Microsoft Azure

WEB APP and SQL DB

Hyun Suk Shin (AZURE CSA)

2017

웹 앱과 DB 개발 및 배포하기

본 실습을 통해 웹 앱 개발 플랫폼에서 웹 앱을 개발 배포 합니다. 아울러 웹 앱과 연동되는 SQL DB를 개발 배포합니다. 플랫폼 서비스가 제공하는 장점 (개발 플랫폼 장점과 확장성)을 학습합니다.

1. 실습을 위해서 사전에 계정과 구독 그리고 실습을 위한 관련 도구(tool)을 다운로드 받습니다.
   1. Azure 계정과 구독
      1. Azure에 접속하기 위해서는 Microsoft 계정이 필요합니다. Microsoft 계정이 없는 경우 <http://msn.com>에서 새로 생성할 수 있습니다.
         1. <http://msn.com> 을 방문하여 오른쪽 상단에 ‘로그인’을 클릭하고 ‘계정을 만드세요’를 클릭합니다.

|  |  |
| --- | --- |
|  |  |

* + - 1. 계정을 생성 할 때 아래와 이미 존재하는 회사 메일 주소를 사용할 수 있습니다.

|  |
| --- |
|  |

* + 1. 제공된 Azure Free Pass를 사용하여 무료 구독을 생성합니다. Free Pass는 30일 동안 제공된 금액 미만으로 사용이 가능하며 제공된 비용을 초과하게되는 경우 또는 30일이 경과하는 경우는 자동으로 중지 됩니다.
  1. DevTest Labs에 Visual Studio Community Edition이 설치된 개발용 가상 컴퓨터를 구성합니다.
     1. 구체적인 방법은 문서 “7. Devtest labs for WebDev.docx”을 참고합니다.
  2. 실습 비용 한도에 따른 유의 사항
     1. 실습을 종료 하고나서 가급적 사용하지 않는 서비스들은 삭제합니다. 불필요한 비용 발생으로 무료 구독이 중지되면 실습 진행이 어렵습니다.
     2. 고사양의 가상 컴퓨터나 서비스를 생성하지 않습니다. 고사양 서비스들은 고비용이 청구되어 단시간에 제공된 무료 구독의 비용을 초과할 수 있습니다.

1. 실습을 위해서는 Windows 가상 컴퓨터, Visual Studio 2015 그리고 Azure SDK 설치가 필수 입니다. .
   1. Visual Studio 2015를 실행합니다. (구체적인 방법은 “7. Devtest labs for WebDev.docx”을 참고합니다..

|  |
| --- |
|  |

1. 새 웹 앱 프로젝트를 생성합니다.
   1. Azure Portal에 로그인합니다. [+ NEW]를 클릭합니다.

|  |
| --- |
|  |

* 1. “New Project” 창에서 “ASP.NET Web Application (.NET Framework)”를 선택합니다. 우측에 Application Insights 항목에 “Add….” 체크를 해지 하도록 합니다.. Project의 이름은 myboard로 명명하고 [OK]를 클릭하여 다음으로 넘어갑니다..

|  |
| --- |
|  |

* 1. “New ASP.NET Web Application - myboard” 창에서 MVC를 선택합니다. 그리고 [OK]를 클릭하여 다음으로 넘어 갑니다..

|  |
| --- |
|  |

* 1. “Create App Service” 창에서는 본 실습에서 개발할 웹 앱을 Azure에 배포할 때 사용할 정보입니다. 본 양식을 작성하면 개발 중간 중간 개발하고 나서 쉽고 빠르게 Azure로 웹 앱을 배포 할 수 있습니다. .

|  |
| --- |
|  |

* 1. 웹 앱의 이름을 명명합니다. 해당 이름은 DNS로 사용되기 때문에 가급적 인터넷에서 유일하게 사용할 수 있는 이름으로 명명합니다. 영문과 숫자를 조합합하여 명명하는 것을 권장합니다.

|  |
| --- |
|  |

* 1. 웹 앱을 배포할 리소스 그룹이 없습니다. 이름을 명명하도록 합니다..

