

Лабораторный­ практикум

Курс: Размещённые на хосте веб-приложения и WebViews

Сентябрь 2015 года

Содержание

**No table of contents entries found.**

Обзор

После выхода Windows 10 набор инструментов Windows Windows Bridge обеспечил доступ платформы UWP для разработчиков под Android, iOS, классических версий ОС Windows и веб-приложений. Windows Bridge для веб-приложений позволяет без проблем адаптировать ваш код для Веб с последующей публикацией вашего адаптивного веб-сайта на Windows Store. Веб-приложения и размещённые на хосте веб-приложения имеют возможность вызывать API UWP напрямую из JavaScript для интеграции с такими функциями как "живые" плитки, активные уведомления, контакты, помощник Cortana и платный контент в приложениях Windows Store.

Размещённые на хосте веб-приложения немедленно отражают изменения, вносимые в кодовую базу, что облегчает обновление контента в приложении. Вы можете смешивать локальный и удалённый контент, чтобы обеспечить наличие автономных резервных копий для сайта.

После создания веб-приложения для Windows, вы, возможно, заинтересуетесь разработкой под другими платформами. ManifoldJS является новым фреймворком с открытым исходным кодом, который генерирует размещённые на хосте веб-приложения для основных платформ.

# Цели

* 1. Пройдя этот курс, вы сможете:
  + Создавать размещённые на хосте веб-приложения
  + Обслуживать удалённый и локальный контент
  + Совмещать онлайн- и оффлайн-контент
  + Вызывать API платформы из веб-приложения
  + Генерировать размещённые на хосте веб-приложения с помощью ManifoldJS

# Системные требования

* 1. Чтобы выполнить этот курс, необходимо обладать следующим набором программных инструментов:
  + Microsoft Windows 10
  + Microsoft Visual Studio 2015
  + Подписка Azure или бесплатная пробная версия

# Дополнительные надстройки

* 1. Если вы захотите проходить дополнительные задачи в этом курсе, вам понадобятся:
  + Node Package Manager (npm)
  + ManifoldJS

# Настройка

* 1. Вы должны осуществить следующие шаги для подготовки своего компьютера для этого курса:
  2. Установите Microsoft Windows 10.
  3. Установите Microsoft Visual Studio 2015. Выберите пользовательскую установку и убедитесь, что Инструменты разработки для приложений Windows выбраны из списка дополнительных функций.
  4. Дополнительно: Установите npm.
  5. Дополнительно: Установите ManifoldJS

*Инструкции и ссылки на установку npm и ManifoldJS вы найдете в Упражнении 3: Задача 1.*

# Упражнения

* 1. Настоящий Лабораторный практикум включает следующие упражнения:
  2. Добавление размещённого на хосте простого веб-приложения
  3. Интеграция с API платформы
  4. Поддержка дополнительных платформ и устройств с помощью ManifoldJS (дополнительно)
  5. Расчетное время для завершения курса: **От 45 до 60 минут**.

Упражнение 1: Взаимодействие с API платформы

* 1. В Windows 10 вы имеете возможность создать непрерывный пользовательский опыт для веб-приложений, взаимодействуя с API платформы. В этом упражнении вы запустите всплывающее уведомление из веб-приложения, которое включает контент, отображаемый через WebView. Хотя в этом курсе веб-контент, отображаемый в WebView, расположен локально, а не на хосте, курс демонстрирует, как JavaScript, запущенный в WebView, может вызвать код C# в вашем аппаратном проекте.

Задача 1 – Создание пустых универсальных JavaScript-приложений для Windows

Начнём с создания проекта из шаблона пустого JavaScript-приложения UWP .

1. В новой версии Visual Studio 2015 выберите **File (Файл) -> New (Новый) -> Project (Проект)**, чтобы открыть диалоговое окно New Project (Новый проект). Далее перейдите**Installed (Установленное)** -> Templates (Шаблоны) -> **JavaScript** и выберите **Blank App (Universal Windows) (Пустой шаблон)**.
2. Назовите свой проект **WebApp** и выберите местоположение файловой системы, в которое будет осуществлено сохранение результатов прохождения Лабораторного практикума. На диске **C** создана папка под именем **"HOL"**, информация о которой будет представлена в скриншотах.
3. Выберите опции **Create new solution (Создать новое решение)** и **Create directory for solution (Создать папку для решения)**. Вы можете снять галочку с опции **Add to source control (Добавить контроль исходного кода)**, если вы не захотите обновлять версию своей работы. Нажмите **ОК** для создания проекта.

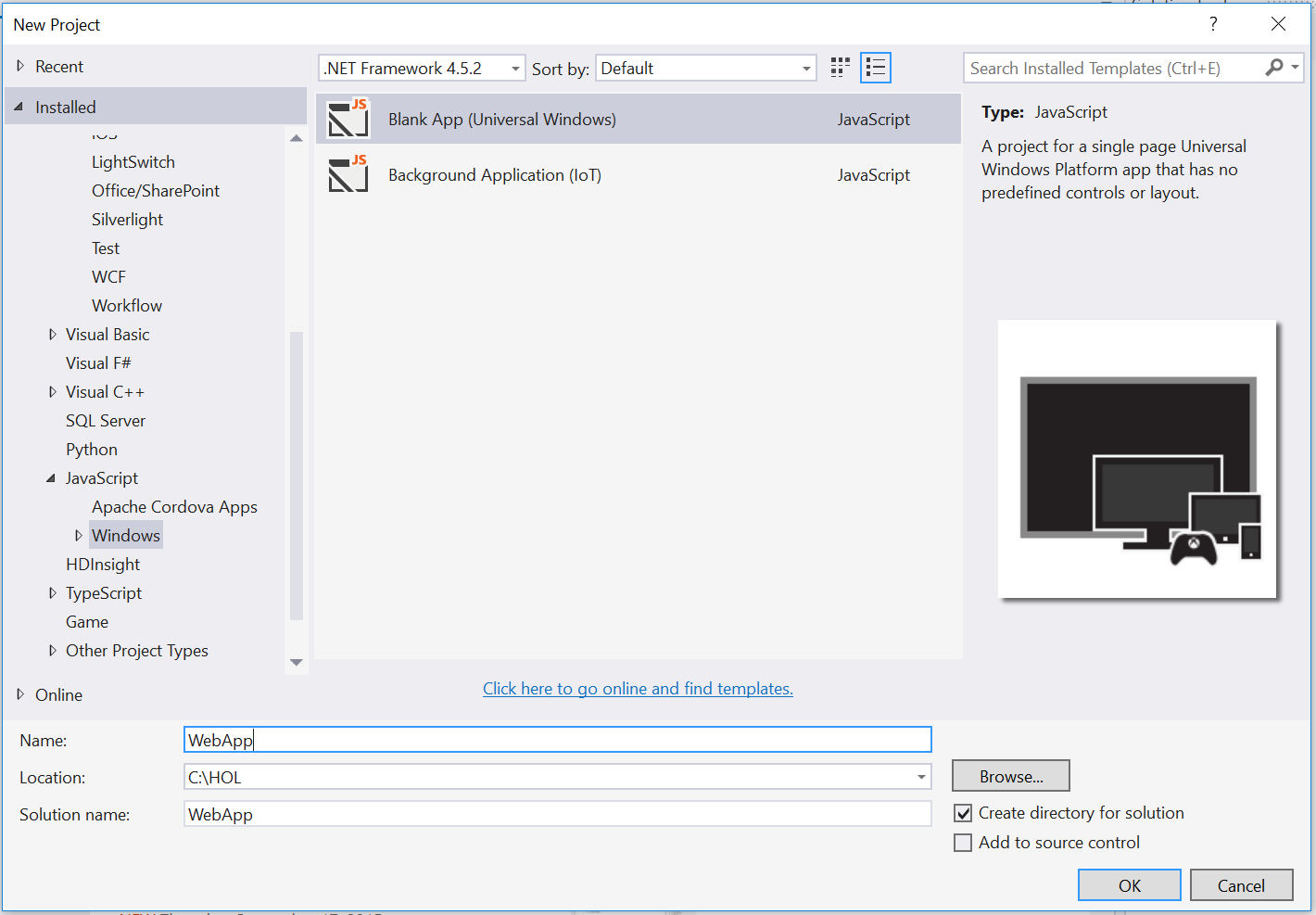


Рисунок 1

Создание нового проекта приложения в Visual Studio 2015.

1. Настройте  **Solution Configuration (Конфигурацию решения)** на **Debug (Отладку)** и  **Solution Platform (Платформу решений)** на **x86**. Выберите **локальный компьютер** в выпадающем списке Debug Target (Цели отладки) возле кнопки Start Debugging (Запустить отладку).

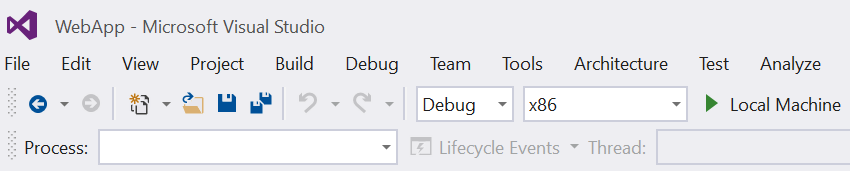


Рисунок 2

* + 1. Конфигурация приложение для запуска на локальном компьютере.
  1. **Примечание:**  Кнопка Start Debugging (Запуск отладки).

1. Нажмите кнопку Start Debugging (Запуск отладки), чтобы создать и запустить своё приложение. На экране будет отображен чёрный фон приложения с текстом “Content goes here.” (Контент располагается здесь).

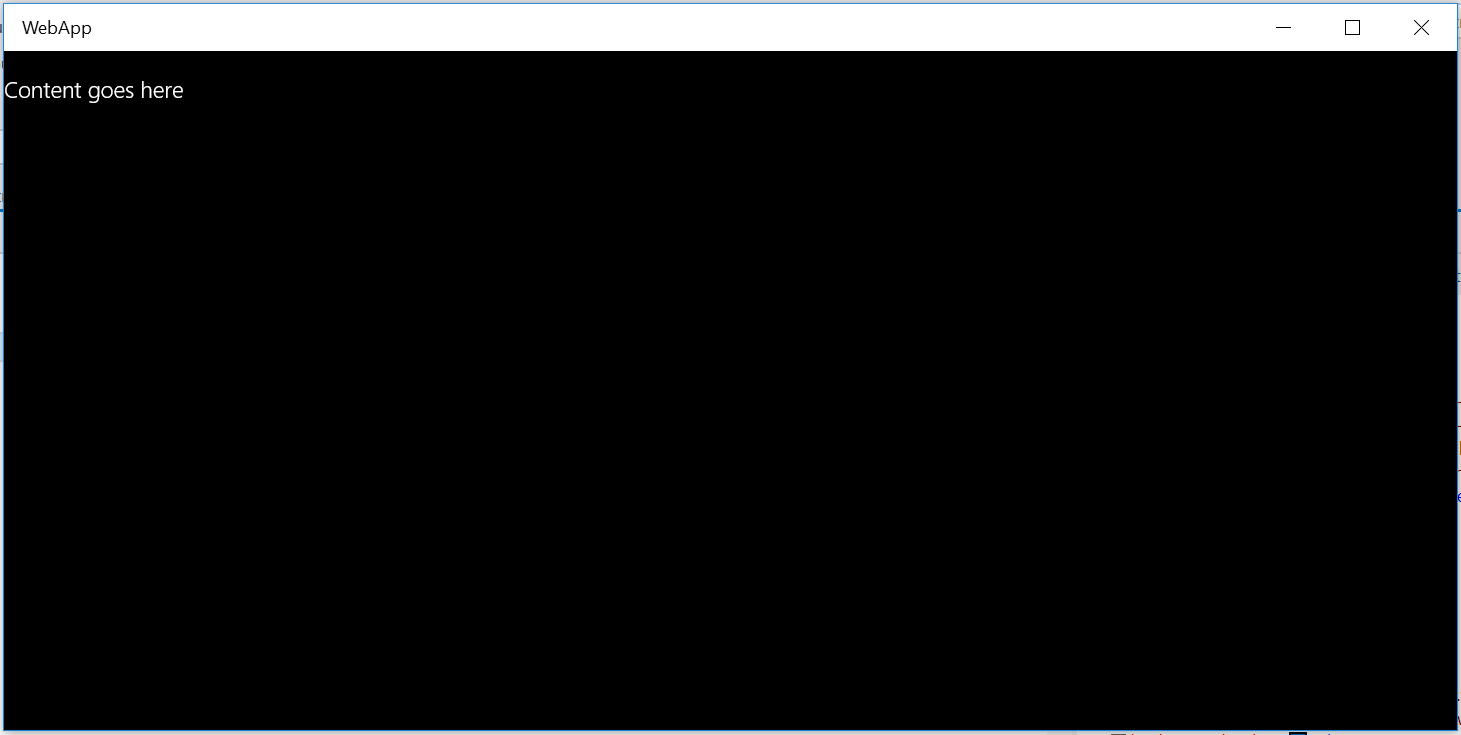


Рисунок 3

Пустое универсальное приложение, запущенное в режиме настольного компьютера.

