Weather Stack Net Core

Sample .NET Core Application for consuming Weather Stack API

Table of Contents

1 Symbol Reference

1.1 Variables

The following table lists variables in this documentation.

Variables

Name	Description
Lobibox (see page 1)	This is variable Lobibox.
MyBookShelfWeb (see page 1)	This is variable MyBookShelfWeb.

1.1.1 Lobibox Variable

JavaScript

```
var Lobibox = Lobibox || {};
```

File

notifications.js (see page 16)

Description

This is variable Lobibox.

1.1.2 MyBookShelfWeb Variable

JavaScript

```
var MyBookShelfWeb = MyBookShelfWeb || {};
```

File

notificationManager.js (see page 16)

Description

This is variable MyBookShelfWeb.

1.2 Files

The following table lists files in this documentation.

Files

Name	Description
AutoComplete.cs (see page 2)	This is file AutoComplete.cs.
AutoCompleteController.cs (see page 3)	This is file AutoCompleteController.cs.
AutoCompleteControllerTests.cs (see page 5)	This is file AutoCompleteControllerTests.cs.
AutoCompleteViewModel.cs (see page 6)	This is file AutoCompleteViewModel.cs.
Current.cs (see page 7)	This is file Current.cs.
CurrentWeather.cs (see page 7)	This is file CurrentWeather.cs.
CurrentWeatherController.cs (see page 8)	This is file CurrentWeatherController.cs.
CurrentWeatherControllerTests.cs (see page 10)	This is file CurrentWeatherControllerTests.cs.
CurrentWeatherViewModel.cs (see page 12)	This is file CurrentWeatherViewModel.cs.
Error.cs (see page 13)	This is file Error.cs.
ErrorResult.cs (see page 13)	This is file ErrorResult.cs.
ErrorViewModel.cs (see page 13)	This is file ErrorViewModel.cs.
GeneralRequest.cs (see page 14)	This is file GeneralRequest.cs.
HomeController.cs (see page 14)	This is file HomeController.cs.
ItemList.cs (see page 15)	This is file ItemList.cs.
Location.cs (see page 15)	This is file Location.cs.
LocationRequest.cs (see page 15)	This is file LocationRequest.cs.
notificationManager.js (see page 16)	Look documentation for more detail http://lobianijs.com/site/lobibox#lobibox-notification-usage
notifications.js (see page 16)	
site.js (see page 24)	Please see documentation at https://docs.microsoft.com/aspnet/core/client-side/bundling-and-minification for details on configuring this project to bundle and minify static web assets.
WeatherStackNetCore.csproj (see page 24)	This is file WeatherStackNetCore.csproj.
WeatherStackNetCore.sln (see page 24)	This is file WeatherStackNetCore.sln.
WeatherStackNetCore.Tests.csproj (see page 24)	This is file WeatherStackNetCore.Tests.csproj.

1.2.1 AutoComplete.cs

This is file AutoComplete.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;

/// <summary>
/// API Request Result Class
/// </summary>
public class AutoComplete
{
    public LocationRequest request { get; set; }
    public List<Location> results { get; set; }
    public string success { get; set; }
    public Error error { get; set; }
}
```

1.2.2 AutoCompleteController.cs

This is file AutoCompleteController.cs.

Body Source

```
?using System.Runtime.Serialization.Json;
using System.Text;
using Microsoft.AspNetCore.Mvc;
using WeatherStackNetCore.Models;
using WeatherStackNetCore.Utils;

namespace WeatherStackNetCore.Controllers;

/// <summary>
/// This controller is used to
/// Call API function which is about location searching
/// </summary>
public class AutoCompleteController : Controller
{
    /// <summary>
    /// Config element
    /// </summary>
    private readonly IConfiguration _configuration;

    /// <summary>
    /// This is the constructor of controller
    /// </summary>
    /// <param name="config"></param>
    public AutoCompleteController(IConfiguration config)
    {
        // this is used to get some key values from appSettings.Json
        _configuration = config;
    }

    /// <summary>
    /// This is used to show Location search page
    /// </summary>
    /// <returns>Location search page</returns>
    public IActionResult IndexWithJQuery()
    {
        var model = new AutoCompleteViewModel();
        return View(model);
    }

    /// <summary>
    /// This is used to show Location search page
    /// </summary>
    /// <returns>Location search page</returns>
    public IActionResult IndexWithModel(AutoCompleteViewModel? model)
    {
        model ??= new AutoCompleteViewModel();
        return View(model);
    }

    /// <summary>
    /// This method is used to make WeatherStack Current API call
    /// And bring the result to the screen
    /// </summary>
    /// <param name="placeName">General Parameter (Like City Name, County Name etc)</param>
    /// <returns>Location Info</returns>
    [HttpPost]
    public async Task<AutoComplete?> GetLocations(string placeName)
    {
        try
        {
            var autoComplete = await CallAutoCompleteFromAPI(placeName);
            return autoComplete;
        }
        catch (Exception e)
        {
            return null;
        }
    }