|  |
| --- |
|  |

* 1. “App Service Plan”이 없습니다. [New]를 클릭하여 새로 생성합니다..

|  |
| --- |
|  |

* + 1. [New]를 클릭하면 아래 화면 처럼 새로운 App Service Plan 옵션을 선택할 수 있습니다.
       1. App Service Plan은 Plan을 구분하기 위한 이름입니다.
       2. Location은 서비스가 운영될 지역을 의미 합니다. 본 실습에서는 Japan West를 선택합니다.
       3. Szie는 웹 서비스의 하드웨어 성능을 의미 합니다. 이번 실습에서는 B1으로 진행합니다.

|  |
| --- |
|  |

* 1. “Explore additional Azure services”를 클릭하여 웹 앱 개발을 위한 리소스를 추가합니다.

|  |
| --- |
|  |

* 1. 초록색 [+] 을 클릭하여 SQL Database를 추가합니다.

|  |
| --- |
|  |

* + 1. SQL DB가 없는 경우 새로운 SQL Server를 먼저 만들어야 합니다. [NEW]를 클릭합니다.

|  |
| --- |
|  |

* + 1. SQL Server의 이름과 인증을 위한 ID 그리고 암호를 입력합니다.

|  |
| --- |
|  |

* + 1. [OK]를 클릭하여 완료 합니다.

|  |
| --- |
|  |

* + 1. [Create]를 클릭하여 마법사를 완료 합니다.

|  |
| --- |
|  |

1. 샘플 ASP.NET Application이 자동으로 생성 됩니다. SQL Server와 DB의 경우 생성이 동시에 진행되는 것을 아래 “Web Publish Activity” 창으로 볼 수 있습니다. 서버와 DB가 생성되는데까지는 3~5 분이 소요될 수 있으나 웹 앱 개발과 동시에 진행이 가능하므로 계속해서 실습을 진행하도록 합니다.

|  |
| --- |
|  |

* 1. 웹 첫 화면에 표시될 웹 앱에 이름을 변경합니다. 솔루션 탐색기에서 Project > Views > Shared > \_Layout.cshtml을 편집합니다.

|  |
| --- |
|  |

* 1. 20번째 줄에 있는 아래 내용을 적절하게 변경합니다.
     1. 아래 표시된 “Application name”을 다른 임의의 이름으로 적절하게 수정합니다.

|  |  |
| --- | --- |
| 전 |  |
| 후 |  |

* 1. 27번 줄에 아래 코드를 추가 합니다..

|  |  |
| --- | --- |
| <li>@Html.ActionLink("Notes", "Index", "Notes")</li> | |
| 전 |  |
| 후 |  |

* 1. 테스트를 위해 가상 컴퓨터에서 웹 앱을 실행해보도록 합니다. 화면 상단에 초록색 빌드 버튼을 클릭합니다. (또는 CTRL + F5로 실행합니다.)

|  |
| --- |
|  |

* 1. 가상 컴퓨터 로컬에서 이상없이 실행되는 것을 확인 합니다. 브라우저 창을 닫으면 자동으로 디버그 모드가 종료됩니다.

|  |
| --- |
|  |

1. Azure에 배포하도록 합니다.
   1. 솔루션 탐색기에서 프로젝트명에서 마우스 오른쪽 클릭을 하면 “Publish”라는 메뉴를 볼 수 있습니다. Publish를 클릭합니다.

|  |
| --- |
|  |

* 1. “Publish Web” 창이 뜨면 [Publish] 버튼을 클락하여 코드를 Azure로 배포 합니다.

|  |
| --- |
|  |

* 1. 빌드 상태에 대한 세부적인 정보가 “Output” 창에 표시 됩니다.

|  |
| --- |
|  |

* 1. 배포가 완료되면 자동으로 웹 브라우저에 배포된 웹 앱이 접속됩니다.