1. Отключите отладку и вернитесь к Visual Studio.

Задача 2 – Создание аппаратного проекта

Цель данного упражнения состоит в том, чтобы запустить всплывающее уведомление из webview. В этой задаче вы создадите компонент Windows Runtime, чтобы обработать уведомление через API платформы.

1. В вашем проекте WebApp кликните правой кнопкой мыши на имени решения в Solution Explorer (Обозреватель решений) и выберите **Add (Добавить) -> New Project (Новый проект)**.
2. **Выберите Visual C# > Windows > Windows Runtime Component (Universal Windows)** и присвойте ей имя **CommunicationWinRT**. Нажмите **OK** для создания проекта.

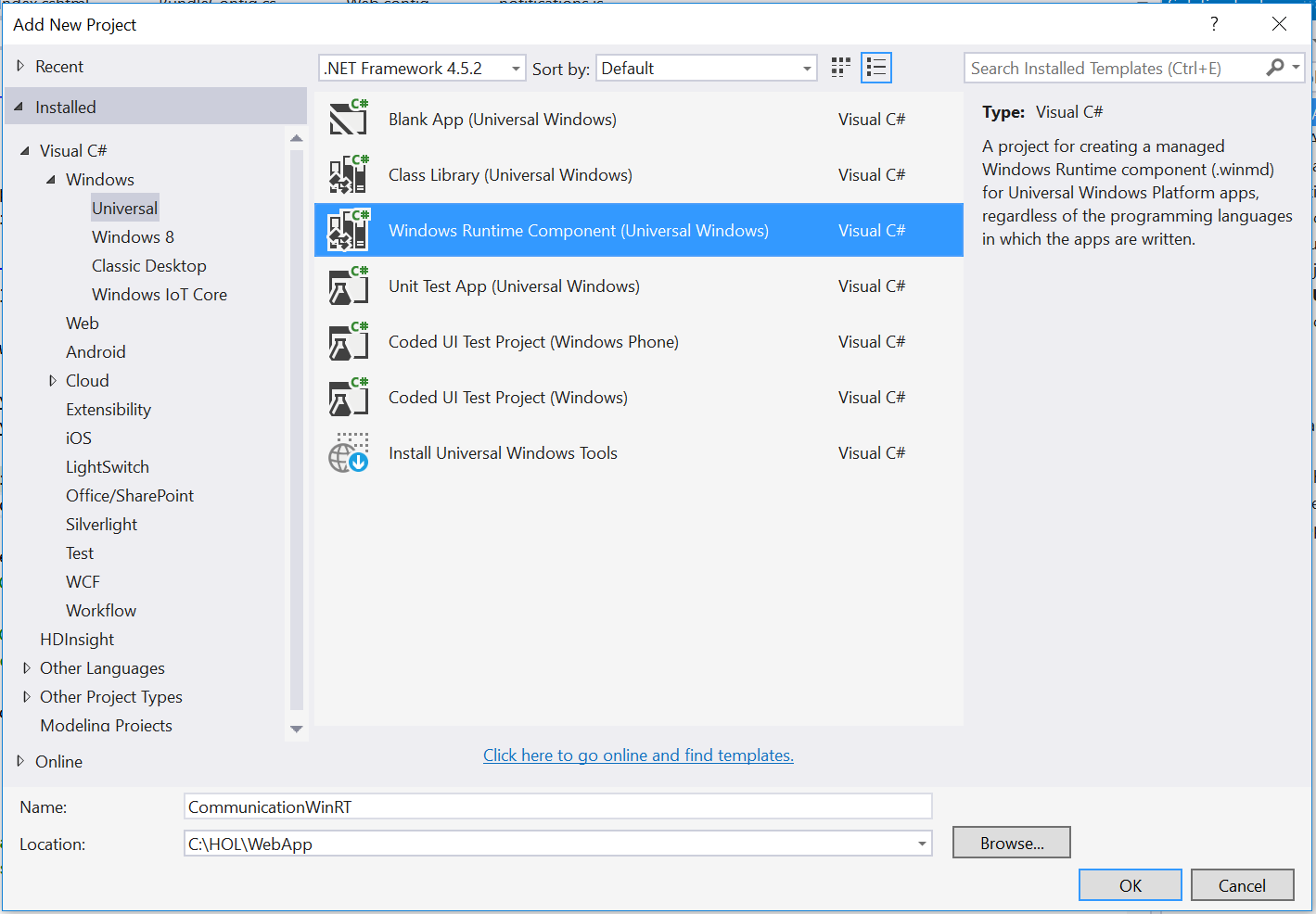


Рисунок 4

Добавление проекта CommunicationWinRT.

1. Кликните правой кнопкой мыши на файле **Class1.cs** в Solution Explorer и выберите **Rename (Переименовать)**, чтобы переименовать его в **CommunicationWinRT**. При появлении запроса на выполнение переименования в проекте всех ссылок на Class1, выберите **Yes (Да)**.

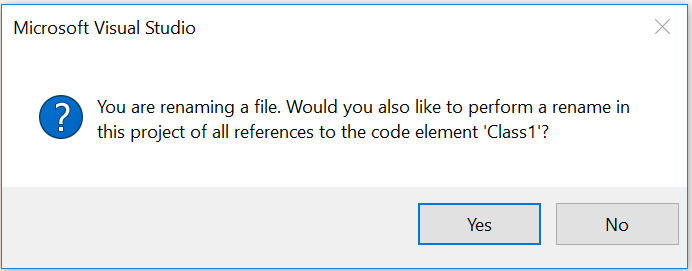


Рисунок 5

Переименование Class1.cs в CommunicationWinRT.cs.

1. Откройте **CommunicationWinRT.cs**. Добавьте пространства имен **Windows.UI.Notifications** и **Windows.Data.Xml.Dom** в класс.
   * 1. C#
   1. using Windows.UI.Notifications;
   2. using Windows.Data.Xml.Dom;
2. Добавьте метаконтент **AllowForWeb** и конструктор в класс **CommunicationWinRt**.
   * 1. C#
   1. namespace CommunicationWinRT
   2. {
   3. [Windows.Foundation.Metadata.AllowForWeb]
   4. public sealed class CommunicationWinRT
   5. {
   6. **Примечание:** Запечатанный класс не может быть унаследован или расширен свойствами JavaScript. Объекты WinRT запечатаны от JavaScript, так что JavaScript не может мешать их поведению.
   7. Для объекта Windows Runtime, проектируемоего и доступного для WebView, обязательно наличие атрибута AllowForWeb.
3. Добавьте метод **async** под конструктором, который принимает значение строки и задержки, которые будут использоваться для создания всплывающего уведомления.
   * 1. C#
   1. public sealed class CommunicationWinRT
   2. {

public async void toastMessage(String message, int delay)  
 {  
 ToastTemplateType toastTemplate = ToastTemplateType.ToastText01;  
 XmlDocument toastXml = ToastNotificationManager.GetTemplateContent(toastTemplate);

* 1. XmlNodeList toastTextElements = toastXml.GetElementsByTagName("text");  
      toastTextElements[0].AppendChild(toastXml.CreateTextNode(message));
  2. ToastNotification toast = new ToastNotification(toastXml);
  3. await Task.Delay(delay);  
      ToastNotificationManager.CreateToastNotifier().Show(toast);  
      }

1. Сохраните **CommunicationWinRT.cs** и вернитесь к проекту.
   1. **Примечание:** Вы можете добавить дополнительные запечатанные открытые классы к пространству имён **CommunicationWinRT**, чтобы реализовать интеграцию с другими API.
   2. Для получения более подробной информации об уведомлениях, см. курс**"Живые" плитки и уведомления**.
2. Кликните правой кнопкой мыши на папке **References (Ссылки** в проекте **WebApp** и выберите **Add (Добавить) -> Reference (Ссылка)**. Добавьте проект **CommunicationWinRT** в качестве ссылки.

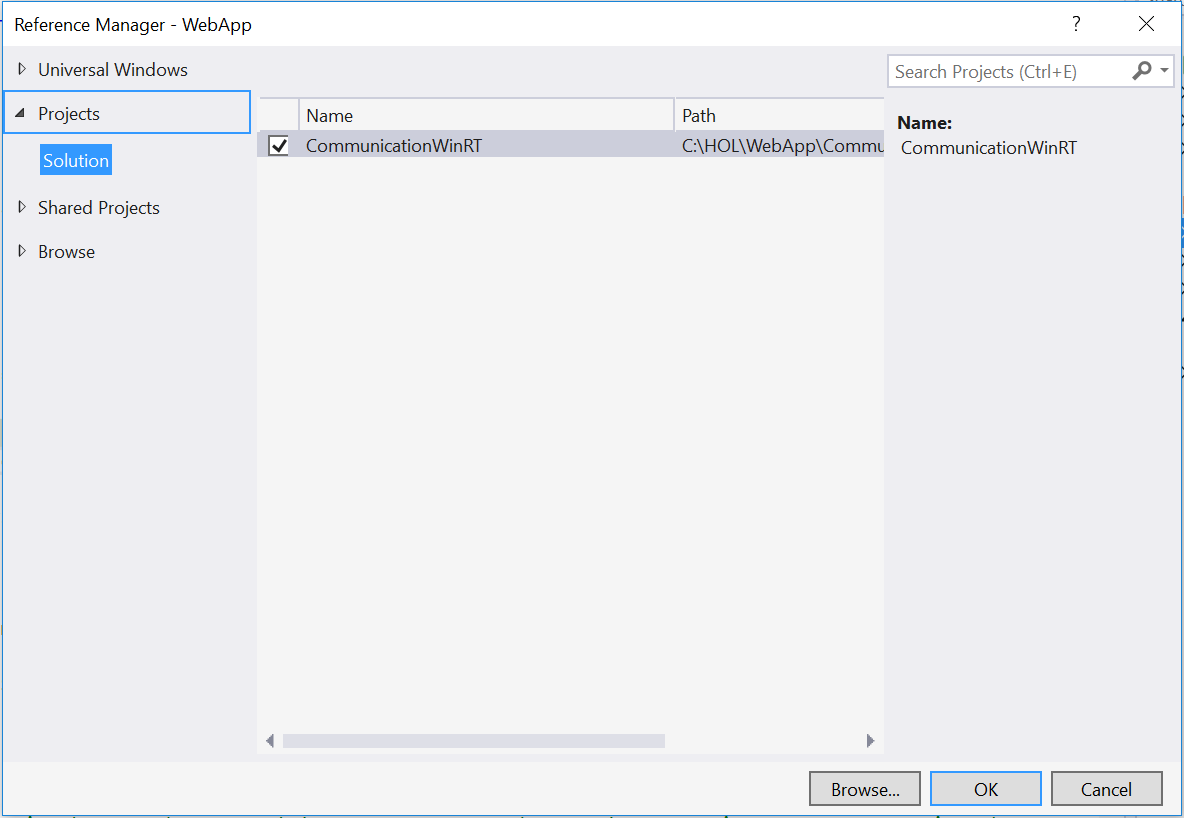


Рисунок 6

Добавление аппаратного проекта как ссылки на веб-проект.

