Z technicznego punktu widzenia strona jQuery Mobile to tak naprawdę znacznik `<div>` wraz z przypisanym mu atrybutem `data-role="page"`. W jednym pliku widoku możesz umieścić dowolną liczbę takich znaczników, a jQuery zamieni je na wiele stron wyświetlanych jednocześnie.

Ponieważ na podstawie jednego widoku dla urządzenia biurkowego może powstać wiele mniejszych widoków dla urządzenia mobilnego (głównie wskutek przeprojektowania strony, aby jej nawigacja lepiej odpowiadała ekranowi urządzenia mobilnego), takie podejście pomaga w zmniejszeniu liczby plików; w przeciwnym razie pliki te trzeba by utworzyć.

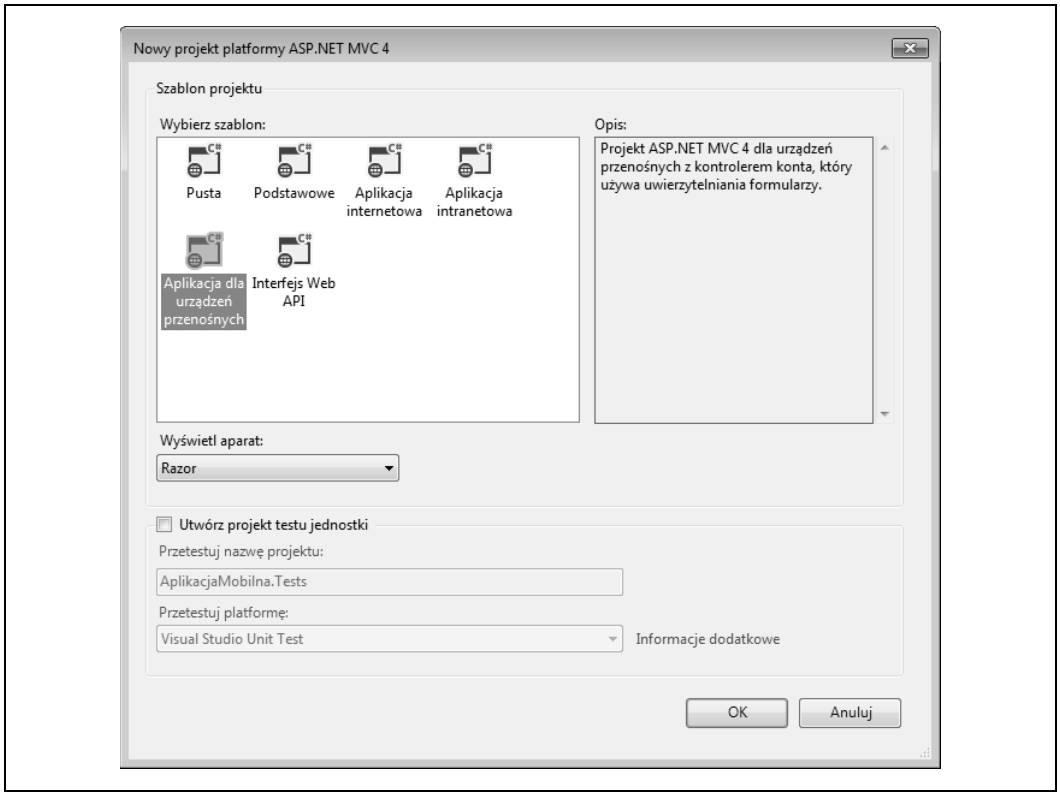
Szablon aplikacji mobilnej w ASP.NET MVC 4

Aby utworzyć nową aplikację mobilną, rozpoczynasz pracę dokładnie tak samo jak w przypadku zwykłej aplikacji sieciowej ASP.NET MVC — wybierz opcję menu *Plik/Nowy/Projekt…*, a następnie wybierz typ *Aplikacja sieci Web platformy ASP.NET MVC 4* (rysunek 10.11).



Rysunek 10.11. Tworzenie nowego projektu

W kolejnym oknie dialogowym wybierz szablon *Aplikacja dla urządzeń przenośnych* (rysunek 10.12).



Rysunek 10.12. Wybór szablonu aplikacji dla urządzeń przenośnych

W ten sposób utworzysz nową aplikację ASP.NET MVC wraz z przykładowymi kontrolerami i widokami zawierającymi oferowane przez platformę ASP.NET MVC funkcje obsługi urządzeń mobilnych. Szablon ten pozwoli Ci na szybkie rozpoczęcie pracy.

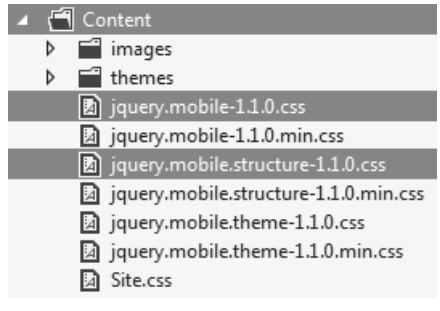
Uruchom projekt, naciskając klawisz *F5* bądź wybierając z menu *Debug/Start*. Visual Studio rozpocznie kompilację rozwiązania, a następnie uruchomi przeglądarkę internetową, w której zostanie wyświetlona strona główna witryny internetowej w wersji przystosowanej dla urządzeń mobilnych (rysunek 10.13).

