```
1  using Moq;
2  using Prism.Regions;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8  using System.Collections;
9
10 using XTest.Services.Imple;
11 using XTest.Prism;
12
13
14 namespace XTest.ViewModels
15 {
16     public abstract class AbstractTestViewModel
17     {
18         // protected readonly ILogger logger;
19         protected readonly RegionManagerMock regionManagerMock;
20         protected readonly EventAggregatorMock eventAggregatorMock;
21         protected readonly MainServiceMock mainServiceMock;
22         protected readonly MeasurementServiceMock
             measurementServiceMock;
23         protected readonly LearningServiceMock learningServiceMock;
24         protected readonly PredictionServiceMock predictionServiceMock;
25
26
27         public AbstractTestViewModel()
28         {
29             // var factory = new
               NLog.Extensions.Logging.NLogLoggerFactory();
30             // this.logger = factory.CreateLogger<AbstractTestViewModel>
               ();
31             regionManagerMock = new RegionManagerMock();
32             eventAggregatorMock = new EventAggregatorMock();
33             mainServiceMock = new MainServiceMock();
34             measurementServiceMock = new MeasurementServiceMock();
35             learningServiceMock = new LearningServiceMock();
36             predictionServiceMock = new PredictionServiceMock();
37         }
38     }
39 }
40
```

```
1  using Prism.Ioc;
2  using System.Windows;
3  using Microsoft.Extensions.Configuration;
4
5  using SampleMvvmPrism.Views;
6  using SampleMvvmPrism.Models.Config;
7  using SampleMvvmPrism.Services.Interface;
8  using SampleMvvmPrism.Services.Imple;
9  using SampleMvvmPrism.Services.Utils;
10
11
12 namespace SampleMvvmPrism
13 {
14     /// <summary>
15     /// Interaction logic for App.xaml
16     /// </summary>
17     public partial class App
18     {
19         protected override Window CreateShell()
20         {
21             return Container.Resolve<MainWindow>();
22         }
23
24         protected override void RegisterTypes(IContainerRegistry
           containerRegistry)
25         {
26             FileUtils fileUtils = new FileUtils();
27             AppSettingsModel appSettings = fileUtils.ReadAppSettings();
28             IMainService mainService = new MainService(appSettings);
29             IMeasurementService measurementService = new
               MeasurementService(appSettings);
30             ILearningService learningServiceService = new
               LearningService(appSettings);
31             IPredictionService predictionService = new PredictionService
               (appSettings);
32             containerRegistry.RegisterInstance<IMainService>
               (mainService);
33             containerRegistry.RegisterInstance<IMeasurementService>
               (measurementService);
34             containerRegistry.RegisterInstance<ILearningService>
               (learningServiceService);
35             containerRegistry.RegisterInstance<IPredictionService>
               (predictionService);
36             containerRegistry.RegisterForNavigation<MeasurementView>();
37             containerRegistry.RegisterForNavigation<LearningView>();
38             containerRegistry.RegisterForNavigation<PredictionView>();
39         }
40     }
41 }
42
```

```
<prism:PrismApplication x:Class="SampleMvvmPrism.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
               presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:SampleMvvmPrism"
             xmlns:materialDesign="http://materialdesigninxaml.net/winfx/
               xaml/themes"
             xmlns:prism="http://prismlibrary.com/" >
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <materialDesign:BundledTheme BaseTheme="Light"
                  PrimaryColor="DeepPurple" SecondaryColor="Lime" />
                <ResourceDictionary Source="pack://application:,,,/
                  MaterialDesignThemes.Wpf;component/Themes/
                  MaterialDesign3.Defaults.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</prism:PrismApplication>
```

```
1
2
3  using System.Runtime.Serialization;
4  using System;
5
6  namespace SampleMvvmPrism.Models.Config
7  {
8      public class AppSettingsModel
9      {
10         public enum SystemTypes
11         {