Задача 2 – Добавление веб-вида для веб-проекта

В Windows 10 ваше JavaScript-приложение может напрямую организовать навигацию на удалённый URI (Универсальный идентификатор ресурсов). Однако, WebView ещё может понадобиться, когда вам будет нужно более подробное управление. В этой задаче вы добавите элемент webview в default.html и подготовите свой проект для отображения локального содержания.

* 1. **Примечание:** WebView может работать локально и указывать на контент на веб-сервере и наоборот. В Windows 10 WebView сейчас отделён от потока UI для улучшения производительности и отзывчивости. WebView активирует доступ к Windows Runtime и использует текущий механизм визуализации Edge для отображения контента.

1. В**default.html** замените контент внутри тега **<body>** на элемент **WebView**. Установите его ID на**toastView**. ID облегчает выборку webview из вашего скрипта JavaScript.
   * 1. HTML
   1. <body class="win-type-body">

<x-ms-webview id="toastView"></x-ms-webview>

</body>

1. Откройте **default.js**. Укажите высоту и ширину элемента webview в **ActivationKind.Launch**.
   * 1. JavaScript
   1. if (args.detail.kind === activation.ActivationKind.launch) {

var webview = document.getElementById('toastView');

* 1. webview.style.height = "100%";
  2. webview.style.width = "100%";

1. Определеите класс **CommunicationWinRT** как объект JavaScript и используйте **addWebAllowedObject**, чтобы выявить и дать ему наименование в webview. Мы решили дать ему имя **toastApi** в рамках контекста webview.
   * 1. JavaScript
   1. if (args.detail.kind === activation.ActivationKind.launch) {

var webview = document.getElementById('toastView');

* 1. webview.style.height = "100%";
  2. webview.style.width = "100%";

var communicationWinRT = new CommunicationWinRT.CommunicationWinRT();

webview.addWebAllowedObject("toastApi", communicationWinRT);

1. Сохраните и закройте **default.js**.
   1. **Примечание:** Webview имеет доступ к API платформы Windows, так что вам не нужно задавать URI контента, чтобы получить доступ к WinRT в этом случае. Если вы захостите удалённый контент без webview, вам нужно будет определить URI контента и уровни доступа для этого контента в манифесте приложений. Более подробная информация о URI контента содержится в упражнении 2.

Задача 3 – Добавление новой HTML-страницы и скрипта для генерации всплывающих уведомлений

Ваше приложение сейчас настроено, чтобы захостить контент в вашем webview. В этой задаче вы добавите HTML-страницу, которая отобразится в webview, и запустите сценарий, чтобы сгенерировать всплывающее уведомление.

1. В проекте WebApp щёлкните правой кнопкой мыши и выберите **Add (Добавить) > New Item (Новый элемент)**. Выберите тип элемента **HTML Page (HTML-страница)** и присвойте ей имя **toastGenerator.html**. Добавьте веб-страницу в проект.

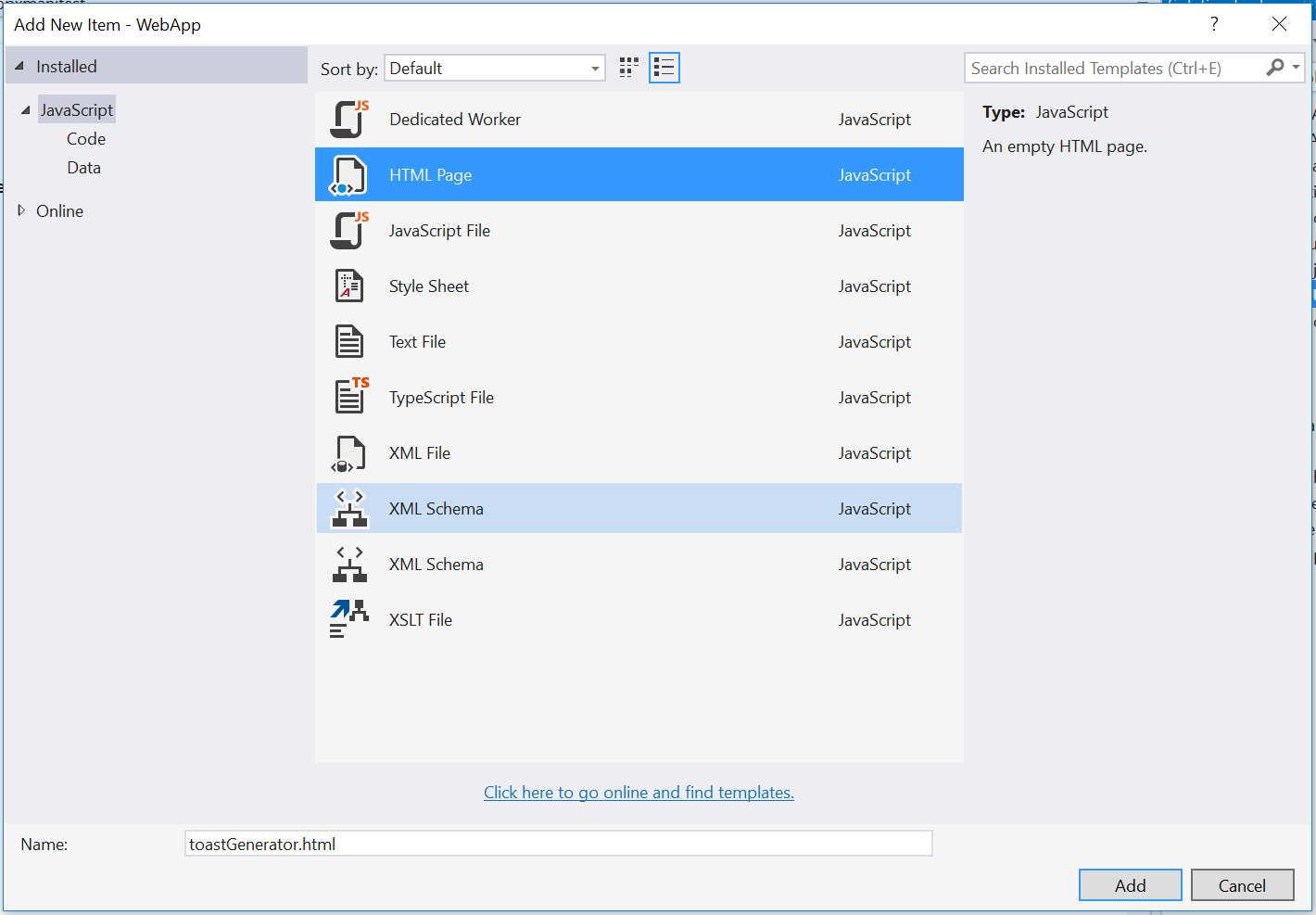


Рисунок 7

Добавление новой HTML-страницы в веб-проект.

1. В **На toastGenerator.html** добавьте заголовок веб-страницы в элемент **<title>**. Создайте кнопку в **<body>**, **нажатие** на которую будет вызывать функцию **sendToast()**. Вы создадите обработчик sendToast() на следующем этапе.
   * 1. HTML
   1. <!DOCTYPE html>  
      <html>  
      <head>  
          <title>Toast Integration</title>  
      </head>  
      <body>  
          <button onclick="sendToast()">Notify</button>  
      </body>  
      </html>
2. Добавьте обработчик **sendToast()** в тег **<script>**. Эта функция ищет объект **toastApi** и вызывает метод **toastMessage()** при его нахождении. Если методом toastApi не будет обнаружен объект, ваш лог покажет сообщение об ошибке.
   * 1. HTML
   1. <!DOCTYPE html>  
      <html>  
      <head>  
          <title>Toast Integration</title>  
          <script type="text/javascript">  
              function sendToast() {  
                  var object = window.toastApi;

            if (object) {  
                console.log(object + ' found');  
                object.toastMessage('Hello Toast', 0);  
            }  
            else {  
                console.log(object + " not found in the DOM.")  
            }  
        }  
    </script>  
</head>  
<body>  
    <button onclick="sendToast()">Notify</button>  
</body>  
</html>

1. Вернитесь к файлу **default.js**, чтобы перейти на веб-страницу **toastGenerator.html**.
   * 1. JavaScript
   1. webview.addWebAllowedObject("toastApi", communicationWinRT);

webview.navigate("ms-appx-web:///toastGenerator.html");

1. Настройте Текущую конфигурацию решения на **Debug (Отладка)** и Платформу решений в соответствии с **x86**. Создайте и запустите свое приложение на **Local Machine (Локальном компьютере).**

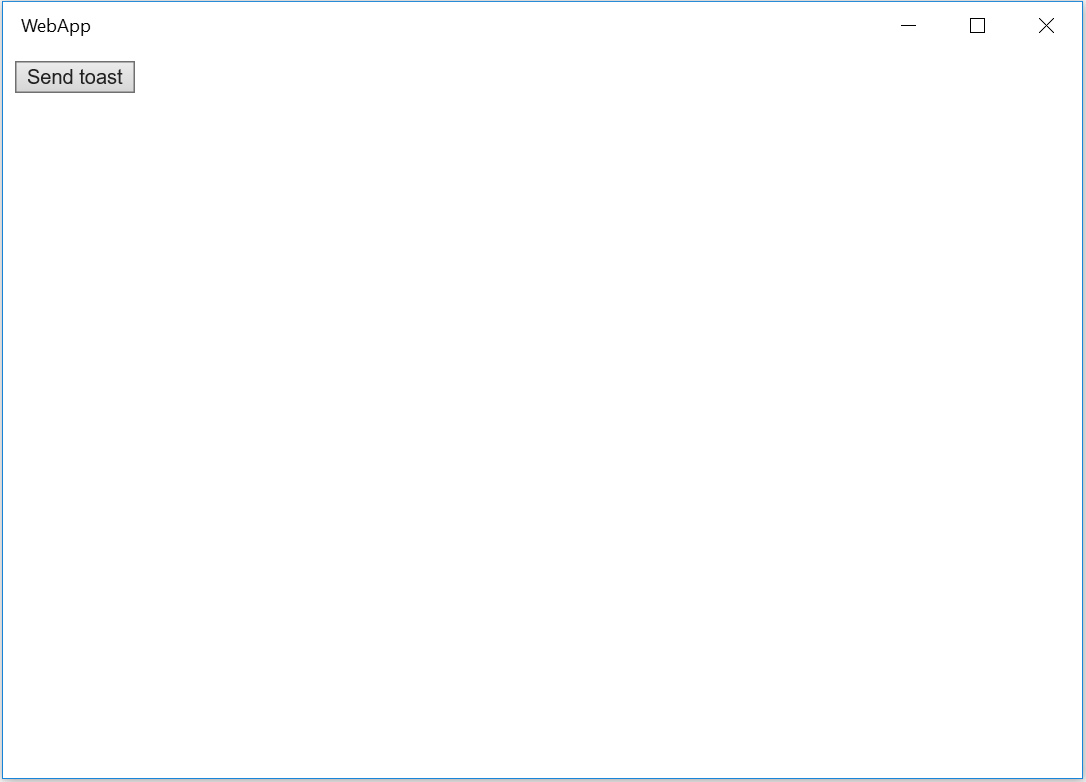


Рисунок 8

Веб-приложение отображает кнопку **Send toast (Отправить уведомление)**.

1. Используйте кнопку **Send toast (Отправить уведомление)** для генерации уведомления. Сообщение **Hello, Toast!** появится возле значка Центра поддержки в панели задач Windows.