Używanie szablonu aplikacji mobilnej w ASP.NET MVC 4

Jak możesz się przekonać, utworzony projekt aplikacji mobilnej standardowo zawiera dużą ilość przygotowanego kodu. Struktura projektu jest podobna do projektu zwykłej witryny internetowej, ale z dwoma zmianami.

- Katalog *Content* zawiera arkusze stylów dla jQuery Mobile (rysunek 10.14):
  - *jquery.mobile-1.1.0.css* (i jego zminimalizowana wersja),
  - *jquery.mobile.structure-1.1.0.css* (i jego zminimalizowana wersja).

Rysunek 10.13. Domyślna strona główna aplikacji mobilnej

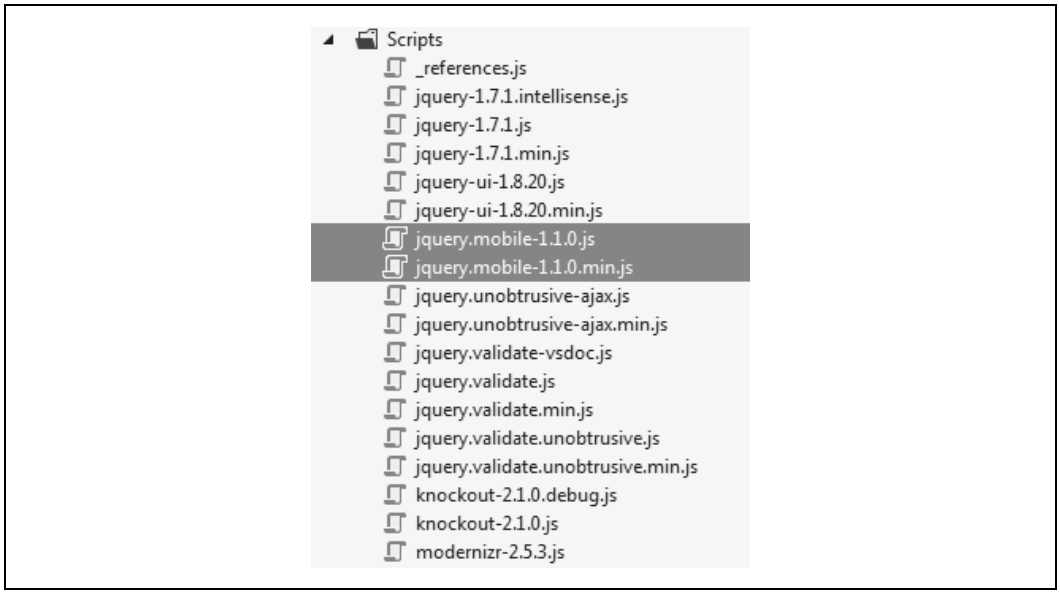


Rysunek 10.14. Nowa zawartość katalogu Content projektu

- Katalog *Scripts* zawiera dwa nowe pliki (rysunek 10.15):
  - *jquery.mobile-1.1.0.js*,
  - *jquery.mobile.structure-1.1.0.js*.

Te nowe pliki stanowią część platformy jQuery Mobile, czyli platformy JavaScript dostarczającej urządzeniom mobilnym całe dobrodziejstwo jQuery i jQueryUI.

Teraz spójrz na zmodyfikowaną wersję pliku *_Layout.cshtml*, którego znacznik `<head>` zawiera kilka nowych wierszy.



Rysunek 10.15. Nowa zawartość katalogu Scripts projektu

Znacznik `<meta name="viewport">` określa wielkość viewportu. To jest bardzo ważne, ponieważ większość przeglądarek w wersji dla urządzeń mobilnych wprawdzie pozwala użytkownikom na zmniejszanie i powiększanie strony wedle własnego uznania, ale lepsze wrażenie wywoła w użytkowniku ustawienie szerokości początkowej. Jak już wcześniej powiedziano, wartość `"width=device-width"` powoduje, że treści jest automatycznie przypisywana szerokość odpowiadająca szerokości używanego urządzenia:

```
<meta name="viewport" content="width=device-width" />
```

Alternatywnym rozwiązaniem jest zdefiniowanie konkretnej szerokości dla viewportu poprzez podanie odpowiedniej wartości wyrażonej w pikselach. Na przykład w poniższym poleceniu początkowa szerokość strony została określona na 320 pikseli:

```
<meta name="viewport" content="width=320px" />
```

Przedstawiony poniżej znacznik powoduje dodanie do strony stylów jQuery Mobile. Ponadto pozwala na konfigurację motywów za pomocą jQuery Theming (*http://jquerymobile.com/demos/1.0/docs/api/themes.html*):

```
<link rel="stylesheet" a href="@Url.Content("~/Content/jquery.mobile-1.0b2.min.css")" />
```