    /// <summary>
    /// This method is used to make WeatherStack Current API call
    /// And bring the result to the screen
    /// But in this method we use the whole model
    /// </summary>
    /// <param name="model">Location Search Page Elements</param>
    /// <returns>Result Page</returns>
    [HttpPost]
    public async Task<IActionResult> GetLocationsWithModel(AutoCompleteViewModel model)
    {
        try
        {
            if (!ModelState.IsValid) return View("IndexWithModel", model);

            var autoComplete = await CallAutoCompleteFromAPI(model.PlaceName);

            model = new AutoCompleteViewModel
            {
                AutoComplete = autoComplete
            };

            if (model.AutoComplete.error != null)
                SetErrorMessage(model.AutoComplete.error.info);
            else
                SetSuccessMessage();

            return View("IndexWithModel", model);
        }
        catch (Exception e)
        {
            model = new AutoCompleteViewModel
            {
                AutoComplete = null
            };

            SetErrorMessage("Something has gone wrong");

            return View("IndexWithModel", model);
        }
    }

    /// <summary>
    /// This method is used to set success message to ViewBag
    /// </summary>
    private void SetSuccessMessage()
    {
        ViewBag.MessageType = "success";
        ViewBag.BoxType = "normal";
        ViewBag.Message = "Operation is successful";
    }

    /// <summary>
    /// This method is used to set error message to ViewBag
    /// </summary>
    /// <param name="errorMessage">Specific Error Message</param>
    private void SetErrorMessage(string errorMessage)
    {
        ViewBag.MessageType = "error";
        ViewBag.BoxType = "large";
        ViewBag.Message = errorMessage;
    }

    /// <summary>
    /// This method is used to call WeatherStack API with place name
    /// And get the city info
    /// </summary>
    /// <param name="placeName">Place Name</param>
    /// <returns>Auto Complete Data (City etc. Data)</returns>
    private async Task<AutoComplete> CallAutoCompleteFromAPI(string? placeName)
    {
        var apiKey = _configuration.GetValue<string>("API_Key");

        var httpClient = new HttpClient();
        httpClient.Timeout = new TimeSpan(0, 0, 30);

        var requestString = "http://api.weatherstack.com/" +
$"autocomplete?access_key={apiKey}&query={placeName}";

        var response = await httpClient.GetAsync(requestString);
        var result = await response.Content.ReadAsStringAsync();

        var serializer = new DataContractJsonSerializer(typeof(AutoComplete));
        var memoryStream = new MemoryStream(Encoding.UTF8.GetBytes(result));
        var autoComplete = (AutoComplete) serializer.ReadObject(memoryStream)!;
        return autoComplete;
    }
}
```

1.2.3 AutoCompleteControllerTests.cs

This is file AutoCompleteControllerTests.cs.

Body Source

```
using Microsoft.Extensions.Configuration;
using Moq;
using WeatherStackNetCore.Controllers;
using WeatherStackNetCore.Models;

namespace WeatherStackNetCore.Tests.Controllers;

/// <summary>
/// Test class for the controller
/// Which is used to get location info from the
/// WeatherStack API
/// </summary>
public class AutoCompleteControllerTests
{
    /// <summary>
    /// Location Info Model
    /// </summary>
    private AutoCompleteViewModel? _model;
    /// <summary>
    /// Config file for controller constructor
    /// </summary>
    private IConfiguration? _config;
    /// <summary>
    /// Mock class instance
    /// </summary>
    private Mock? _mock;
    /// <summary>
    /// Controller instance
    /// </summary>
    private AutoCompleteController? _controller;
```

```
        /// <summary>
        /// This method is used to create setup
        /// Assign values which are necessary
        /// Before using test methods
        /// </summary>
        [SetUp]
        public void Setup()
        {
            _model = new AutoCompleteViewModel
            {
                PlaceName = "Ankara"
            };
            _config = new ConfigurationManager();
            _controller = new AutoCompleteController(_config);
        }

        /// <summary>
        /// Test method which is used for
        /// Index action with model
        /// </summary>
        [Test]
        public void IndexWithModel_Test()
        {
            var result = _controller?.IndexWithModel(_model);
            Assert.That(result, Is.Not.Null);
        }

        /// <summary>
        /// Test method which is used for
        /// Get Location action with Model
        /// </summary>
        [Test]
        public void GetLocationsWithModel_Test()
        {
            var result = _controller?.GetLocationsWithModel(_model);
            Assert.That(result, Is.Not.Null);
        }

        /// <summary>
        /// Test method which is used for
        /// Index action with JQuery
        /// </summary>
        [Test]
        public void IndexWithJQuery_Test()
        {
            var result = _controller?.IndexWithJQuery();
            Assert.That(result, Is.Not.Null);
        }

        /// <summary>
        /// Test method which is used for
        /// Get Location action with JQuery
        /// </summary>
        [Test]
        public void GetLocationsWithJQuery_Test()
        {
            var result = _controller?.GetLocations(_model.PlaceName);
            Assert.AreSame(result.Exception, null);
        }
    }
```

1.2.4 AutoCompleteViewModel.cs

This is file AutoCompleteViewModel.cs.

Body Source

```
using System.ComponentModel.DataAnnotations;
using WeatherStackNetCore.Utils;

namespace WeatherStackNetCore.Models;

/// <summary>
/// This model class is used to create the search page
/// And stores the result info too
/// </summary>
public class AutoCompleteViewModel
{
    #region Fields