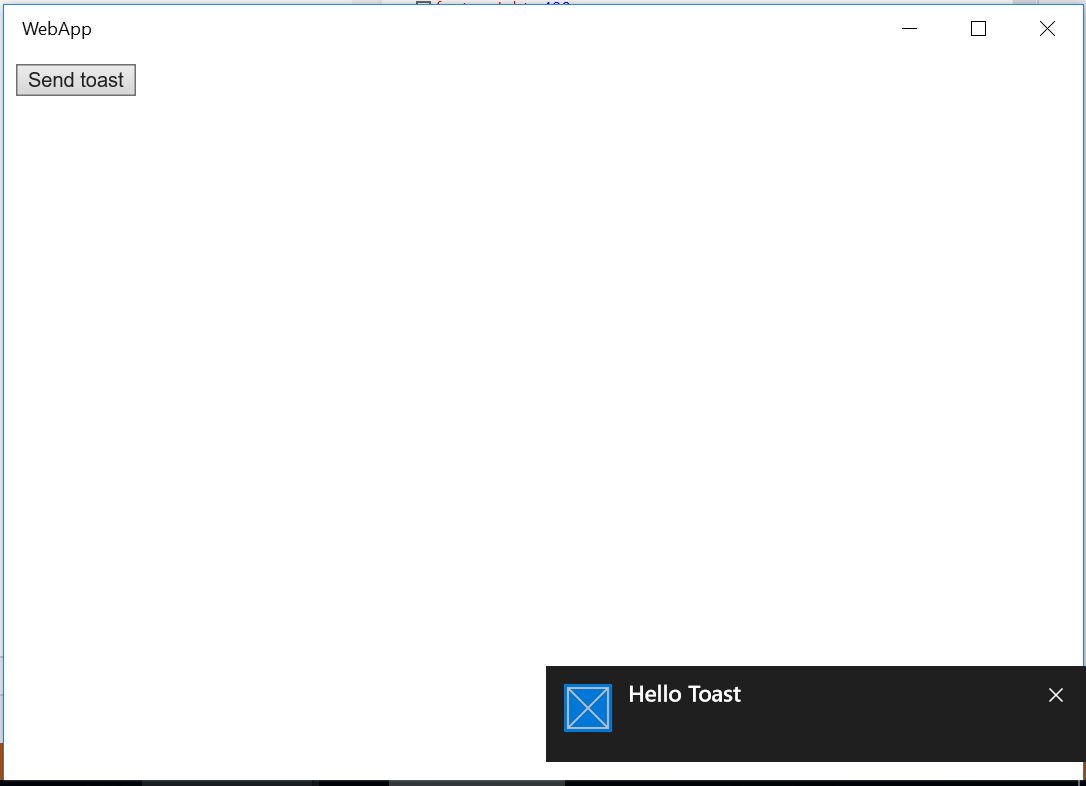


Рисунок 9

Веб-проект успешно интегрирован в аппаратный проект, чтобы сгенерировать всплывающее уведомление.

1. Отключите отладку и вернитесь к Visual Studio.

Упражнение 2: Создание размещённого на хосте веб-приложения

* 1. При наличии сайта с адаптивным дизайном онлайн, вы можете создать размещённое на хосте веб-приложение для Windows Store за несколько минут. По сути, размещённое на хосте веб-приложение работает, как браузер, использующий механизм визуализации Edge. В этом упражнении вы создадите проект веб-сайта MVC, опубликуете его на Azure, используете в качестве хоста в веб-приложение, и запустите всплывающее уведомление от размещённого на хосте контента.

Задача 1 – Создание веб-сайта на Azure

В этой задаче вы создадите проект веб-сайта MVC.

1. Щелкните правой кнопкой мыши на Solution Explorer (Обозреватель решений). Затем **Add (Добавить) > New Project (Новый проект)**. Выберите тип проекта **Visual C# > Web > ASP.NET Web Application**. Оставляйте опцию Application Insights не выбранной и присвойте ему имя **AzureWebApp**.

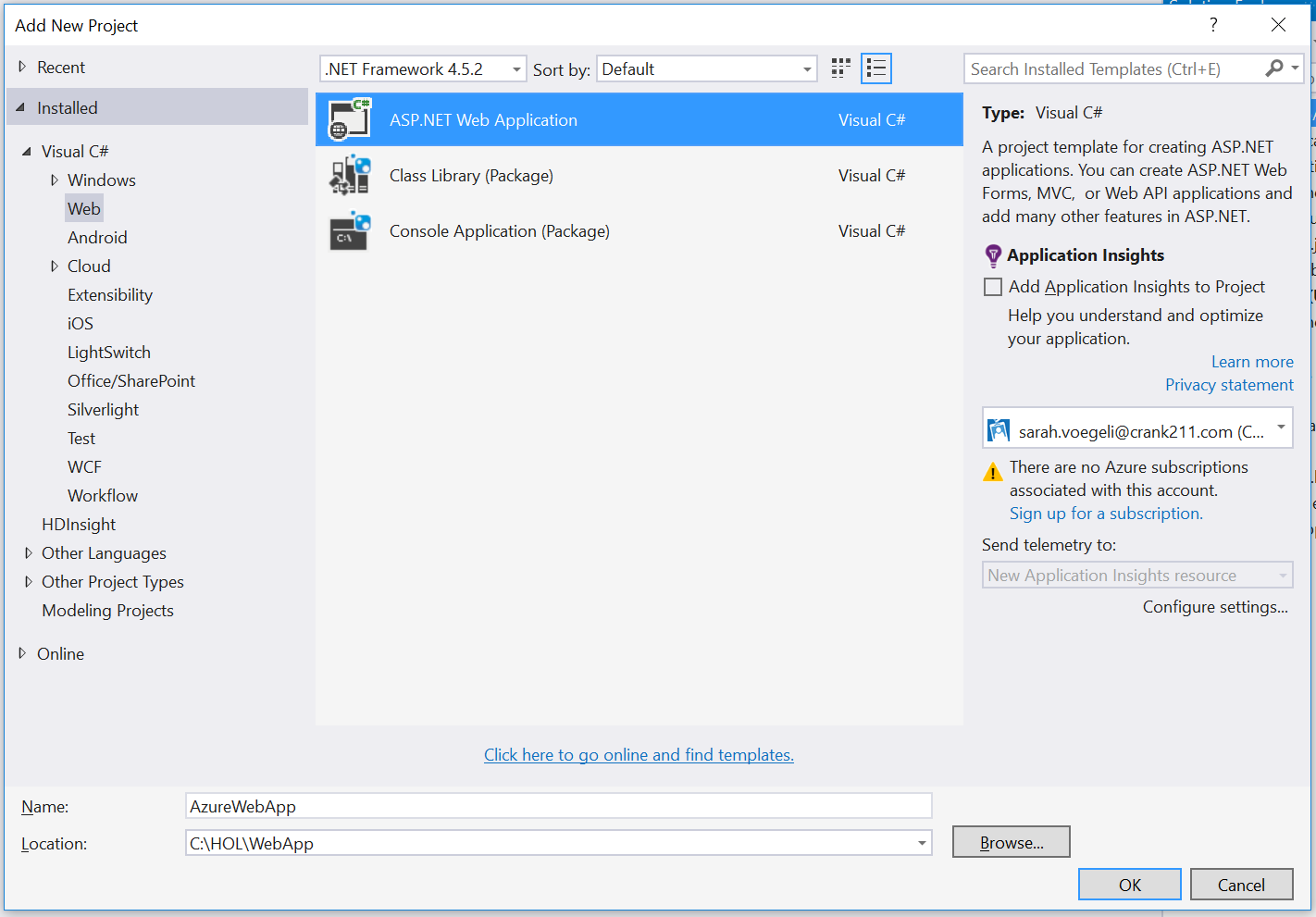


Рисунок 10

Добавление проекта веб-сайта.

1. Когда диалоговое окно **New ASP.NET Project (новый проект ASP.NET)**  появится на экране, выберите шаблон **MVC**. Поставьте галочку в блоке **Add folders and core references (Добавить папки и ссылки)** для **MVC**. Оставьте опции**Web Forms (Веб-Формы)**, **Web API(веб-API)**, **Add unit tests (Добавить тестирование элементов)** и **Host in the cloud (Хост в облаке)** неотмеченными. Вы опубликуете веб-сайт в Azure в следующей задаче.

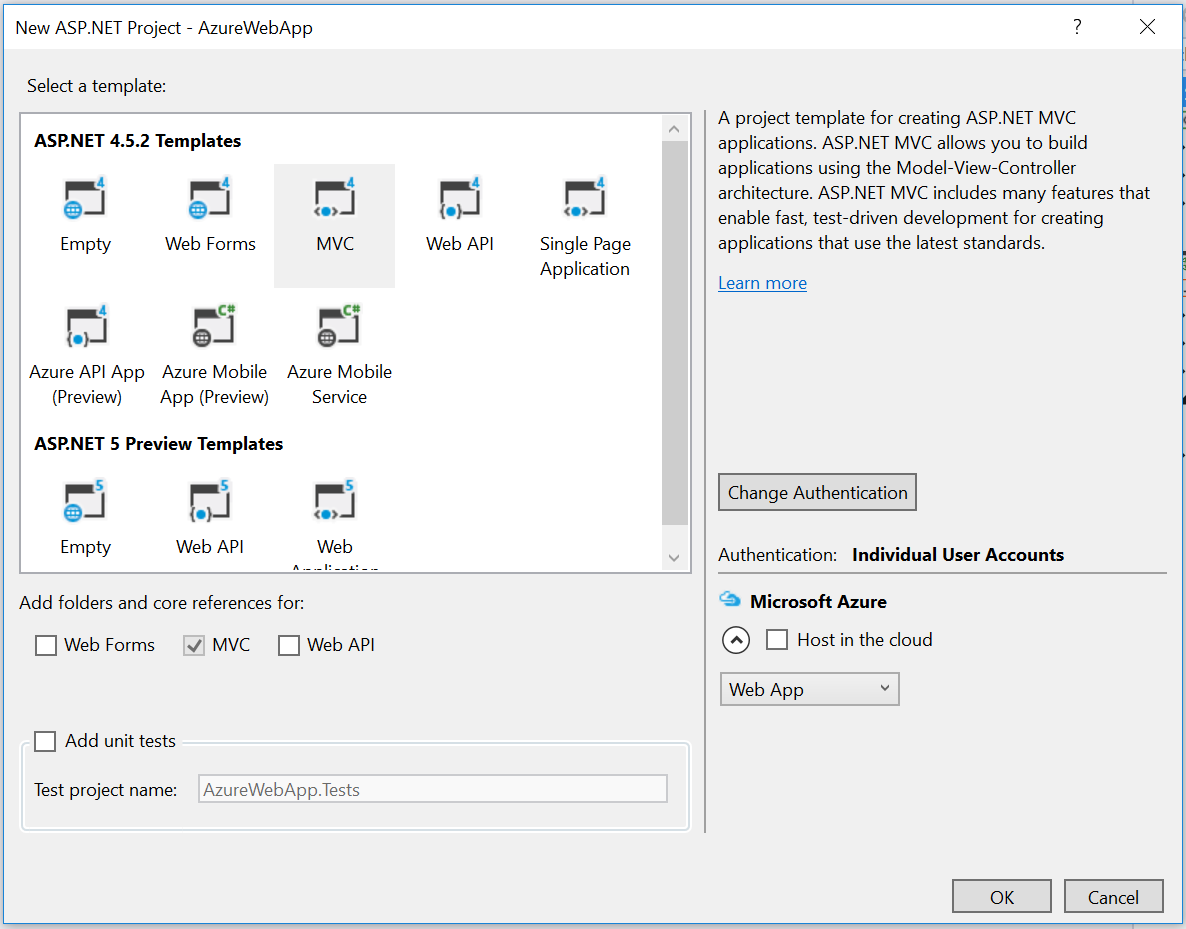


Рисунок 11

Выбор опций MVC.

1. Нажмите ОК для создания проекта.
2. Кликните правой кнопкой мыши на проете **AzureWebApp** в Solution Explorer (Обозреватель решений) и выберите **Set as StartUp project (Установить как проект для запуска).**
3. С помощью Solution Platform (Платформы решения), настроенной на значение **x86**, и Debug Target (Цель отладки) настроенной на **Internet Explorer**, создайте и запустите своё приложение. Вы увидите стартовую веб-страницу MVC, выполняющуюся на локальном хосте.