12             LOCAL,
13             CLOUD
14         }
15
16         public class SystemNetworks
17         {
18             public string Ip { get; set; }
19             public int Port { get; set; }
20
21             public override string ToString()
22             {
23                 return $"SystemNetworks(){{Ip={Ip},Port={Port}}}";
24             }
25         }
26
27         public string SystemDir { get; set; }
28         public SystemTypes SystemType { get; set; }
29         public SystemNetworks SystemNetwork { get; set; }
30
31         public override string ToString()
32         {
33             return $"AppConfigModel(){{SystemDir={SystemDir},SystemType=
                 {SystemType},SystemNetworks={SystemNetwork}}}";
34         }
35     }
36 }
37
```

```
1  using Moq;
2  using Prism.Events;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9
10 namespace XTest.Prism
11 {
12     public class EventAggregatorMock
13     {
14         private Mock<IEventAggregator> mock;
15         public Dictionary<int, object> parameters;
16         private int index = 0;
17
18         public IEventAggregator Object { get { return
             this.mock.Object; } }
19
20         public EventAggregatorMock()
21         {
22             this.mock = new Mock<IEventAggregator>();
23             this.parameters = new Dictionary<int, object>();
24         }
25
26         public void SetEvent<T1>()
27         {
28             var psEvent = this.NewPubSubEvent<T1>(this.index);
29             this.mock.Setup(v => v.GetEvent<PubSubEvent<T1>>()).Returns
               (psEvent);
30             this.index++;
31         }
32
33         private PubSubEvent<T1> NewPubSubEvent<T1>(int index)
34         {
35             this.parameters.Add(index, null);
36             var psEvent = new Mock<PubSubEvent<T1>>();
37             psEvent.Setup(v => v.Publish(It.IsAny<T1>())).Callback<T1>(t
                  => this.parameters[index] = t);
38             return psEvent.Object;
39         }
40
41         public object GetParam(int funcIndex)
42         {
43             if (this.parameters.ContainsKey(funcIndex))
44             {
45                 return this.parameters[funcIndex];
46             }
47             else
48             {
49                 throw new Exception($"TargetFuncHasNotCalled
                    [{funcIndex}]:メソッド呼び出し情報がありません。");
```

```
50              }
51          }
52      }
53  }
54
```

```
 1  using CsvHelper;
 2  using Microsoft.Extensions.Configuration;
 3  using Microsoft.Win32;
 4  using SampleMvvmPrism.Models.Config;
 5  using System;
 6  using System.Collections.Generic;
 7  using System.Globalization;
 8  using System.IO;
 9  using System.Linq;
10  using System.Text;
11  using System.Text.Json;
12  using System.Threading.Tasks;
13
14  namespace SampleMvvmPrism.Services.Utils
15  {
16      public class FileUtils
17      {
18          public AppSettingsModel ReadAppSettings()
19          {
20              IConfigurationRoot configurationRoot = new
                  ConfigurationBuilder()
21                  .AddJsonFile("appsettings.json")
22                  .Build();
23              AppSettingsModel appSettings =
                  configurationRoot.Get<AppSettingsModel>();
24              return appSettings;
25          }
26
27          public RES ReadJson<RES>(string path)
28          {
29              string jsonContent = File.ReadAllText(path);
30              RES jsonModel = JsonSerializer.Deserialize<RES>
                  (jsonContent);
31              return jsonModel;
32          }
33
34          public string OpenFile()
35          {
36              // Microsoft.Win32はリソース解放(Dispose)不要、throwしない
37              OpenFileDialog openFileDialog = new OpenFileDialog()
                  { Filter = "すべてのファイル|*.*" };
38              if (openFileDialog.ShowDialog() == true)
39              {
40                  return openFileDialog.FileName;
41              }
42              else
43              {
44                  throw new Exception("ファイルオープンに失敗しました。");
45              }
46          }
47
48          public string SaveFile()
49          {
```

```
50              // Microsoft.Win32はリソース解放(Dispose)不要、throwしない
51              SaveFileDialog saveFileDialog = new SaveFileDialog()
                  { Filter = "すべてのファイル|*.*" };