    [Required(ErrorMessage = "Please Enter A Place Name")]
    [Display(Name = "Place Name:")]
    public string? PlaceName { get; set; }

    public AutoComplete? AutoComplete { get; init; }

    #endregion
}
```

1.2.5 Current.cs

This is file Current.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;

/// <summary>
/// Query Place Current Weather Information Class
/// </summary>
public class Current
{
    public string observation_time { get; set; }
    public decimal temperature { get; set; }
    public decimal weather_code { get; set; }
    public List<string> weather_icons { get; set; }
    public List<string> weather_descriptions { get; set; }
    public decimal wind_speed { get; set; }
    public decimal wind_degree { get; set; }
    public string wind_dir { get; set; }
    public decimal pressure { get; set; }
    public decimal precip { get; set; }
    public decimal humidity { get; set; }
    public decimal cloudcover { get; set; }
    public decimal feelslike { get; set; }
    public decimal uv_index { get; set; }
    public decimal visibility { get; set; }
}
```

1.2.6 CurrentWeather.cs

This is file CurrentWeather.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;
```

```
/// <summary>
/// API Request Result Class
/// </summary>
public class CurrentWeather
{
    public GeneralRequest request { get; set; }
    public Location location { get; set; }
    public Current current { get; set; }
    public string success { get; set; }
    public Error error { get; set; }
}
```

1.2.7 CurrentWeatherController.cs

This is file CurrentWeatherController.cs.

Body Source

```
?using System.Runtime.Serialization.Json;
using System.Text;
using Microsoft.AspNetCore.Mvc;
using WeatherStackNetCore.Models;
using WeatherStackNetCore.Utils;

namespace WeatherStackNetCore.Controllers;

/// <summary>
/// This controller is used to
/// Call API function which is about Current Weather Info searching
/// </summary>
public class CurrentWeatherController : Controller
{
    /// <summary>
    /// Config element
    /// </summary>
    private readonly IConfiguration _configuration;

    /// <summary>
    /// This is the constructor of controller
    /// </summary>
    /// <param name="config"></param>
    public CurrentWeatherController(IConfiguration config)
    {
        // this is used to get some key values from appSettings.Json
        _configuration = config;
    }

    /// <summary>
    /// This is used to show Current Weather search page
    /// </summary>
    /// <returns>Current Weather search page</returns>
    public IActionResult IndexWithJQuery()
    {
        var model = new CurrentWeatherViewModel();
        return View(model);
    }

    /// <summary>
    /// This is used to show Current Weather search page
    /// </summary>
    /// <returns>Current Weather search page</returns>
    public IActionResult IndexWithModel(CurrentWeatherViewModel? model)
    {
        model ??= new CurrentWeatherViewModel();
        return View(model);
    }
    /// <summary>
    /// This method is used to make WeatherStack Current API call
    /// And bring the result to the screen
    /// </summary>
    /// <param name="placeName">General Parameter (Like City Name, County Name etc)</param>
    /// <param name="unit">Unit Parameter</param>
    /// <param name="language">Language Parameter (You shouldn't fill this parameter if
your key is free version)</param>
    /// <returns>Current Weather Info</returns>
    [HttpPost]
    public async Task<CurrentWeather?> GetCurrentWeather(string placeName, string unit,
string language)
    {
        try
        {
            var currentWeather = await GetCurrentWeatherFromAPI(placeName, unit, language);
            return currentWeather;
        }
        catch (Exception e)
        {
            return null;
        }
    }

    /// <summary>
    /// This method is used to make WeatherStack Current API call
    /// And bring the result to the screen
    /// But in this method we use the whole model
    /// </summary>
    /// <param name="model">Current Weather Search Page Elements</param>
    /// <returns>Result Page</returns>
    [HttpPost]
    public async Task<IActionResult> GetCurrentWeatherWithModel(CurrentWeatherViewModel
model)
    {
        try
        {
            if (!ModelState.IsValid) return View("IndexWithModel", model);

            var currentWeather = await GetCurrentWeatherFromAPI(model.PlaceName,
model.Unit, model.Language);

            model = new CurrentWeatherViewModel
            {
                CurrentWeather = currentWeather
            };

            if (model.CurrentWeather.error != null)
                SetErrorMessage(model.CurrentWeather.error.info);
            else
                SetSuccessMessage();

            return View("IndexWithModel", model);
        }
        catch (Exception e)
        {
            model = new CurrentWeatherViewModel
            {
                CurrentWeather = null
            };

            SetErrorMessage("Something has gone wrong");

            return View("IndexWithModel", model);
        }
    }

    /// <summary>
    /// This method is used to set success message to ViewBag
    /// </summary>
    private void SetSuccessMessage()
    {
        ViewBag.MessageType = "success";
        ViewBag.BoxType = "normal";
        ViewBag.Message = "Operation is successful";
    }

    /// <summary>
    /// This method is used to set error message to ViewBag
    /// </summary>
    /// <param name="errorMessage">Specific Error Message</param>
    private void SetErrorMessage(string errorMessage)
    {
        ViewBag.MessageType = "error";
        ViewBag.BoxType = "large";
        ViewBag.Message = errorMessage;
    }