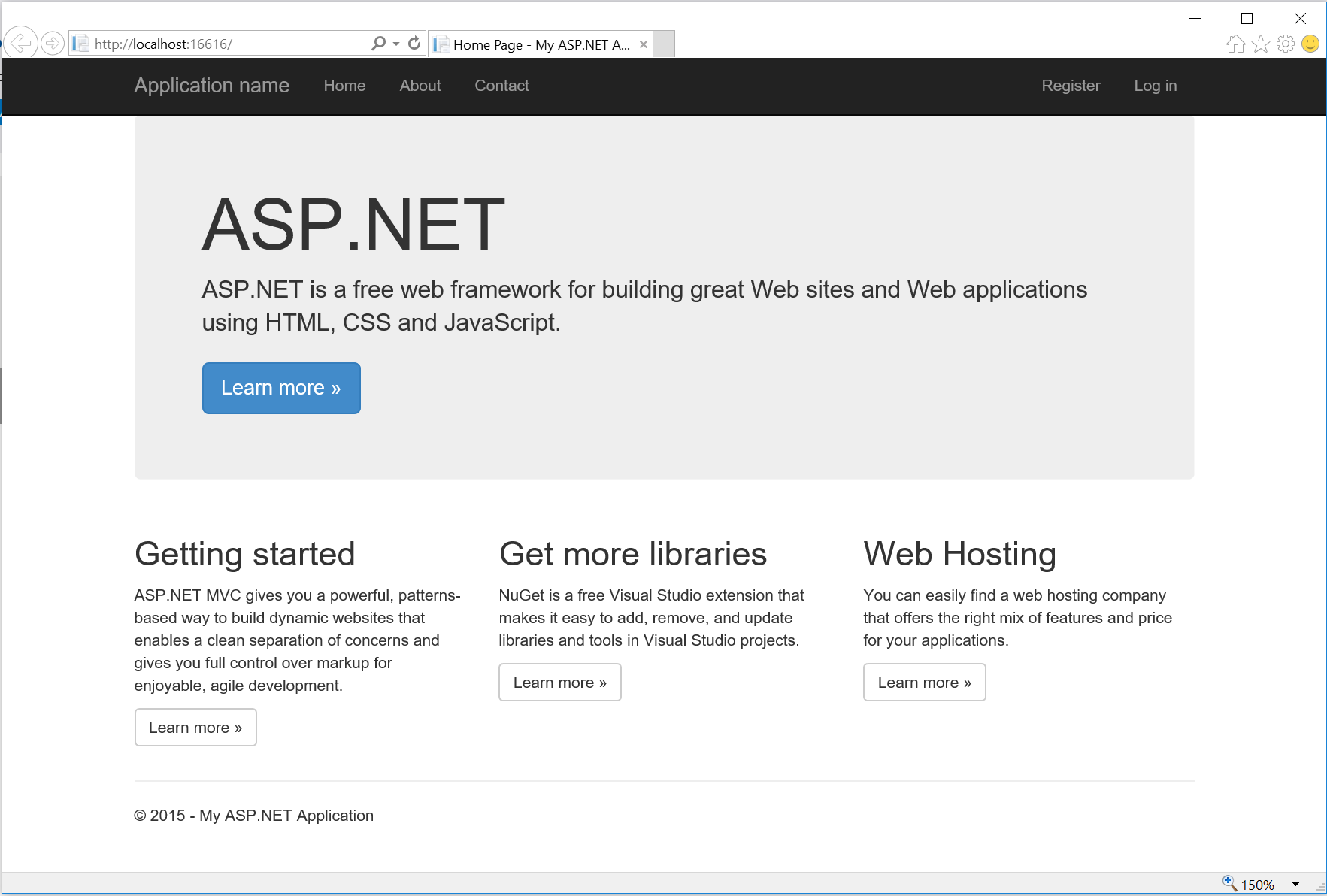


Рисунок 12

Стартовая веб-страница сайта MVC, выполняющаяся в Internet Explorer.

1. Отключите отладку и вернитесь к Visual Studio.

Задача 2 – Публикация на сайте Azure

Чтобы создать настоящее размещённое на хосте веб-приложение, нужно его опубликовать на сервере. В этой задаче вы будете выполнять шаги по публикации сайта по существующей подписке Azure.

* 1. **Примечание:** Если вы ещё не имеете подписки Azure, вы можете подписаться на бесплатную пробную 30-дневную версию по адресу<https://azure.microsoft.com/en-us/pricing/free-trial/>. Бесплатная пробная версия будет поддерживать демо-версию нашего приложения.
  2. Кроме того, если вам был выдан сетификат Azure Pass, вы можете подписаться на ограниченный по времени бесплатный период, используя промо-код Azure Pass по адресу <http://www.microsoftazurepass.com/>.

1. Кликните правой кнопкой мыши на проекте **AzureWebApp** в Solution Explorer (Обозреватель решений) и выберите **Publish (Опубликовать)**.
2. При появлении диалогового окна публикации, нужно выбрать **Microsoft Azure Web Apps** в качестве цели публикации.

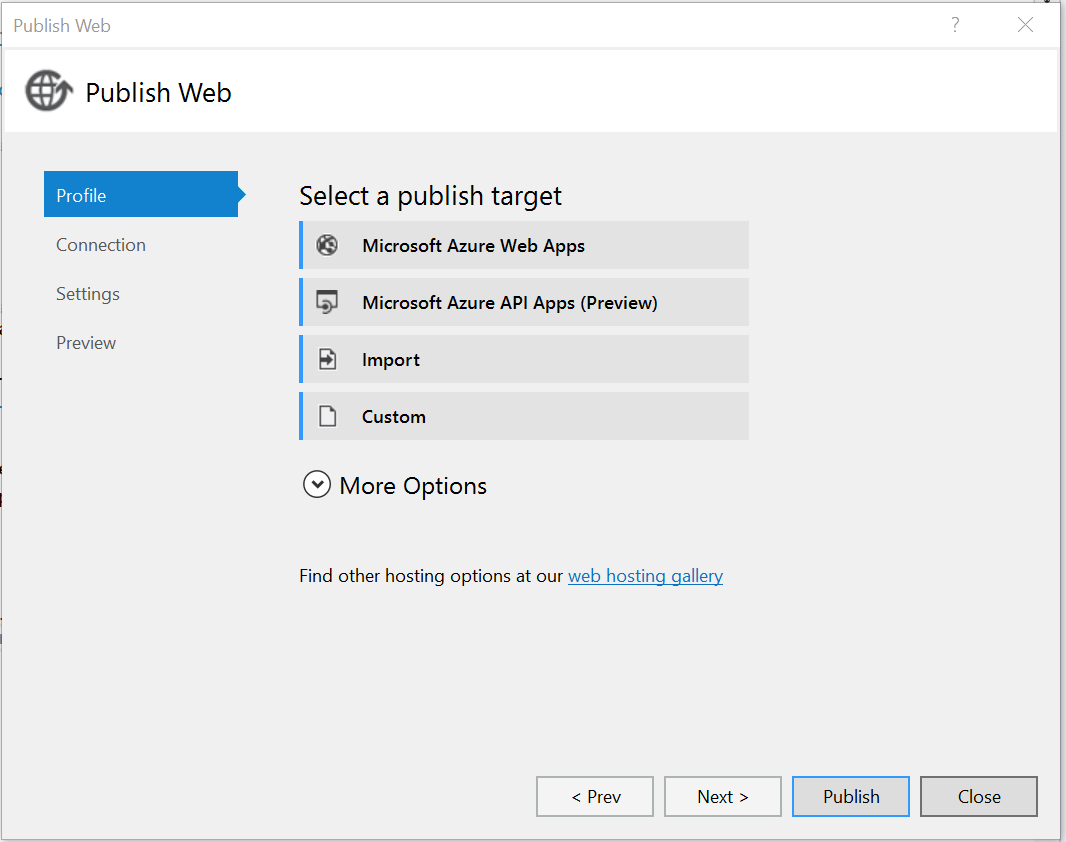


Рисунок 13

Диалоговое окно публикации.

1. Откроется диалоговое окно для подключения вашей учётной записи Azure. Используйте выпадающее меню, чтобы выбрать или добавить учётную запись, которую вы использовали для входа на Azure. Повторно введите свои параметры для входа в аккаунт.

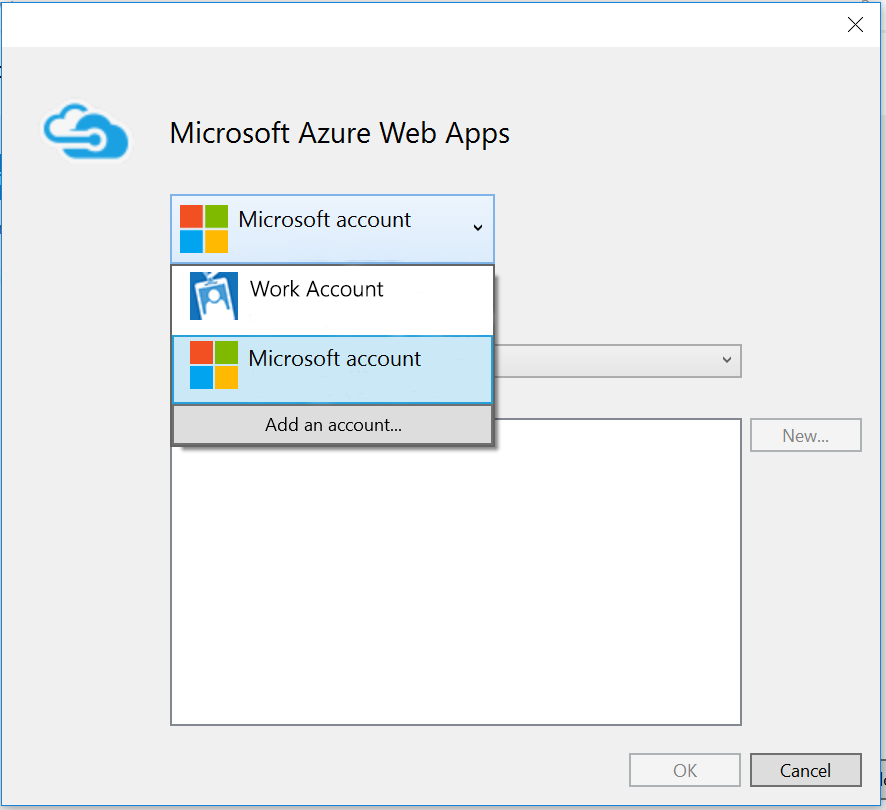


Рисунок 14

Вход на Azure.

* 1. **Примечание:** Если вы вошли на Azure, однако сообщение в диалоговом окне **This account has no subscriptions (Эта учетная запись не имеет подписки)** сохраняется, возможно, причина в том, что у вас имеется несколько учётных записей Microsoft. Учётные записи Microsoft и учётные записи Microsoft Work/School имеют отличия, даже если вы создали их с одним и тем же адресом электронной почты.
  2. Проверьте, какая учётная запись связана с вашей подпиской на Azure, и добавьте её параметры для входа, чтобы продолжить.

1. После входа на Azure, используйте выпадающее меню Subscription (Подписка), чтобы выбрать подписку, которую вы хотели бы использовать. Если вы используете бесплатную пробную версию, она будет единственной доступной подпиской.