Wreszcie poniższy znacznik skryptu dodaje do strony pliki platformy jQuery Mobile. W ten sposób możliwe staje się wykonywanie operacji w technologii AJAX, animacji, sprawdzania poprawności danych itd.:

```
<script type="text/javascript" src="@Url.Content("~/Scripts/jquery.mobile-1.0b2.min.js")">
↪</script>
```

Teraz spójrz na zmodyfikowany kod HTML strony, zawierający kilka nowych atrybutów. Platforma jQuery Mobile identyfikuje różne elementy, takie jak strony, przyciski, `listview` itd., dzięki atrybutom `data-role`. Analizując zawartość znacznika `<body>`, możesz dostrzec, że szablon standardowo umieścił atrybuty `data-role` w znacznikach `<div>`:

```
<body>
   <div data-role="page" data-theme="b">
      <div data-role="header">
         @if (IsSectionDefined("Header")) {
            @RenderSection("Header")
         } else {
            <h1>@ViewBag.Title</h1>
            @Html.Partial("_LogOnPartial")
         }
      </div>

      <div data-role="content">
`        @RenderBody()
      </div>
   </div>
</body>
```

Pierwszy znacznik `<div>` ma przypisany atrybut `data-role="page"`, który identyfikuje ten znacznik `<div>` jako pojedynczą stronę w aplikacji mobilnej. Podobnie nagłówek strony został określony atrybutem `data-role="header"`, natomiast znacznik `<body>` atrybutem `data-role="content"`.

Platforma jQuery Mobile definiuje różne atrybuty dla doskonale znanych elementów HTML, takich jak `<h1>`, `<h2>`, `<p>`, `<table>`, a także dla list i elementów formularza, np. przycisków, pól tekstowych, list wyboru itd. Więcej informacji na temat platformy jQuery Mobile, dokładną dokumentację, przykłady itd. znajdziesz na witrynie internetowej *http://jquerymobile.com/*.

Podsumowanie

W rozdziale przedstawiono różne aspekty programowania sieciowego dla urządzeń mobilnych. Dowiedziałeś się, czym tak naprawdę jest aplikacja mobilna i jakie są różnice pomiędzy witrynami internetowymi przeznaczonymi dla urządzeń mobilnych i dla urządzeń biurkowych. Zaprezentowano różne platformy oraz dostępne techniki, dzięki którym możesz wydajnie tworzyć aplikacje mobilne. Dowiedziałeś się, jak dzięki wykorzystaniu różnych możliwości oferowanych przez przeglądarki internetowe zapewnić użytkownikowi jak najlepsze wrażenia podczas używania aplikacji.

Przedstawiono także oferowane przez platformę ASP.NET MVC 4 funkcje pomagające w tworzeniu aplikacji mobilnych:

- usprawnienia w domyślnym szablonie aplikacji mobilnej;
- możliwość dostosowania do własnych potrzeb szablonu domyślnego poprzez zmianę jego układu graficznego, widoków i widoków częściowych;
- obsługę funkcji oferowanych przez konkretne przeglądarki internetowe (np. widok dla iPhone'a) oraz opcję nadpisania możliwości przeglądarek internetowych;
- usprawnienie widoków mobilnych poprzez użycie platformy jQuery Mobile.

Skorowidz

A

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

S

T

U

W

X

Z

Ż

Programowanie

O'REILLY®

ASP.NET MVC 4. Programowanie

Platforma ASP.NET to główny konkurent języka Java w zakresie tworzenia aplikacji internetowych oraz zaawansowanych stron internetowych. Jej autorzy zadbali o to, aby każda kolejna wersja ułatwiała pracę programistom w coraz szerszym zakresie. Programiści to doceniają i ASP.NET znajduje się wśród języków najczęściej wybieranych przy tworzeniu zaawansowanych projektów.

Jeżeli chcesz w pełni wykorzystać potencjał ASP.NET MVC 4, przyda Ci się wyjątkowa książka. Ta, którą trzymasz w rękach, bez wątpienia taka jest! W trakcie lektury poznasz niuanse architektury MVC oraz dowiesz się, jak tworzyć sieciowe API. Ponadto wykorzystasz Entity Framework do wydajnego korzystania z baz danych oraz zaznajomisz się ze sposobami na równoległe przetwarzanie żądań. Szczególną uwagę powinieneś zwrócić na rozdział poświęcony zapewnieniu jakości — wykorzystanie testów automatycznych znacząco ułatwi Ci życie! Książka ta powinna trafić na podręczną półkę każdego programisty ASP.NET!

Sięgnij po tę książkę i:

- poznaj wzorzec MVC
- stwórz zaawansowane Web API
- zobacz, jak uatrakcyjnić Twoją aplikację dzięki technologii AJAX
- zbuduj bezpieczną aplikację
- zbuduj system testów automatycznych oraz ciągłej integracji

Poznaj i wykorzystaj możliwości ASP.NET w Twoim projekcie!

ISBN 978-83-246-6644-7

9 788324 666447

Cena 79,00 zł