    /// <summary>
    /// This method is used to get Current Weather Info from Weather Stack API
    /// And return the result
    /// </summary>
    /// <param name="placeName">Place Name Info</param>
    /// <param name="unit">Unit Info</param>
    /// <param name="language">Language Info</param>
    /// <returns></returns>
    private async Task<CurrentWeather> GetCurrentWeatherFromAPI(string placeName, string
unit, string language)
    {
        var apiKey = _configuration.GetValue<string>("API_Key");

        var httpClient = new HttpClient();
        httpClient.Timeout = new TimeSpan(0, 0, 30);

        var requestString = "http://api.weatherstack.com/" +
$"current?access_key={apiKey}&query={placeName}";
        if (!string.IsNullOrEmpty(unit))
            requestString += $"&unit={unit}";
        if (!string.IsNullOrEmpty(language))
            requestString += $"&language={language}";

        var response = await httpClient.GetAsync(requestString);
        var result = await response.Content.ReadAsStringAsync();

        var serializer = new DataContractJsonSerializer(typeof(CurrentWeather));
        var memoryStream = new MemoryStream(Encoding.UTF8.GetBytes(result));
        var currentWeather = (CurrentWeather) serializer.ReadObject(memoryStream)!;

        return currentWeather;
    }
}
```

1.2.8 CurrentWeatherControllerTests.cs

This is file CurrentWeatherControllerTests.cs.

Body Source

```
using Microsoft.Extensions.Configuration;
using Moq;
using WeatherStackNetCore.Controllers;
using WeatherStackNetCore.Models;

namespace WeatherStackNetCore.Tests.Controllers;
```

```
/// <summary>
/// Test class for the controller
/// Which is used to get current weather info from the
/// WeatherStack API
/// </summary>
public class CurrentWeatherControllerTests
{
    /// <summary>
    /// Current Weather Info Model
    /// </summary>
    private CurrentWeatherViewModel? _model;
    /// <summary>
    /// Config file for controller constructor
    /// </summary>
    private IConfiguration? _config;
    /// <summary>
    /// Mock class instance
    /// </summary>
    private Mock? _mock;
    /// <summary>
    /// Controller instance
    /// </summary>
    private CurrentWeatherController? _controller;

    /// <summary>
    /// This method is used to create setup
    /// Assign values which are necessary
    /// Before using test methods
    /// </summary>
    [SetUp]
    public void Setup()
    {
        _model = new CurrentWeatherViewModel
        {
            PlaceName = "Konya"
        };
        _config = new ConfigurationManager();
        _controller = new CurrentWeatherController(_config);
    }

    /// <summary>
    /// Test method which is used for
    /// Index action with model
    /// </summary>
    [Test]
    public void IndexWithModel_Test()
    {
        var result = _controller?.IndexWithModel(_model);
        Assert.That(result, Is.Not.Null);
    }

    /// <summary>
    /// Test method which is used for
    /// Get Current Weather Info action with Model
    /// </summary>
    [Test]
    public void GetCurrentWeatherWithModel_Test()
    {
        var result = _controller?.GetCurrentWeatherWithModel(_model);
        Assert.That(result, Is.Not.Null);
    }

    /// <summary>
    /// Test method which is used for
    /// Index action with JQuery
    /// </summary>
    [Test]
    public void IndexWithJQuery_Test()
    {
        var result = _controller?.IndexWithJQuery();
        Assert.That(result, Is.Not.Null);
    }

    /// <summary>
    /// Test method which is used for
    /// Get Current Weather Info action with JQuery
    /// </summary>
    [Test]
    public void GetCurrentWeatherWithJQuery_Test()
    {
        var result = _controller?.GetCurrentWeather(_model.PlaceName, _model.Unit,
_model.Language);
        Assert.AreSame(result.Exception, null);
    }
}
```

1.2.9 CurrentWeatherViewModel.cs

This is file CurrentWeatherViewModel.cs.

Body Source

```
using System.ComponentModel.DataAnnotations;
using WeatherStackNetCore.Utils;

namespace WeatherStackNetCore.Models;

/// <summary>
/// This model class is used to create the search page
/// And stores the result info too
/// </summary>
public class CurrentWeatherViewModel
{
    #region Fields

    [StringLength(60, MinimumLength = 3)]
    [Required(ErrorMessage = "Please Enter A Place Name")]
    [Display(Name = "Place Name:")]
    public string? PlaceName { get; set; }

    [Display(Name = "Unit:")] public string? Unit { get; init; }

    [Display(Name = "Language:")] public string? Language { get; init; }

    /// <summary>
    /// This is used to fill dropdownlist for units
    /// </summary>
    public List<ItemList> UnitList { get; init; }

    /// <summary>
    /// This is used to fill dropdownlist for languages
    /// </summary>
    public List<ItemList> LanguageList { get; init; }

    /// <summary>
    /// API Call result
    /// </summary>
    public CurrentWeather? CurrentWeather { get; init; }

    #endregion

    public CurrentWeatherViewModel()
    {
        UnitList = new List<ItemList>
```

```
          {
              new() {Text = "Metric", Value = "m"},
              new() {Text = "Scientific", Value = "s"},
              new() {Text = "Fahrenheit", Value = "f"}
          };