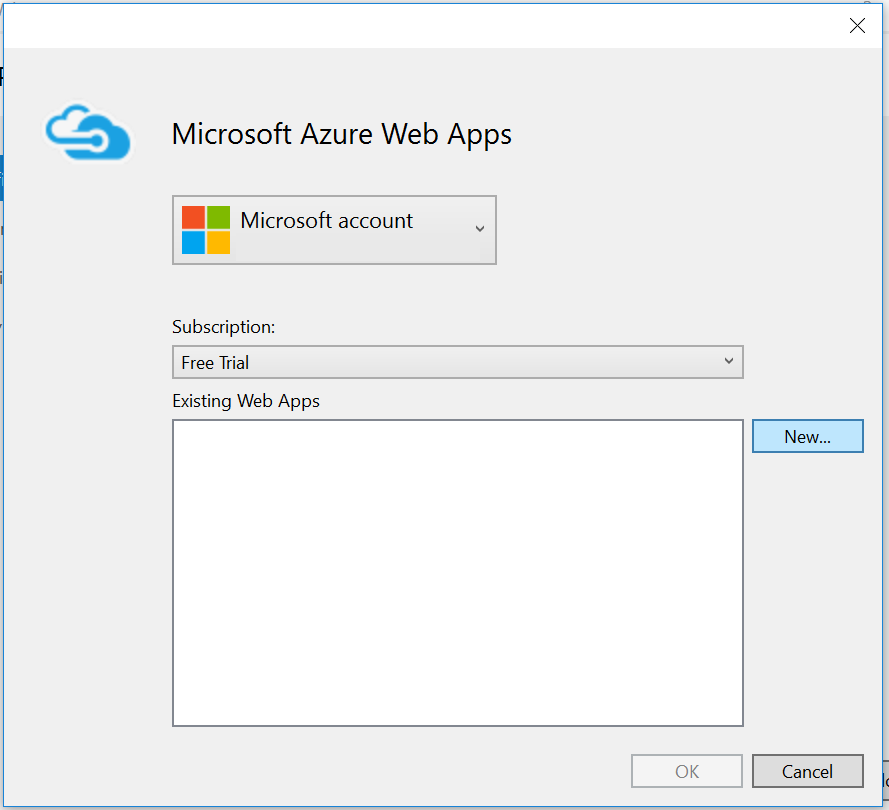


Рисунок 15

Выбор подписки.

1. Находясь в диалоговом окне Microsoft Azure Web Apps, воспользуйтесь кнопкой **New (Новое)**, чтобы добавить веб-приложение в свою учётную запись.

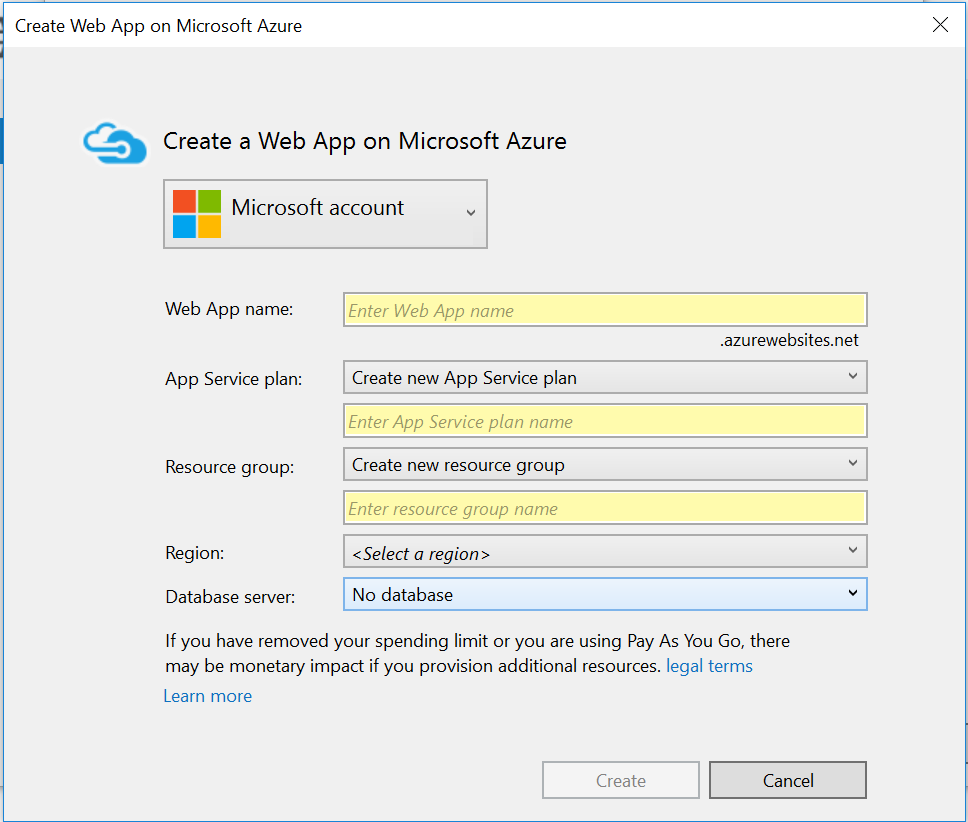


Рисунок 16

Настройка веб-приложения на Azure.

1. Присвойте веб-приложению уникальное имя. Ваше приложение будет использовать в качестве хоста azurewebsites.net. Зелёная галочка появится, если выбранное вами имя будет свободно.
2. Используйте выпадающие меню, чтобы выбрать или создать **App Service plan (Сервис-план приложения)** и **Resource group (Группа ресурсов)**. Выберите ближайший к вам регион из выпадающего списка **Region (Регион)**.
3. Оставьте для опции **Data server (Сервер данных)** значение **No database (Нет базы данных)**. Нажмите **Create (Создать)**, чтобы добавить своё веб-приложение. Может потребоваться несколько минут на завершение процесса.
4. После того, как вы добавите веб-приложение, вы возвратитесь к диалоговому окну Publish (Опубликовать). Оставьте метод публикации, установленный как Web Deploy, и сохраните все настройки по умолчанию на вкладках **Connection (Соединение)**, **Settings (Настройки)** и **Preview (Предпросмотр)**.

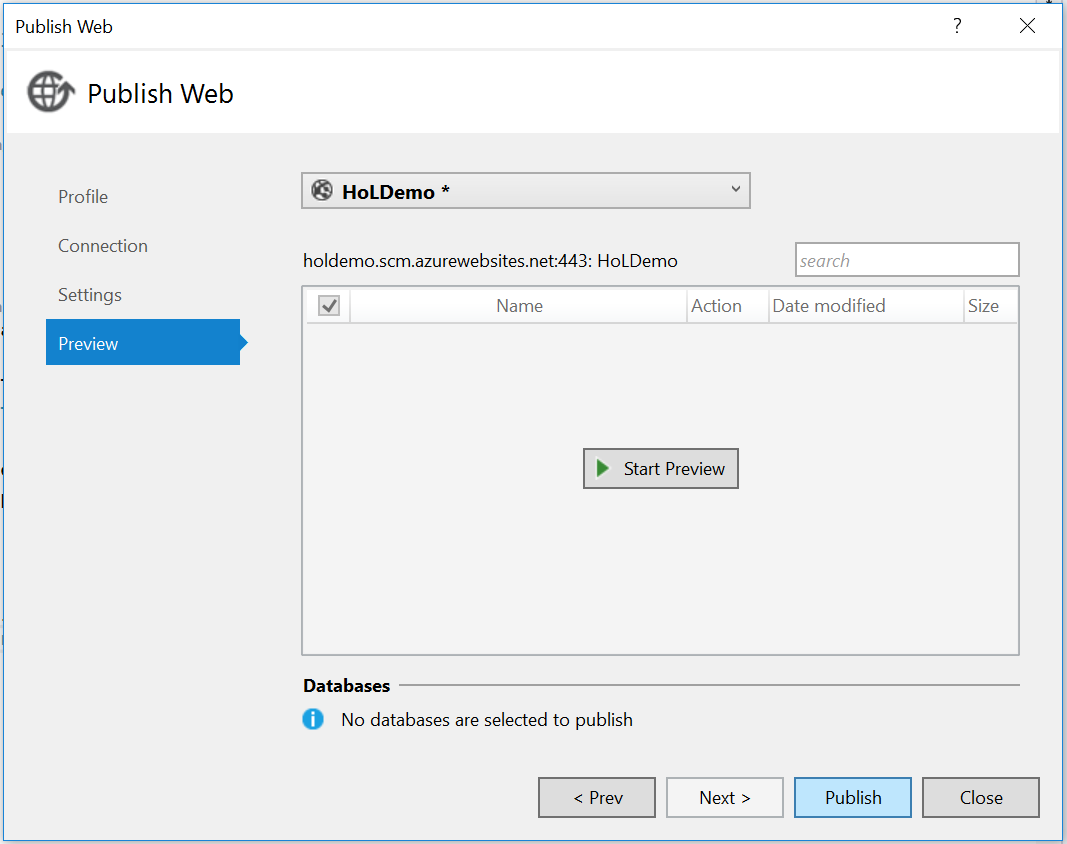


Рисунок 17

Публикация веб-приложения.

1. Используйте кнопку **Publish (Опубликовать)**, чтобы опубликовать своё веб-приложение.
2. Ваше веб-приложение откроется в браузере с URI на **azurewebsites.net**. Измените размер окна браузера, чтобы оценить адаптивность веб-приложения.

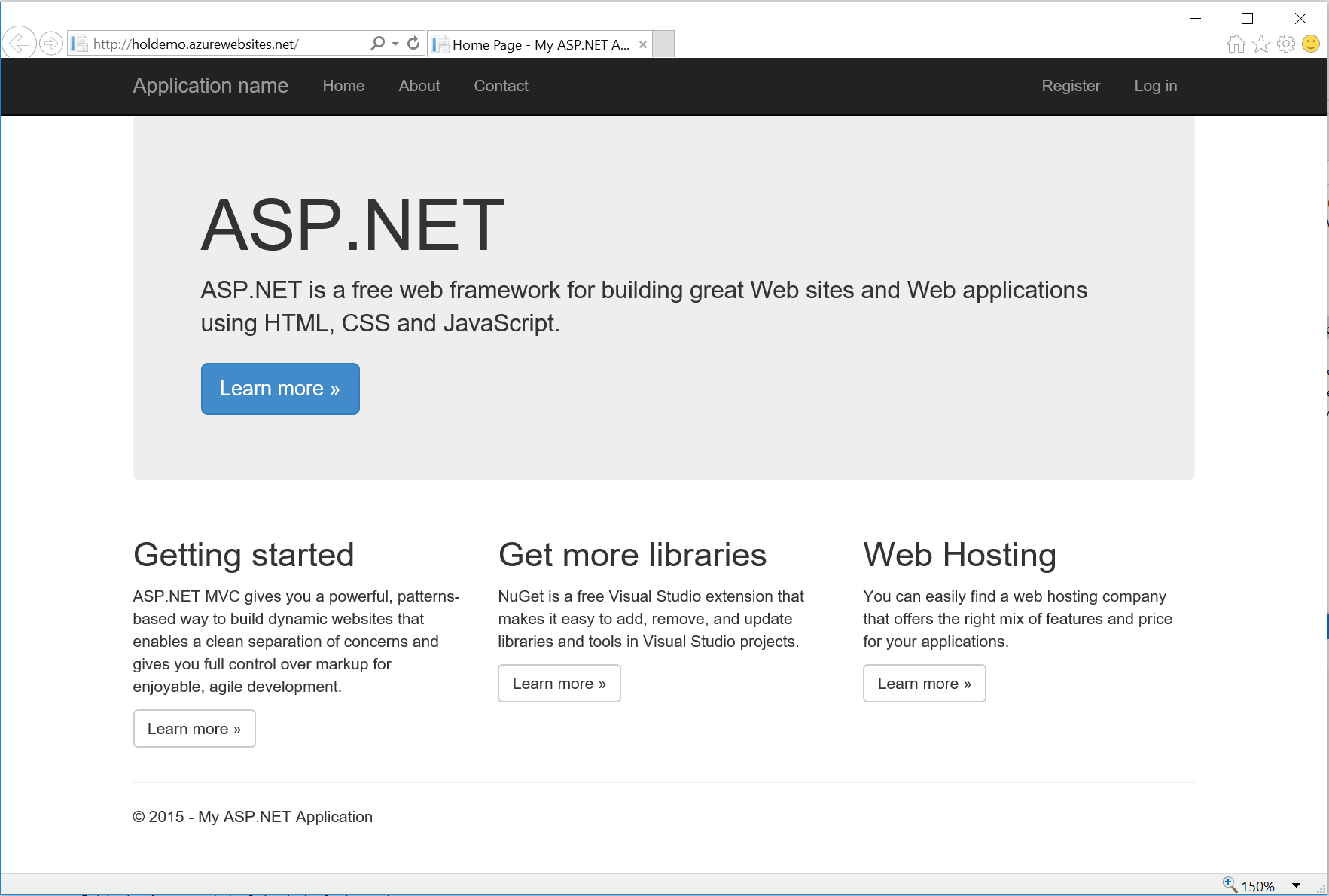


Рисунок 18

Опубликованное веб-приложение.