|  |
| --- |
|  |

1. SSL를 활성화하여 웹 앱의 보안 수준을 향상 시킵니다.
   1. 프로젝트 속성 창으로 접근합니다. 프로젝트를 클릭하고 F4 키를 누릅르면 프로젝트의 Properties 창을 볼 수 있습니다. (이미 기본으로 활성화 되어있을 수 있습니다.)

|  |
| --- |
|  |

* 1. SSL Enabled를 False에서 True로 변경합니다

|  |
| --- |
|  |

* 1. SSL URL을 더블클릭하고 CTRL+C로 복사 합니다.

|  |
| --- |
|  |

* 1. 프로젝트에서 마우스 오른쪽 클릭하고 Properties를 클릭합니다..

|  |
| --- |
|  |

* 1. 복사한 주소를 “Project Url”에 붙여 넣습니다. 그리고 변경된 내용을 저장 (CTRL+S) 합니다.

|  |
| --- |
|  |

1. 보안이 강화된 프로젝트를 실행합니다. 아래와 같이 “Internet Explorer”가 선택되었는지 확인하고 초록색 재생 버튼을 클릭하여 웹 앱을 실행해봅니다.

|  |
| --- |
|  |

* 1. Self-signed certificate를 생성할 것인지에 대한 창이 뜨면 [Yes]를 클릭하여 계속 진행합니다. 만약 아래와같은 경고창이 뜨지 않고 HTTPS로 접속이 안되는 경우는 IIS Express 를 삭제 후 재설치합니다.

|  |
| --- |
|  |

* 1. [Yes]를 클릭하여 계속 진행합니다.

|  |
| --- |
|  |

* + 1. Internet Explorer 경우 SSL 경고 없이 홈페이지가 바로 보입니다.

|  |
| --- |
|  |

* + 1. IE가 아닌 다른 웹 브라우저를 사용하는 경우는 아래와 같은 SSL 경고가 보일 수 있습니다. 특히 FireFox 경우는 아래와 같은 경고를 보여주지만 “I Understand the Risks”를 클릭하면 경고를 무시하고 홈 페이지에 접속할 수 있게 됩니다.

|  |
| --- |
| FireFox Cert Warning |

1. 이제부터는 데이터베이스를 활용해서 데이터를 저장하고 관리할 수 있는 기능을 웹 엡에 추가하도록 합니다. 이번 실습에서는 EF (Entity Framework)를 사용하여 간단한 텍스트 기록을 위한 기능을 개발합니다.
   1. 프로젝트의 Models에서 마우스 오른쪽 클릭하여 Add > Class를 클릭하여 데이터 모델을 위한 클래스를 작성합니다.

|  |
| --- |
|  |

* 1. Class가 선택되었는지 확인하고 이름(Name:)에 Note라고 입력 합니다. [Add]를 클릭하여 추가 합니다.

|  |
| --- |
|  |

* 1. 다음과 같이 Note.cs 파일이 생성 됩니다.

|  |
| --- |
|  |

* 1. 생성된 파일을 삭제하고 아래 코드를 복사 (CTRL+C)하여 붙여넣(CTRL+V)습니다. .

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  using System.ComponentModel.DataAnnotations;  using System.Globalization;  namespace myboard.Models  {  public class Note  {  public int NoteId { get; set; }  public string Subject { get; set; }  public string Content { get; set; }  public string Writer { get; set; }  [DataType(DataType.DateTime)]  public DateTime Writenon { get; set; }  public string Complete { get; set; }  [DataType(DataType.EmailAddress)]  public string Email { get; set; }  }  } |
|  |

* 1. Build를 실행하여 추가된 클래스가 프로젝트에 반영되도록 합니다.

|  |
| --- |
|  |

1. 데이터를 정의 했으니 해당 데이터를 CRUD할 수 있도록 추가로 개발합니다.
   1. 새로운 컨트롤 부분을 추가합니다. 프로젝트에서 마우스 오른쪽을 클릭하고 “Add” > “Controller”를 클릭합니다.

|  |
| --- |
|  |

* 1. “add Scaffold” 창이 뜨면 아래와 같이 “MVC 5 Controller … Framework”를 선택하고 [Add]를 클릭합니다.