          LanguageList = new List<ItemList>
          {
              new() {Text = "English", Value = "en"},
              new() {Text = "Turkish", Value = "tr"},
              new() {Text = "German", Value = "de"}
          };
      }
  }
```

1.2.10 Error.cs

This is file Error.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;

/// <summary>
/// Error Class
/// </summary>
public class Error
{
    public string type { get; set; }
    public string info { get; set; }
    public int code { get; set; }
}
```

1.2.11 ErrorResult.cs

This is file ErrorResult.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;

/// <summary>
/// Error Class
/// </summary>
public class ErrorResult
{
    public string success { get; set; }
    public Error error { get; set; }
}
```

1.2.12 ErrorViewModel.cs

This is file ErrorViewModel.cs.

Body Source

```
namespace WeatherStackNetCore.Models;

public class ErrorViewModel
```

```
{
    public string? RequestId { get; init; }

    public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
}
```

1.2.13 GeneralRequest.cs

This is file GeneralRequest.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;

/// <summary>
/// API Request Class
/// </summary>
public class GeneralRequest
{
    public string type { get; set; }
    public string query { get; set; }
    public string language { get; set; }
    public string unit { get; set; }
}
```

1.2.14 HomeController.cs

This is file HomeController.cs.

Body Source

```
?using System.Diagnostics;
using Microsoft.AspNetCore.Mvc;
using WeatherStackNetCore.Models;

namespace WeatherStackNetCore.Controllers;

public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;

    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }

    public IActionResult Index()
    {
        return View();
    }

    public IActionResult Privacy()
    {
        return View();
    }

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
    public IActionResult Error()
    {
        return View(new ErrorViewModel {RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier});
    }
}
```

1.2.15 ItemList.cs

This is file ItemList.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;

public class ItemList
{
    public string Text { get; set; }
    public string Value { get; set; }
}
```

1.2.16 Location.cs

This is file Location.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;

/// <summary>
/// Query Place Detailed Location Info Class
/// </summary>
public class Location
{
    public string name { get; set; }
    public string country { get; set; }
    public string region { get; set; }
    public string lat { get; set; }
    public string lon { get; set; }
    public string timezone_id { get; set; }
    public string localtime { get; set; }
    public decimal localtime_epoch { get; set; }
    public string utc_offset { get; set; }
}
```

1.2.17 LocationRequest.cs

This is file LocationRequest.cs.

Body Source

```
?namespace WeatherStackNetCore.Utils;

/// <summary>
/// API Request Class
/// </summary>
public class LocationRequest
{
    public string query { get; set; }
    public int results { get; set; }
}
```

1.2.18 notificationManager.js

Look documentation for more detail http://lobianijs.com/site/lobibox#lobibox-notification-usage

Body Source

```
?//Look documentation for more detail
http://lobianijs.com/site/lobibox#lobibox-notification-usage
var MyBookShelfWeb = MyBookShelfWeb || {};


MyBookShelfWeb.notification = {

    ShowErrorMessage: function (title,message) {
        Lobibox.notify('error', {
            title:title,
            msg: message
        });
    },

    ShowOKMessage: function (title, message) {
        Lobibox.notify('success', {
            title: title,
            msg: message
        });
    },

    ShowInfoMessage: function (title, message) {
        Lobibox.notify('info', {
            title: title,
            msg: message
        });
    },

    ShowWarningMessage: function (title, message) {
        Lobibox.notify('warning', {
            title: title,
            msg: message
        });
    }
}
```

Variables

Name	Description
MyBookShelfWeb (see page 1)	This is variable MyBookShelfWeb.

1.2.19 notifications.js

@arboshiki *

- Generates random string of n length.
- String contains only letters and numbers