1. Закройте браузер и вернитесь в Visual Studio.

Задача 3 – Хост для веб-приложения

На данный момент вы имеете динамический адаптивный сайт, который готов к использованию приложения в качестве хоста. В этой задаче вы настроите проект размещённого на хосте веб-приложения, чтобы захостить сайт **AzureWebApp**.

1. Щелкните правой кнопкой мыши на Solution Explorer (Обозреватель решений). Затем **Add (Добавить) > New Project (Новый проект)**. Далее выберите **Installed (Установленное) > Templates (Шаблоны) > JavaScript** а затем выберите шаблон **Blank App приложения (Universal Windows).**
2. Назовите проект **HostedWebApp** и добавьте его в решение.

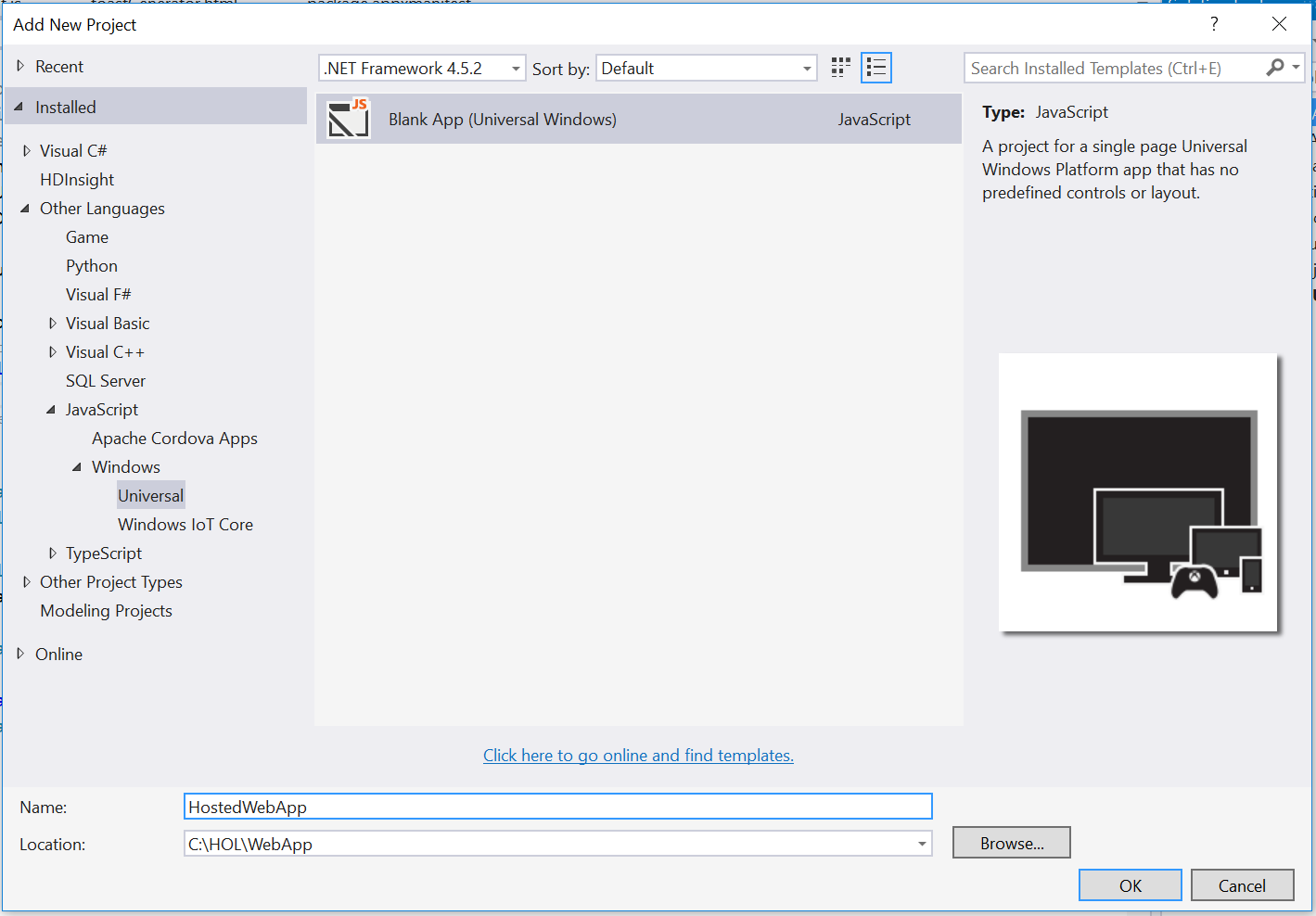


Рисунок 19

Создание проекта HostedWebApp в Visual Studio 2015.

1. Удалите папки **css**, **js** и **WinJS** и файл **default.html** из проекта **HostedWebApp**.
   1. **Примечание:** Когда вы создаёте веб-приложение, которое содержит только размещённое на хосте содержание, вы можете удалить папки **css**, **js** и **WinJS**, а также файл **default.html**. Мы удалили эти файлы, но вы могли бы сохранить их и использовать, чтобы реализовать автономную веб-страницу посадки для своего приложения.
2. Откройте **Package.appxmanifest** в редакторе манифестов.
3. На вкладке **Application (Приложение)** замените **Start page (Стартовая страница)** на URI своего сайта Azure.

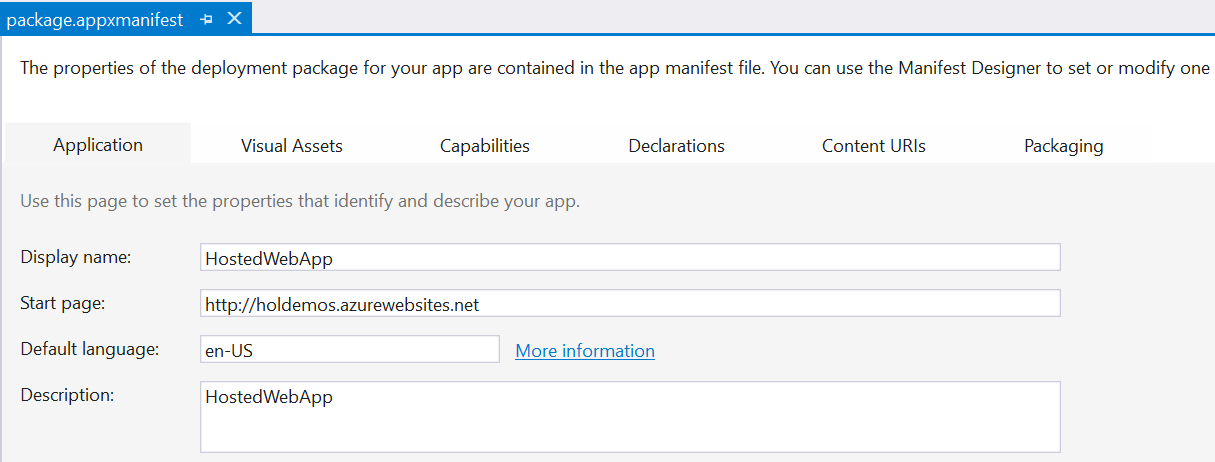


Рисунок 20

Установка вашего сайта на Azure в качестве стартовой страницы.

1. Откройте вкладку **Content URIs (URI контента)**. Добавьте URI вашего веб-сайта Azure как URI контента. Установил значение **Rule (Правило)** на **Include (Включать)** и **WinRT access (Доступ WinRT)** на **All (Все)**. Вы будете использовать доступ к WinRT в следующей задаче.

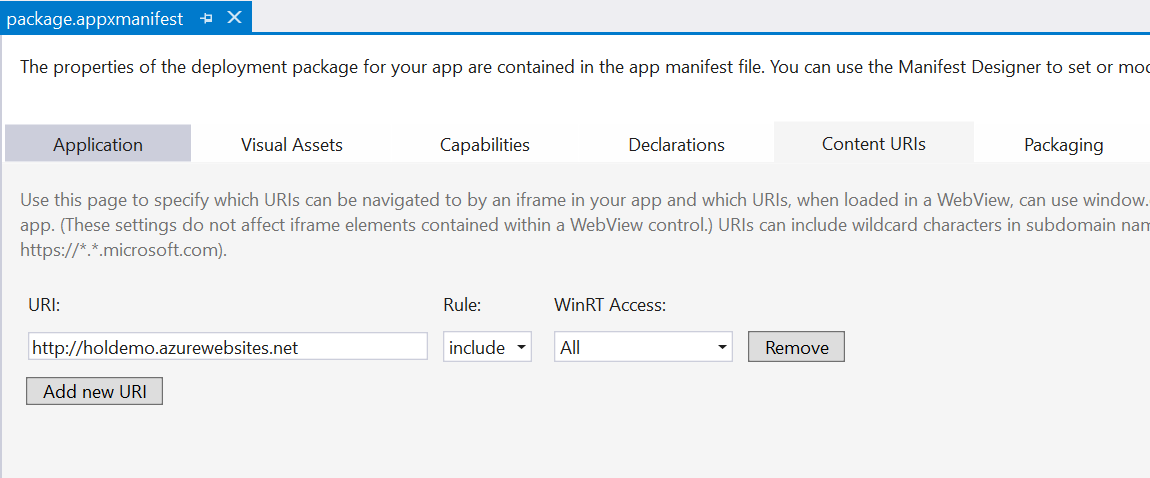


Рисунок 21

Добавление URI контента для вашего сайта Azure и предоставление ему доступа к WinRT.

* 1. **Примечание:** Правилах URI для контента приложения (ACURS) для вашего приложения определяют веб-страницы, которые могут использоваться приложением. Например, вы можете решить, что вашим пользователям можно просматривать ваш веб-сайт в приложении, но все внешние ссылки будут открываться в браузере. Эти вставки и исключения позволяют вам контролировать границы своего приложения и помешают ему вести себя подобно стандартному веб-браузеру. URI контента также дают вам возможность активировать доступ к Windows Runtime или отключать доступ для различных частей приложения и определять, кому может быть предоставлен доступ: **None (Никто)**, **All (Все)** или **Allow for web only (Только для веб)**.
  2. Чтобы указать удалённый URI, используйте протокол **http://**. Чтобы указать локальный URI, используйте протокол **ms-appx-web:///**.

1. Сохраните и закройте манифест. Кликните правой кнопкой мыши на проекте **HostedWebApp** в Solution Explorer (Обозреватель решений) и выберите **Set as StartUp project (Установить как проект для запуска)**.
2. Создайте и запустите приложение **HostedWebApp** на **локальном компьютере**. Вы увидите сайт Azure в окне приложения. Размещённое на хосте веб-приложение получает динамичный контент с сервера Azure, так что вы будете видеть самую последнюю версию сайта каждый раз, когда вы запускаете приложение. Измените размер окна приложения, чтобы посмотреть, как соответствующая схема сайта отображается в приложение.