52              if (saveFileDialog.ShowDialog() == true)
53              {
54                  return saveFileDialog.FileName;
55              }
56              else
57              {
58                  throw new Exception("ファイル保存に失敗しました。");
59              }
60          }
61
62          public List<RES> ReadCsv<RES>(string path)
63          {
64              using (var reader = new StreamReader(path))
65              {
66                  using (var csv = new CsvReader(reader,
                      CultureInfo.InvariantCulture))
67                  {
68                      return csv.GetRecords<RES>().ToList();
69                  }
70              }
71          }
72
73          public void WriteCsv<REQ>(string path, List<REQ> records)
74          {
75              using (var writer = new StreamWriter(path))
76              {
77                  using (var csv = new CsvWriter(writer,
                      CultureInfo.InvariantCulture))
78                  {
79                      csv.WriteRecords(records);
80                  }
81              }
82          }
83      }
84  }
85
```

```
1  using SampleMvvmPrism.Models.Config;
2  using SampleMvvmPrism.Models.Csv;
3  using SampleMvvmPrism.Models.Http;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace SampleMvvmPrism.Services.Interface
11 {
12     public interface ILearningService
13     {
14         public LearningConfigModel ReadConfigLearning();
15         public Task<LearningResponseModel> PostHttpLearning
             (LearningRequestModel requestModel);
16         public Task<LearningResponseModel> PostHttpLearningConfig
             (LearningConfigModel configModel);
17     }
18 }
19
```

```
1  using SampleMvvmPrism.Models.Config;
2  using SampleMvvmPrism.Models.Csv;
3  using SampleMvvmPrism.Models.Http;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace SampleMvvmPrism.Services.Interface
11 {
12     public interface IMainService
13     {
14         public AppSettingsModel GetAppSettings();
15         /*
16         public MeasurementConfigModel ReadConfigMeasurement();
17         public LearningConfigModel ReadConfigLearning();
18         public PredictionConfigModel ReadConfigPrediction();
19         public List<MeasurementCsvModel> ReadCsvMeasurement();
20         public List<PredictionCsvModel> ReadCsvPrediction();
21         public void WriteCsvPrediction(List<PredictionResultsCsvModel>
             csvModels);
22         public List<MeasurementRequestModel> FormatDatasetMeasurement
             (IEnumerable<MeasurementCsvModel> csvModel);
23         public List<PredictionRequestModel> FormatDatasetPrediction
             (IEnumerable<PredictionCsvModel> csvModel);
24         public List<PredictionResultsCsvModel>
             FormatDatasetPredictionResults
             (IEnumerable<PredictionResponseModel> respounceModel);
25         public Task<List<MeasurementResponseModel>> PostHttpMeasurement
             (List<MeasurementRequestModel> requestModels);
26         public Task<MeasurementResponseModel> PostHttpMeasurementConfig
             (MeasurementConfigModel configModel);
27         public Task<LearningResponseModel> PostHttpLearning
             (LearningRequestModel requestModel);
28         public Task<LearningResponseModel> PostHttpLearningConfig
             (LearningConfigModel configModel);
29         public Task<List<PredictionResponseModel>> PostHttpPrediction
             (List<PredictionRequestModel> requestModel);
30         public Task<PredictionResponseModel> PostHttpPredictionConfig
             (PredictionConfigModel configModel);
31         */
32     }
33 }
34
```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SampleMvvmPrism.Models.Config
{
    public class LearningConfigModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public double Value { get; set; }
    }
}

```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SampleMvvmPrism.Models.Http
{
    public class LearningRequestModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public double Value { get; set; }
    }
}

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SampleMvvmPrism.Models.Http
8  {
9      public class LearningResponseModel
10     {
11         public int Id { get; set; }
12         public string Name { get; set; }
13         public double Value { get; set; }
14     }
15 }
16
```

```