*

- @param {int} n
- @returns {String}

Body Source

```
//Author      : @arboshiki
/**
 * Generates random string of n length.
 * String contains only letters and numbers
 *
 * @param {int} n
 * @returns {String}
 */
Math.randomString = function (n) {
    var text = "";
    var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

    for (var i = 0; i < n; i++)
        text += possible.charAt(Math.floor(Math.random() * possible.length));

    return text;
};
var Lobibox = Lobibox || {};
(function () {

    var LobiboxNotify = function (type, options) {
//------------------------------------------------------------------------------
//----------------PROTOTYPE VARIABLES-------------------------------------------
//------------------------------------------------------------------------------
        this.$type = null;
        this.$options = null;
        this.$el = null;
//------------------------------------------------------------------------------
//-----------------PRIVATE VARIABLES--------------------------------------------
//------------------------------------------------------------------------------
        var me = this;
//------------------------------------------------------------------------------
//-----------------PRIVATE FUNCTIONS--------------------------------------------
//------------------------------------------------------------------------------
        var _processInput = function (options) {

            if (options.size === 'mini' || options.size === 'large' || options.size ===
'awesome') {
                options = $.extend({}, Lobibox.notify.OPTIONS[options.size], options);
            }
            options = $.extend({}, Lobibox.notify.OPTIONS[me.$type],
Lobibox.notify.DEFAULTS, options);

            if (options.size !== 'mini' && options.title === true) {
                options.title = Lobibox.notify.OPTIONS[me.$type].title;
            } else if (options.size === 'mini' && options.title === true) {
                options.title = false;
            }
            if (options.icon === true) {
                options.icon = Lobibox.notify.OPTIONS.icons[options.iconSource][me.$type];
            }
            if (options.sound === true) {
                options.sound = Lobibox.notify.OPTIONS[me.$type].sound;
            }
            if (options.sound) {
                options.sound = options.soundPath + options.sound + options.soundExt;
            }
            return options;
        };

        var _appendInWrapper = function ($el, $wrapper) {
            if (me.$options.size === 'normal') {
                if ($wrapper.hasClass('bottom')) {
                    $wrapper.prepend($el);
                } else {
                    $wrapper.append($el);
```

```
                  }

              } else if (me.$options.size === 'mini') {
                  if ($wrapper.hasClass('bottom')) {
                      $wrapper.prepend($el);
                  } else {
                      $wrapper.append($el);
                  }
              } else if (me.$options.size === 'large') {
                  var tabPane = _createTabPane().append($el);
                  var $li = _createTabControl(tabPane.attr('id'));
                  $wrapper.find('.lb-notify-wrapper').append(tabPane);
                  $wrapper.find('.lb-notify-tabs').append($li);
                  _activateTab($li);
                  $li.find('>a').click(function () {
                      _activateTab($li);
                  });
              }
              else if (me.$options.size === 'awesome') {
                  var tabPane = _createTabPane().append($el);
                  var $li = _createTabControl(tabPane.attr('id'));
                  $wrapper.find('.lb-notify-wrapper').append(tabPane);
                  $wrapper.find('.lb-notify-tabs').append($li);
                  _activateTab($li);
                  $li.find('>a').click(function () {
                      _activateTab($li);
                  });
              }
          };
          var _activateTab = function ($li) {
              $li.closest('.lb-notify-tabs').find('>li').removeClass('active');
              $li.addClass('active');
              var $current = $($li.find('>a').attr('href'));
              $current.closest('.lb-notify-wrapper').find('>.lb-tab-pane').removeClass('active
');
              $current.addClass('active');
          };
          var _createTabControl = function (tabPaneId) {
              var $li = $('<li></li>', {
                  'class': Lobibox.notify.OPTIONS[me.$type]['class']
              });
              $('<a></a>', {
                  'href': '#' + tabPaneId
              }).append('<i class="tab-control-icon ' + me.$options.icon + '"></i>')
                  .appendTo($li);
              return $li;
          };
          var _createTabPane = function () {
              return $('<div></div>',
                  {
                      'class': 'lb-tab-pane',
                      'id': Math.randomString(10)
                  });
          };
          var _createNotifyWrapper = function () {
              var selector = (me.$options.size === 'large' ? '.lobibox-notify-wrapper-large'
: '.lobibox-notify-wrapper')
                      + "." + me.$options.position.replace(/\s/gi, '.'),
                  $wrapper;

              //var classes = me.$options.position.split(" ");
              $wrapper = $(selector);
              if ($wrapper.length === 0) {
                  $wrapper = $('<div></div>')
                      .addClass(selector.replace(/\./g, ' ').trim())
                      .appendTo($('body'));
                  if (me.$options.size === 'large') {
                      $wrapper.append($('<ul class="lb-notify-tabs"></ul>'))
                          .append($('<div class="lb-notify-wrapper"></div>'));
```

```
                }
                if (me.$options.size === 'awesome') {
                    $wrapper.append($('<ul class="lb-notify-tabs"></ul>'))
                        .append($('<div class="lb-notify-wrapper"></div>'));
                }
            }
            return $wrapper;
        };
        var _createNotify = function () {
            var OPTS = Lobibox.notify.OPTIONS,
                $iconEl,
                $innerIconEl,
                $iconWrapper,
                $body,
                $msg,
                $notify = $('<div></div>', {
                    'class': 'lobibox-notify ' + OPTS[me.$type]['class'] + ' ' +
OPTS['class'] + ' ' + me.$options.showClass
                });

            $iconWrapper = $('<div
class="lobibox-notify-icon-wrapper"></div>').appendTo($notify);
            $iconEl = $('<div class="lobibox-notify-icon"></div>').appendTo($iconWrapper);
            $innerIconEl = $('<div></div>').appendTo($iconEl);

            // Add image or icon depending on given parameters