|  |
| --- |
|  |

* 1. Model class와 Data context class를 다음과같이 선택합니다. 그리고 […]을 클릭하여 layout page를 선택합니다.

|  |
| --- |
|  |

* 1. Layout page는 프로젝트 하위폴더 View > Shared > \_Layout.cshtml을 선택하고 [OK]를 클릭합니다.

|  |
| --- |
|  |

* 1. 다음과 같이 동일하게 설정되었는지 확인하고 [Add]를 클릭하여 완료 합니다..

|  |
| --- |
|  |

1. 처음 웹 앱을 배포할 때에 데이터베이스가 올바르게 구성될 수 있도록 데이터 초기화 및 샘플 데이터 추가를 진행합니다.
   1. Visual studio 메뉴, TOOLS에서 NuGet Package Manager > Package Manager Console을 클릭합니다.

|  |
| --- |
|  |

* 1. 콘솔창이 뜨면 아래 명령을 실행합니다. 아래 명령이 실행되면 프로젝트에 Migration 폴더개 생성 됩니다.

|  |
| --- |
| enable-migrations |
|  |

* 1. 이어서 콘솔창에서 다음 명령을 실행합니다.

|  |
| --- |
| add-migration Initial |
|  |

* 1. 프로젝트 Migrations 폴더에 Configuration.cs를 다음과 같이 편집합니다.

|  |
| --- |
| namespace myboard.Migrations  {  using System;  using System.Data.Entity;  using System.Data.Entity.Migrations;  using System.Linq;  internal sealed class Configuration : DbMigrationsConfiguration<myboard.Models.ApplicationDbContext>  {  public Configuration()  {  AutomaticMigrationsEnabled = false;  }  protected override void Seed(myboard.Models.ApplicationDbContext context)  {  // This method will be called after migrating to the latest version.  // You can use the DbSet<T>.AddOrUpdate() helper extension method  // to avoid creating duplicate seed data. E.g.  //  // context.People.AddOrUpdate(  // p => p.FullName,  // new Person { FullName = "Andrew Peters" },  // new Person { FullName = "Brice Lambson" },  // new Person { FullName = "Rowan Miller" }  // );  //  }  }  } |
| namespace myboard.Migrations  {  using System;  using System.Data.Entity;  using System.Data.Entity.Migrations;  using System.Linq;  using myboard.Models;  internal sealed class Configuration : DbMigrationsConfiguration<myboard.Models.ApplicationDbContext>  {  public Configuration()  {  AutomaticMigrationsEnabled = false;  }  protected override void Seed(myboard.Models.ApplicationDbContext context)  {  context.Notes.AddOrUpdate(  p => p.NoteId,  new Note {  Subject = "Azure Hands on Lab",  Content = "Azure Hands on Lab for Web App and SQL DB development.",  Writer = "HyunSuk Shin",  Writenon = DateTime.UtcNow.ToUniversalTime(),  Complete = "Completed",  Email = "hyunsuk.shin@outlook.com"  }  );  }  }  } |

* 1. 이어서 콘솔창에서 다음 명령을 실행합니다.

|  |
| --- |
| update-database |
|  |

* 1. 명령이 완료되면 웹 앱을 실행합니다.

|  |
| --- |
|  |

* 1. 주소창에 아래와 같이 Notes를 추가히여 데이터가 보이는지 확인합니다.

|  |
| --- |
|  |

1. 웹 앱 개발이 종료되었습니다. 실제 Azure에 배포하여 실제 서비스를 제공하도록 합니다.
   1. 프로젝트에서 마우스 오른쪽을 클릭하여 Publish를 클릭합니다.

|  |
| --- |
|  |

* 1. 아래와 같이 Publish 창이 뜨면 [ < Prev ]를 클릭하여 Settings 단계로 돌아 갑니다.

|  |
| --- |
|  |

* 1. Settings 단계에서 ApplicationDbContext가 아래와같이 올바르게 표시되는지 확인합니다. 보이지 않는 경우 해당 connection string이 보일때까지 잠시 대기 합니다. 그리고 “Execute Code First ….”에 체크박스를 체크하도록 합니다. 그리고 [Publish] 를 클릭하여 Azure 실제 배포하도록 합니다.

