

Solution 'LotteryGenerator' (3 projects)

- └─ LotteryGenerator
 - └─ Properties
 - └─ References
 - └─ App_Data
 - └─ App_Start
 - └─ Areas
 - └─ Content
 - └─ Controllers
 - └─ LotteryController.cs
 - └─ fonts
 - └─ Models
 - └─ LotteryGenerator.cs
 - └─ LotterySet.cs
 - └─ RandomHelper.cs
 - └─ Providers
 - └─ Results
 - └─ Scripts
 - └─ Views
 - └─ ApplicationInsights.config
 - └─ favicon.ico
 - └─ Global.asax
 - └─ packages.config
 - └─ Project_Readme.html
 - └─ Startup.cs
 - └─ Web.config
- └─ LotteryGenerator.Tests
 - └─ Properties
 - └─ References
 - └─ Controllers
 - └─ LotteryControllerTest.cs
 - └─ App.config
 - └─ packages.config
- └─ LotteryGeneratorClient
 - └─ Properties
 - └─ References
 - └─ scripts
 - └─ ApplicationInsights.config
 - └─ LotteryView.html
 - └─ packages.config
 - └─ Web.config

```
using LotteryGenerator.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
```

```

namespace LotteryGenerator.Controllers
{
    public class LotteryController : ApiController
    {
        LotteryGenerator.Models.LotteryGenerator _lotteryGeneratorObj;

        public LotteryGenerator.Models.LotteryGenerator LotteryGeneratorObj
        {
            get
            {
                if (_lotteryGeneratorObj == null)
                    _lotteryGeneratorObj = new
LotteryGenerator.Models.LotteryGenerator();
                return _lotteryGeneratorObj;
            }
            set { _lotteryGeneratorObj = value; }
        }

        public LotterySet Get()
        {
            return LotteryGeneratorObj.GetLoterry();
        }

        public LotterySet[] List(int numberOfSets)
        {
            var _setResult = new List<LotterySet>();

            for (int i = 1; i <= numberOfSets; i++)
            {
                _setResult.Add(LotteryGeneratorObj.GetLoterry());
            }

            return _setResult.ToArray();
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace LotteryGenerator.Models
{
    public class LotteryGenerator
    {
        private readonly int _lotterySetMaxLimit;

        private readonly int _lotteryMinRange;

        private readonly int _lotteryMaxRange;

        public LotteryGenerator(int lotterySetMaxLimit = 6, int lotteryMinRange = 1,
int lotteryMaxRange = 49)
        {
            _lotterySetMaxLimit = lotterySetMaxLimit;
            _lotteryMinRange = lotteryMinRange;
            _lotteryMaxRange = lotteryMaxRange;
        }
        public LotterySet GetLoterry()

```

```

    {
        Random _randomHelper = RandomHelper.Instance;

        var _lotterySet = new LotterySet(_lotterySetMaxLimit);

        for (int i = 1; i < _lotterySetMaxLimit; i++)
        {
            int foo;
            do
            {
                foo = _randomHelper.Next(_lotteryMinRange, _lotteryMaxRange);
            } while (_lotterySet.Contains(foo));

            _lotterySet.Add(foo);
        }

        return _lotterySet;
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace LotteryGenerator.Models
{
    public class LotterySet : List<int>
    {
        public LotterySet(int lotterySetMaxLimitparams)
        {
            base.Capacity = lotterySetMaxLimitparams;
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Web;

namespace LotteryGenerator.Models
{
    public static class RandomHelper
    {
        private static int _seedCounter = new Random().Next();

        [ThreadStatic]
        private static Random _rng;

        public static Random Instance
        {
            get
            {
                if (_rng == null)
                {
                    int seed = Interlocked.Increment(ref _seedCounter);
                    _rng = new Random(seed);
                }
                return _rng;
            }
        }
    }
}

```

```

    }
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Web.Http;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using LotteryGenerator;
using LotteryGenerator.Controllers;
using LotteryGenerator.Models;

namespace LotteryGenerator.Tests.Controllers
{
    [TestClass]
    public class LotteryControllerTest
    {
        [TestMethod]
        public void List()
        {
            // Arrange
            var controller = new LotteryController();

            // Act
            LotterySet[] result = controller.List(5);

            // Assert
            Assert.IsNotNull(result);

            Assert.AreEqual(5, result.Count());
        }

        [TestMethod]
        public void Get()
        {
            // Arrange
            var controller = new LotteryController();

            // Act
            LotterySet result = controller.Get();

            // Assert
            Assert.IsNotNull(result);

            Assert.AreEqual(6, result.Count);
            Assert.IsTrue(result[0] > 0);
            Assert.IsTrue(result[1] > 0);
            Assert.IsTrue(result[2] > 0);
            Assert.IsTrue(result[3] > 0);
            Assert.IsTrue(result[4] > 0);
            Assert.IsTrue(result[5] > 0);

            Assert.IsTrue(result.FindAll(i => i == result[0]).Count == 1);
        }
    }
}

```

```

<!DOCTYPE html>
<html>
<head>
  <title>Hello AngularJS</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.3/angular.min.js"></script>
  <script src="scripts/AngularSetup.js"></script>
</head>
<body ng-app="myApp" ng-controller='MyCtrl'>
  <style>

  </style>

  <div ng-repeat="item in items">
    <table>
      <tr ng-style="{background: {{item.Id}}>1 ? 'red':'green'}" ng-
bind="item.Id"> {{item.name}}, {{item.title}},{{item.id}}</tr>
      <!--<tr></tr>
      <tr></tr>
      <tr></tr>
      <tr></tr>
      <tr></tr>-->
    </table>
  </div>
  <br />
  <div ng-style="{background: val>3 ? 'red':'green'}">
    <!--<div ng-app="myApp" ng-controller='MyCtrl' ng-style="{background:
myColor}"-->
    <!--<input type="text" ng-model="myColor" placeholder="enter a color name"-->

    <input type="text" ng-model="val" placeholder="enter a color name">

    <div ng-repeat="item in items" ng-class="{ 'pending-delete': item.checked}">
      name: {{item.name}}, {{item.title}},{{item.id}}
      <input type="checkbox" ng-model="item.checked" />
      <span ng-show="item.checked" /><span>(will be deleted)</span>
    </div>
    <p>
      <div ng-hide="myColor== 'red'">I will hide if the color is set to
'red'.</div>
    </div>
  </body>
</html>

```

```

angular.module('myApp', [])
.controller('MyCtrl', function ($scope) {
  $scope.items = [{
    id: 1,
    name: 'Misko',
    title: 'Angular creator'
  }, {
    id: 2,
    name: 'Igor',
    title: 'Meetup master'
  }, {
    id: 3,
    name: 'Vojta',
    title: 'All-around superhero'
  }
  ];
});

```