using CsvHelper;
using Microsoft.Win32;
using SampleMvvmPrism.Models.Config;
using SampleMvvmPrism.Models.Csv;
using SampleMvvmPrism.Models.Http;
using SampleMvvmPrism.Services.Interface;
using SampleMvvmPrism.Services.Utils;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Text.Json;
using System.Threading;
using System.Threading.Tasks;


namespace SampleMvvmPrism.Services.Imple
{
    public class LearningService : ILearningService
    {
        private readonly AppSettingsModel appSettings;
        private readonly RestUtils restUtils;
        private readonly FileUtils fileUtils;
        private readonly SemaphoreSlim semaphore = new SemaphoreSlim(1,
          1);

        public LearningService(AppSettingsModel appSettings)
        {
            this.appSettings = appSettings;
            restUtils = new RestUtils();
            fileUtils = new FileUtils();
        }

        public LearningConfigModel ReadConfigLearning()
        {
            string filePath = fileUtils.OpenFile();
            LearningConfigModel configModel =
              fileUtils.ReadJson<LearningConfigModel>(filePath);
            return configModel;
        }

        public async Task<LearningResponseModel> PostHttpLearning
          (LearningRequestModel requestModel)
        {
            await semaphore.WaitAsync();
            try
            {
                string url = $"http://{appSettings.SystemNetwork.Ip}:
                  {appSettings.SystemNetwork.Port}/learning";
                LearningResponseModel responseModel = await
                  restUtils.ModelsPost<LearningRequestModel,
                  LearningResponseModel>(url, requestModel);
```

```
48                  return responseModel;
49              }
50              finally
51              {
52                  semaphore.Release();
53              }
54          }
55
56          public async Task<LearningResponseModel> PostHttpLearningConfig
              (LearningConfigModel configModel)
57          {
58              await semaphore.WaitAsync();
59              try
60              {
61                  string url = $"http://{appSettings.SystemNetwork.Ip}:
                      {appSettings.SystemNetwork.Port}/learning/config";
62                  LearningResponseModel responseModel = await
                      restUtils.ModelsPost<LearningConfigModel,
                      LearningResponseModel>(url, configModel);
63                  return responseModel;
64              }
65              finally
66              {
67                  semaphore.Release();
68              }
69          }
70      }
71  }
72
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using Moq;
7
8  using SampleMvvmPrism.Services.Interface;
9  using SampleMvvmPrism.Models.Http;
10 using SampleMvvmPrism.Models.Config;
11
12
13 namespace XTest.Services.Imple
14 {
15     public class LearningServiceMock
16     {
17         public Mock<ILearningService> mock;
18         public Dictionary<string, object> parameters;
19         public Dictionary<string, object> returns;
20
21         public ILearningService Object { get { return
             this.mock.Object; } }
22
23         public LearningServiceMock()
24         {
25             this.parameters = new Dictionary<string, object>();
26             this.returns = new Dictionary<string, object>();
27
28             this.mock = new Mock<ILearningService>();
29             this.mock.Setup(it => it.ReadConfigLearning())
30                 .Callback(() => this.parameters["ReadConfigLearning"] =
                   new Object())
31                 .Returns(() => (LearningConfigModel)this.GetReturn
                   ("ReadConfigLearning"));
32             this.mock.Setup(it => it.PostHttpLearning
                 (It.IsAny<LearningRequestModel>()))
33                 .Callback<LearningRequestModel>((model) =>
                   this.parameters["PostHttpLearning"] = (model))
34                 .Returns(() => (Task<LearningResponseModel>)
                   this.GetReturn("PostHttpLearning"));
35             this.mock.Setup(it => it.PostHttpLearningConfig
                 (It.IsAny<LearningConfigModel>()))
36                 .Callback<LearningConfigModel>((model) =>
                   this.parameters["PostHttpLearningConfig"] = (model))
37                 .Returns(() => (Task<LearningResponseModel>)