            if (me.$options.img) {
                $innerIconEl.append('<img src="' + me.$options.img + '"/>');
            } else if (me.$options.icon) {
                $innerIconEl.append('<div class="icon-el"><i class="' + me.$options.icon +
'"></i></div>');
            } else {
                $notify.addClass('without-icon');
            }
            // Create body, append title and message in body and append body in notification
            $msg = $('<div class="lobibox-notify-msg">' + me.$options.msg + '</div>');

            if (me.$options.messageHeight !== false) {
                $msg.css('max-height', me.$options.messageHeight);
            }

            $body = $('<div></div>', {
                'class': 'lobibox-notify-body'
            }).append($msg).appendTo($notify);

            if (me.$options.title) {
                $body.prepend('<div class="lobibox-notify-title">' + me.$options.title +
'<div>');
            }
            _addCloseButton($notify);
            // burada tï¿½klandï¿½ï¿½ï¿½nda kapanmasï¿½ iï¿½in gerekli dï¿½zenlemeler
yapï¿½lï¿½yor
            if (me.$options.size === 'normal' || me.$options.size === 'mini' ||
me.$options.size === 'large' || me.$options.size === 'awesome') {
                _addCloseOnClick($notify);
                _addDelay($notify);
            }

            // Give width to notification
            if (me.$options.width) {
                $notify.css('width', _calculateWidth(me.$options.width));
            }

            return $notify;
        };
        var _addCloseButton = function ($el) {
            if (!me.$options.closable) {
                return;
            }
```

```
                $('<span class="lobibox-close">&times;</span>').click(function () {
                    me.remove();
                }).appendTo($el);
            };
            var _addCloseOnClick = function ($el) {
                if (!me.$options.closeOnClick) {
                    return;
                }
                $el.click(function () {
                    me.remove();
                });
            };
            var _addDelay = function ($el) {
                if (!me.$options.delay) {
                    return;
                }
                if (me.$options.delayIndicator) {
                    var delay = $('<div class="lobibox-delay-indicator"><div></div></div>');
                    $el.append(delay);
                }
                var time = 0;
                var interval = 1000 / 30;
                var currentTime = new Date().getTime();
                var timer = setInterval(function () {
                    if (me.$options.continueDelayOnInactiveTab){
                        time = new Date().getTime() - currentTime;
                    } else {
                        time += interval;
                    }

                    var width = 100 * time / me.$options.delay;
                    if (width >= 100) {
                        width = 100;
                        me.remove();
                        timer = clearInterval(timer);
                    }
                    if (me.$options.delayIndicator) {
                        delay.find('div').css('width', width + "%");
                    }

                }, interval);

                if (me.$options.pauseDelayOnHover) {
                    $el.on('mouseenter.lobibox', function () {
                        interval = 0;
                    }).on('mouseleave.lobibox', function () {
                        interval = 1000 / 30;
                    });
                }
            };
            var _findTabToActivate = function ($li) {
                var $itemToActivate = $li.prev();
                if ($itemToActivate.length === 0) {
                    $itemToActivate = $li.next();
                }
                if ($itemToActivate.length === 0) {
                    return null;
                }
                return $itemToActivate;
            };
            var _calculateWidth = function (width) {
                width = Math.min($(window).outerWidth(), width);
                return width;
            };
    //------------------------------------------------------------------------------
    //----------------PROTOTYPE FUNCTIONS-------------------------------------------
    //------------------------------------------------------------------------------
            /**
             * Delete the notification
```

```
         *
         * @returns {LobiboxNotify}
         */
        this.remove = function () {
            me.$el.removeClass(me.$options.showClass)
                .addClass(me.$options.hideClass);
            var parent = me.$el.parent();
            var wrapper = parent.closest('.lobibox-notify-wrapper-large');

            var href = '#' + parent.attr('id');

            var $li = wrapper.find('>.lb-notify-tabs>li:has(a[href="' + href + '"])');
            $li.addClass(Lobibox.notify.OPTIONS['class'])
                .addClass(me.$options.hideClass);
            setTimeout(function () {
                if (me.$options.size === 'normal' || me.$options.size === 'mini') {
                    me.$el.remove();
                } else if (me.$options.size === 'large') {

                    var $newLi = _findTabToActivate($li);
                    if ($newLi) {
                        _activateTab($newLi);
                    }
                    $li.remove();
                    parent.remove();
                }
                else if (me.$options.size === 'awesome') {

                    var $newLi = _findTabToActivate($li);
                    if ($newLi) {
                        _activateTab($newLi);
                    }
                    $li.remove();
                    parent.remove();
                }
                var list = Lobibox.notify.list;
                var ind = list.indexOf(me);
                list.splice(ind, 1);
                var next = list[ind];
                if (next && next.$options.showAfterPrevious){
                    next._init();
                }
            }, 500);
            return me;
        };
        me._init = function () {
            // Create notification
            var $notify = _createNotify();
            if (me.$options.size === 'mini') {
                $notify.addClass('notify-mini');
            }

            if (typeof me.$options.position === 'string') {
                var $wrapper = _createNotifyWrapper();
                _appendInWrapper($notify, $wrapper);
                if ($wrapper.hasClass('center')) {
                    $wrapper.css('margin-left', '-' + ($wrapper.width() / 2) + "px");
                }
            } else {
                $('body').append($notify);
                $notify.css({
                    'position': 'fixed',
                    left: me.$options.position.left,