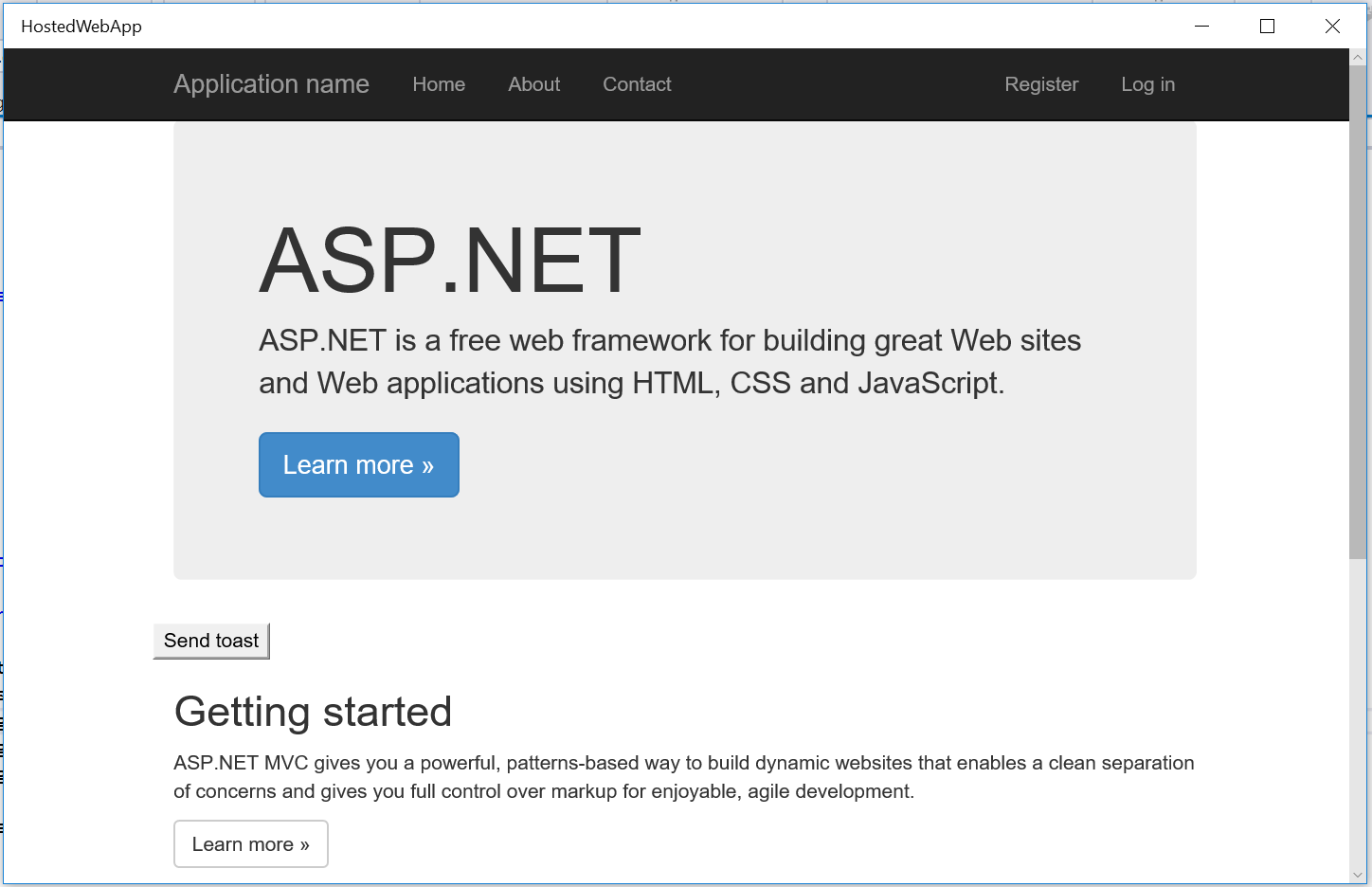


Рисунок 22

Сайт Azure в приложении.

1. Отключите отладку и вернитесь к Visual Studio.

Задача 4 – Отправление всплывающего уведомления

Сейчас, когда у нас есть размещённое на хосте работающее веб-приложение, давайте настроим отправку всплывающего уведомления, когда оно обнаруживает, что использует содержание размещённое на хосте приложения.

1. Вернитесь к проекту **AzureWebApp**. Щелкните правой кнопкой мыши на папке **Scripts (Скрипты)** и выберите **Add (Добавить) > JavaScript file (Файл JavaScript)**. Присвойте ему имя **notifications.js**.
2. Откройте **App\_Start > BundleConfig.cs**. Добавьте **notifications.js** к набору, содержащему bootstrap.js и respond.js. Не забудьте добавить запятую после предыдущего скрипта в списке.
   * 1. C#
   1. bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
   2. "~/Scripts/bootstrap.js",
   3. "~/Scripts/respond.js",
   4. "~/Scripts/notifications.js"));
3. Откройте notifications.js и скопируйте код в следующие функции, чтобы создать всплывающее уведомление.
   1. **Примечание:** Условная функция **if (window.Windows** определяет, доступна ли API. Этот код выполняется только внутри размещённого на хосте веб-приложения. Вы дали размещённому на хосте веб-приложению право использовать WinRT в URI контента в предыдущей задаче.
      1. JavaScript
   2. if (window.Windows) {
   3. console.log('this printed from the file')
   4. function createToast(title, message, imgUrl, imgAlt, tag) {
   5. console.log('create toast triggered')
   6. // Namespace: Windows.UI.Notifications
   7. if (typeof Windows !== 'undefined' &&
   8. typeof Windows.UI !== 'undefined' &&
   9. typeof Windows.UI.Notifications !== 'undefined') {
   10. // Установка переменных для быстрого доступа
   11. var notifications = Windows.UI.Notifications,
   12. templateType = notifications.ToastTemplateType.toastImageAndText02,
   13. templateContent = notifications.ToastNotificationManager.getTemplateContent(templateType),
   14. toastMessage = templateContent.getElementsByTagName('text'),
   15. toastImage = templateContent.getElementsByTagName('image'),
   16. toastElement = templateContent.selectSingleNode('/toast');
   17. var launchParams = {
   18. type: 'toast',
   19. id: tag || 'demoToast',
   20. heading: title || 'Demo title',
   21. body: message || 'Demo message'
   22. };
   23. var launchString = JSON.stringify(launchParams);
   24. // Настройка сообщения и изображения для щаблона уведомления
   25. toastMessage[0].appendChild(templateContent.createTextNode(message || 'Demo message'));
   26. toastImage[0].setAttribute('src', imgUrl || 'https://unsplash.it/150/?random');
   27. toastImage[0].setAttribute('alt', imgAlt || 'Random sample image');
   28. toastElement.setAttribute('duration', 'long');
   29. toastElement.setAttribute('launch', launchString); // Optional Launch Parameter
   30. // Show the toast
   31. var toast = new notifications.ToastNotification(templateContent);
   32. var toastNotifier = new notifications.ToastNotificationManager.createToastNotifier();
   33. toast.tag = 'demoToast';
   34. console.log(toast);
   35. toastNotifier.show(toast);
   36. } else {
   37. var alertText = title || 'Demo Title';
   38. alert(alertText);
   39. }
   40. }

function notify() {

* 1. var title, message, imgUrl, imgAlt, tag
  2. title = 'Alert'
  3. message = "Hello from your hosted web app"
  4. imgUrl = http://assets.onestore.ms/cdnfiles/onestorerolling-1509-27004/shell/v2\_5/images/logo/icon-windows-logo.png';
  5. imgAlt = 'this is image alt'
  6. tag = 'tag'
  7. createToast(title, message, imgUrl, imgAlt, tag);
  8. }
  9. }

1. Откройте **Views (Виды) -> Домашняя страница (Home) -> Index.cshtml**. Добавьте кнопку для отправки всплывающего уведомления, и настройте событие onclick для функции notify().
   * 1. CSHTML
   1. <div class="row">
   2. <div class="col-md-4">
   3. <button onclick="notify()">Send toast</button>
   4. <h2>Getting started</h2>
2. Кликните правой кнопкой мыши на проекте **AzureWebApp** в Solution Explorer (Обозреватель решений) и выберите **Publish (Опубликовать)**. На этот раз вам не нужно будет проходить процесс настройки. Повторно введите свои параметры для входа в аккаунт.
   1. **Примечание:** Вы увидите кнопку **Send toast (Отправить уведомление)** в браузере, когда он откроется после публикации. Нажатие на кнопку не выполняет никакого действия, так так **if (window.Windows)** возвращает значение false, когда веб-сайт отображается в браузере вместо приложения UWP. В идеале, кнопка должна быть скрыта, когда сайт выполняется вне размещённого на хосте веб-приложения.
3. Создайте и запустите приложение **HostedWebApp** на локальном компьютере. Теперь на стартовой странице отображается кнопка **Send toast (Отправить уведомление)**. Кликните по кнопке, чтобы отправить уведомление.
   1. **Примечание:** Кликните по кнопке Send toast (Отправить уведомление) только один раз. Если бы вы захотите отправить уведомление повторно, откройте Центр Уведомлений и удалите предыдущее уведомление.
   2. Для поиска проблем с уведомлением, откройте консоль JavaScript в Visual Studio при работающем приложении. Кликните по кнопке **Send toast (Отправить уведомление)**. При срабатывании функции **createToast** вы увидите сообщение в журнале **create toast triggered**.
4. Отключите отладку и вернитесь к **Visual Studio**.

Упражнение 3: Поддержка дополнительных платформы и устройств с помощью ManifoldJS (дополнительно)

* 1. Размещённые на хосте веб-приложения являются отличным способом быстро расширить ваши веб-проекты на новые платформы. ManifoldJS - это инструмент, который использует существующие метаданные с вашего веб-сайта для генерации размещённых на хосте приложений для множества платформ, включая iOS, Android, Windows 10, Chrome OS и Firefox OS. Для платформ, которые не поддерживают размещённые на хосте веб-приложения по умолчанию, ManifoldJS использует Cordova.
  2. Манифест, генерируемые ManifoldJS, отвечает стандарту W3C для манифестов веб-приложений и включает метаданные, такие как стартовая страница для сайта, белый список URL, название сайта, цветовая схема и изображения.

**Примечание:** Последние изменения в ManifoldJS см. по ссылке <http://www.manifoldjs.com/>. Дополнительную информацию о манифестве W3C для веб-приложений см. по ссылке <https://w3c.github.io/manifest/>.

Задача 1 – Установка ManifoldJS и создание манифеста

Установите ManifoldJS для генерации манифеста.

1. Откройте командную строку с правами Администратора. С помощью установленного менеджер пакетов, используйте команду **npm install –g manifoldjs**, чтобы установить ManifoldJS глобально на своем компьютере для разработки.
   * 1. Command Prompt
   1. > npm install –g manifoldjs
   2. **Примечание:** Посетите сайт<https://nodejs.org/>, чтобы загрузить и менеджер пакетов Node Package Manager (npm).
2. Сгенерируйте манифест для своего сайта на <http://www.manifoldjs.com/generator>. Вы также можете загрузить свой манифест на сайт, и специальный инструмент проверит и предупредит вас о любых пробелах, которые он обнаружит.
   1. **Примечание:** Если ваш сайт не будет иметь манифеста, ManifoldJS сгенерирует его для вас. Однако, вы ещё можете создать свой собственный манифест, чтобы указать данные своего сайта и предоставить изображения приложений.
3. Загрузите манифест в корневой каталог своего сайта на сервер. Манифест обычно хранится в той же директории, что и файл index.html.

Задача 2 – Генерация размещённого на хосте веб-приложения

В этой задаче вы сгенерируете размещённые на хосте веб-приложения из своего веб-сайта для множества платформ.

1. Вернитесь на локальный компьютер. Создайте папку для хранения своих размещённых на хосте веб-приложений и перемещайтесь в неё с помощью командной строки. Передайте URL работающего веб-сайта в manifoldjs, чтобы сгенерировать манифест. Мы будем использовать для этого Bing. Вы также можете использовать опции **-l отладки** для получения более информативного результата.
   * 1. Command Prompt
   1. > manifoldjs http://www.bing.com/
2. Исследуйте код, сгенерированный ManifoldJS, в папке размещённых на хосте приложений.
3. Чтобы установить и запустить сгенерированное приложение под Windows10, выполните следующую команду в папке приложения, созданной ManifoldJS:
   * 1. Command Prompt
   1. > manifoldjs run windows
4. Сгенерированное вами приложение установится и запустится.

Summary

* 1. Размещённые на хосте веб-приложения и веб-виды предоставляют мощные возможности, чтобы интегрировать существующие веб-проекты в Windows Store и API платформы. В этом курсе мы создали веб-приложение, чтобы захостить локальный и удалённый контент, и научились интегрировать API Windows с помощью JavaScript. Мы также изучили возможности генерации размещённых на хосте веб-приложений для различных платформ.