|  |
| --- |
|  |

* 1. 다음과 같이 데이터가 올바르게 보이는지 확인합니다.

|  |
| --- |
|  |

* 1. [Create New] 를 클릭하여 샘플 데이터를 입력하여 정상 동작 여부를 확인 합니다.

|  |
| --- |
|  |

* + 1. 다음과같이 양식을 채워서 작성하고 [Create] 를 클릭합니다.

|  |
| --- |
|  |

* 1. 정상적으로 데이터가 입력됨을 확인합니다.

|  |
| --- |
|  |

1. 웹 앱 개발이 종료되었습니다. 실제 서비스 운영을 위해서 웹 앱이 운영되는 서버의 사양을 올리고 Auto-Scale을 적용합니다.
   1. Azure Portal에 접속하고 개발/배포한 웹 앱을 선택합니다.

|  |
| --- |
|  |

* 1. Settings > Scale up (App Service plan)을 클릭합니다.

|  |
| --- |
|  |

* 1. 현재 B1으로 선택 되어있습니다. 해당 앱을 P1으로 상향 조정하기 위해 P1을 선택 한 뒤 [Select]를 클릭하여 완료 합니다.

|  |
| --- |
|  |

* 1. App Service plan 업데이트 진행됩니다.

|  |
| --- |
|  |

* 1. Overview에서 다음과같이 App Service plan/pricing tier가 변경 되었는지 확인 합니다.

|  |
| --- |
|  |

* 1. 다음은, Auto-Scale을 설정합니다. Settings 에서 Scale out (App Service plan)을 클릭합니다.

|  |
| --- |
|  |

* 1. 현재 설정은 별도의 규칙 없이 1개의 인스턴스가 동작하는 것을 확인 할 수 있습니다. \*Scale by에 옵션을 변경하여 규칙을 적용해 봅니다.

|  |
| --- |
|  |

* 1. 규칙을 다음과 같이 “schedule and performance rules”로 선택합니다. [Add Profile] 을 클릭합니다.

|  |
| --- |
|  |

* 1. Name에 “week”를 입력하고 Type은 “recurrence”를 선택합니다. Target Range는 scale out될 인스턴스의 수량을 의미하여 이번 실습에서는 최소 2대에서 최대 10로 설정합니다.

|  |
| --- |
|  |

* 1. 아래 그림과 같이 동일하게 설정되었다면 [OK]를 클릭하여 첫 번째 프로필 추가를 완료 합니다.

|  |
| --- |
|  |

* 1. 첫 번째 프로파일이 추가 된것을 확인합니다. 그리고 프로파일에 규칙을 추가 합니다. 규칙 추가는 [Add Rule]을 클릭하여 진행합니다.

|  |
| --- |
|  |

* 1. 규칙으로는 CPU Percentage를 선택합니다.

|  |
| --- |
|  |

* 1. CPU 조건에 따라 대응되는 Action을 정의 합니다. 아래와 같이 설정하는 경우 CPU 사용률이 지난 10분 동안 최대(MAX) 60%이상을 기록한 경우 인스턴스수를 1개 증가 합니다. 이 때 증가치는 최대 범위로 지정된 10개를 넘을 수 없습니다. 1개가 증가한 경우 5분 동안은 Scale 동작을 하지 않습니다. 빈번한 인스턴스 증감은 성능 저하를 초래 할 수 있기 때문에 해당 옵션이 제공되고 있습니다.

|  |
| --- |
|  |

* 1. 이번에는 인스턴스가 감소하는 조건을 추가 합니다.

|  |
| --- |
|  |

* 1. [Add Profile]을 클릭하여 프로파일을 추가 합니다.

|  |
| --- |
|  |

* 1. Weekend 프로파일을 생성합니다. Type은 recurrence로 설정하고, Days에는 Sunday와 Saturday만 선택합니다. 나머지 옵션은 변경없이 그대로 두고 [OK]를 클릭하여 저장합니다.