                    top: me.$options.position.top
                });
            }

            me.$el = $notify;
            if (me.$options.sound) {
                var snd = new Audio(me.$options.sound); // buffers automatically when
created
                snd.play();
            }
            if (me.$options.rounded) {
                me.$el.addClass('rounded');
            }
            me.$el.on('click.lobibox', function(ev){
                if (me.$options.onClickUrl){
                    window.location.href = me.$options.onClickUrl;
                }
                if (me.$options.onClick && typeof me.$options.onClick === 'function'){
                    me.$options.onClick.call(me, ev);
                }
            });
            me.$el.data('lobibox', me);
        };
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
        this.$type = type;
        this.$options = _processInput(options);
        if (!me.$options.showAfterPrevious || Lobibox.notify.list.length === 0){
            this._init();
        }

    };

    Lobibox.notify = function (type, options) {
        if (["default", "info", "warning", "error", "success"].indexOf(type) > -1) {
            var lobibox = new LobiboxNotify(type, options);
            Lobibox.notify.list.push(lobibox);
            return lobibox;
        }
    };
    Lobibox.notify.list = [];
    Lobibox.notify.closeAll = function () {
        var list = Lobibox.notify.list;
        for (var i in list){
            list[i].remove();
        }
    };
    //User can set default options to this variable
    Lobibox.notify.DEFAULTS = {
        title: true,                // Title of notification. If you do not include the
title in options it will automatically takes its value
        //from Lobibox.notify.OPTIONS object depending of the type of the notifications or
set custom string. Set this false to disable title
        size: 'normal',                                        // normal, mini, large
        soundPath: '/Content/tools/otomasyon-notification/sounds/',       // The folder
path where sounds are located
        soundExt: '.ogg',           // Default extension for all sounds
        showClass: 'fadeInDown',    // Show animation class.
        hideClass: 'zoomOut',       // Hide animation class.
        icon: true,                 // Icon of notification. Leave as is for default icon
or set custom string
        msg: '',                    // Message of notification
        img: null,                  // Image source string
        closable: true,             // Make notifications closable
        hideCloseButton: false,     // Notification may be closable but you can hide close
button and it will be closed by clicking on notification itsef
        delay: 6000,                // Hide notification after this time (in miliseconds)
        delayIndicator: true,       // Show timer indicator
        closeOnClick: true,         // Close notifications by clicking on them
        width: 500,                 // Width of notification box
        sound: true,                // Sound of notification. Set this false to disable
sound. Leave as is for default sound or set custom soud path
        // Place to show notification. Available options: "top left", "top right", "bottom
left", "bottom right", "center top", "center bottom"
```

```
        // It can also be object {left: number, top: number} to position notification at
any place
        position: "bottom right",
        iconSource: 'bootstrap',    // "bootstrap" or "fontAwesome" the library which will
be used for icons
        rounded: false,             // Whether to make notification corners rounded
        messageHeight: 60,          // Notification message maximum height. This is not for
notification itself, this is for <code>.lobibox-notify-msg</code>
        pauseDelayOnHover: true,    // When you mouse over on notification delay (if it is
enabled) will be paused.
        onClickUrl: null,           // The url which will be opened when notification is
clicked
        showAfterPrevious: false,   // Set this to true if you want notification not to be
shown until previous notification is closed. This is useful for notification queues
        continueDelayOnInactiveTab: true, // Continue delay when browser tab is inactive

        // Events
        onClick: null
    };
    //This variable is necessary.
    // large boyutunda sadece hata mesajlarï¿½ gï¿½sterildiï¿½inden
    // title ï¿½zelliï¿½i gï¿½ncellendi
    Lobibox.notify.OPTIONS = {
        'class': 'animated-fast',
        large: {
            width: 500,
            messageHeight: 96,
            'title': 'Error'
        },
        awesome: {
            width: 500,
            messageHeight: 96,
            'title': 'Situation'
        },
        mini: {
            'class': 'notify-mini',
            messageHeight: 32
        },
        default: {
            'class': 'lobibox-notify-default',
            'title': 'Default',
            sound: false
        },
        success: {
            'class': 'lobibox-notify-success',
            'title': 'Success',
            sound: 'sound2'
        },
        error: {
            'class': 'lobibox-notify-error',
            'title': 'Error',
            sound: 'sound4'
        },
        warning: {
            'class': 'lobibox-notify-warning',
            'title': 'Warning',
            sound: 'sound5'
        },
        info: {
            'class': 'lobibox-notify-info',
            'title': 'Info',
            sound: 'sound6'
        },
        icons: {
            bootstrap: {
                success: 'glyphicon glyphicon-ok-sign',
                error: 'glyphicon glyphicon-remove-sign',
                warning: 'glyphicon glyphicon-exclamation-sign',
                info: 'glyphicon glyphicon-info-sign'
            },
            fontAwesome: {
                success: 'fa fa-check-circle',
                error: 'fa fa-times-circle',
                warning: 'fa fa-exclamation-circle',
                info: 'fa fa-info-circle'
            }
        }
    };
})();
```

Variables

Name	Description
Lobibox (see page 1)	This is variable Lobibox.

1.2.20 site.js

Please see documentation at https://docs.microsoft.com/aspnet/core/client-side/bundling-and-minification for details on configuring this project to bundle and minify static web assets.

Body Source

```
?// Please see documentation at
https://docs.microsoft.com/aspnet/core/client-side/bundling-and-minification
// for details on configuring this project to bundle and minify static web assets.

// Write your JavaScript code.
```

1.2.21 WeatherStackNetCore.csproj

This is file WeatherStackNetCore.csproj.

1.2.22 WeatherStackNetCore.sln

This is file WeatherStackNetCore.sln.

1.2.23 WeatherStackNetCore.Tests.csproj

This is file WeatherStackNetCore.Tests.csproj.

Index