|  |
| --- |
|  |

* 1. Scale 의 변화가 생기면 관리자 및 추가 email 주소로 알림 메일을 보내도록 설정합니다.

|  |
| --- |
|  |

* 1. 끝으로 [Save]를 클릭하여 Scale out 프로파일과 규칙을 저장합니다.

|  |
| --- |
|  |

1. 배포된 웹 앱을 사용자들이 잘 사용하는지, 사용자 경험은 어떤지 추적하는 것은 매우 중요합니다. 쉽고 빠르게 웹 앱의 서비스 사용 상태에 대한 정보를 취합하여 분석할 수 있도록 합니다.
   1. 웹 앱에서 Monitoring > Application Insights를 클릭하여 아래와 같이 새로운 리소스에 Application Insights를 추가합니다.

|  |
| --- |
|  |

* 1. 웹 앱 성능 추적을 위해 추가된 코드가 로드 되도록 재시작합니다. 아래와 같이 화면에 표시된 경고 문구를 클릭하도록 합니다.

|  |
| --- |
|  |

* 1. 잠시 기다리면 성공적으로 구성되었다는 것을 확인 할 수 있습니다. 테스트를 위해서 개발/배포한 웹 앱에 접속합니다. 추가적으로 몇 개 메뉴들을 클릭하여 성능 정보가 수집되도록 합니다.

|  |
| --- |
|  |

* 1. 아래와 같이 실시간으로 성능정보가 수집되는 것을 확인 할 수 있습니다.

|  |
| --- |
|  |

1. 부하 테스트를 통해서 개발한 웹 앱의 성능이 어떤지 살펴보도록 하겠습니다. 부하 테스트는 특정 웹 페이지 또는 Visual Studio에서 작성한 복합적인 부하 테스트 시나리오를 기준으로 부하를 줄 수있습니다. 이번 실습에서는 개발한 Notes 기능에 부하를 주는 단순 테스트를 진행 하도록 하겠습니다.

|  |
| --- |
|  |

* 1. [+ New]를 클릭하여 새로운 부하 테스트 시나리오를 입력합니다.

|  |
| --- |
|  |

* 1. ”Configure Test Using”을 클릭하여 Test Type을 Manual Test로 선택하고 URL에 [http://*url*/Notes](http://url/Notes) 를 입력합니다.

|  |
| --- |
|  |

* 1. 부하를 발생하는 지역은 East Asia를 선택 합니다.

|  |
| --- |
|  |

* 1. 테스트 시간에 여유가 있는 경우 Duration (Minutes)를 10분 정도로 하여 테스트를 합니다. [Run Test]를 클릭하여 부하 테스트 시나리오 입력을 완료 합니다.

|  |
| --- |
|  |

* 1. 처음 부하 테스트를 하는 경우 VSTS (Visual Studio Team Service) 계정이 임의로 생성 됩니다.

|  |
| --- |
|  |

* 1. 계정 생성이 완료되면 입력한 부하 테스트 시나리오가 Queue 됩니다. 부하를 위한 리소스 확보 후 실제 부하를 주는 실행에는 약 15분정도 대기가 발생 할 수 있습니다.

|  |
| --- |
|  |

* 1. 부하 테스트를 위한 자원이 확보되고 부하를 주는 5분을 포함하여 약 20분 ~ 25분이 소요될 수 있습니다. 부하가 진행된다면 아래 그림에서 볼 수 있는 것 처럼 Web App 관리 메뉴에서 실시간 메트릭을 확인 할 수 있습니다.

|  |
| --- |
|  |

* 1. 부하 테스트가 완료되면 아래와 같이 테스트 결과를 요약하여 볼 수 있습니다. 부하를 주는 동안 Web App의 사용률과 요청/응답 성공/실패 건수, 초당 요청 처리 수 등을 확인 할 수 있습니다.

|  |
| --- |
|  |

실습이 종료되었습니다. 불필요한 서비스를 삭제하여 비용 지출을 최소화 하도록 합니다.