                   this.GetReturn("PostHttpLearningConfig"));
38         }
39
40         public object GetParam(string targetFunc)
41         {
42             if (this.parameters.ContainsKey(targetFunc))
43             {
44                 return this.parameters[targetFunc];
```

```
45              }
46              else
47              {
48                  throw new Exception($"TargetFuncHasNotCalled
                      [{targetFunc}]:メソッド呼び出し情報がありません。");
49              }
50          }
51
52          public object SetReturn(string targetFunc, object value)
53          {
54              return this.returns[targetFunc] = value;
55          }
56
57          private object GetReturn(string targetFunc)
58          {
59              if (this.returns.ContainsKey(targetFunc))
60              {
61                  return this.returns[targetFunc];
62              }
63              else
64              {
65                  throw new Exception($"TargetFuncHasNotSetReturn
                      [{targetFunc}]:メソッドのリターン値が設定されていません。");
66              }
67          }
68      }
69  }
70
```

```cs
using System.Windows.Controls;

namespace SampleMvvmPrism.Views
{
    /// <summary>
    /// Interaction logic for LearningView.xaml
    /// </summary>
    public partial class LearningView : UserControl
    {
        public LearningView()
        {
            InitializeComponent();
        }
    }
}

```

```
<UserControl x:Class="SampleMvvmPrism.Views.LearningView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
               presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
               compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:SampleMvvmPrism.Views"
             xmlns:materialDesign="http://materialdesigninxaml.net/winfx/
               xaml/themes"
             mc:Ignorable="d"
             d:DesignHeight="600" d:DesignWidth="800" >
    <Grid>
        <StackPanel
        Margin="0,0,0,0"
        HorizontalAlignment="Center"
        VerticalAlignment="Center">
            <Label
            Margin="0,0,0,10"
            HorizontalAlignment="Center"
            FontSize="32">
                学習機能
            </Label>

            <TextBlock
            HorizontalAlignment="Center"
            FontSize="20"
            Text=""
            Visibility="Hidden" />
            <StackPanel
                Margin="0,0,0,10"
                HorizontalAlignment="Center"
                Orientation="Horizontal">
                <Label Width="150" FontSize="24">IP</Label>
                <TextBox
                    Width="350"
                    HorizontalAlignment="Right"
                    FontSize="35"
                    IsReadOnly="false"
                    IsEnabled="true"
                    Text="{Binding Ip, Mode=TwoWay,
                   UpdateSourceTrigger=PropertyChanged}"/>
            </StackPanel>
            <StackPanel
                Margin="0,0,0,10"
                HorizontalAlignment="Center"
                Orientation="Horizontal">
                <Label Width="150" FontSize="24">ポート</Label>
                <TextBox
                    Width="350"
                    HorizontalAlignment="Right"
                    FontSize="35"
                    IsReadOnly="false"
```

```
                    IsEnabled="true"
                    Text="{Binding Port, Mode=TwoWay,
                   UpdateSourceTrigger=PropertyChanged}"/>
            </StackPanel>
            <StackPanel
                Margin="0,0,0,10"
                HorizontalAlignment="Center"
                Orientation="Horizontal">
                <Label Width="150" FontSize="24">コンフィグ</Label>
                <Button
                    Width="350"
                    FontSize="17"
                    HorizontalAlignment="Center"
                    Content="Open(.json)"
                    Command="{Binding OnClickButton}"
                    CommandParameter="CONFIG"
                    Style="{StaticResource MaterialDesignFlatButton}" />
            </StackPanel>
            <StackPanel
                Margin="0,0,0,10"
                HorizontalAlignment="Center"
                Orientation="Horizontal">
                <Label Width="150" FontSize="24">APIコール</Label>
                <Button
                    Width="350"
                    FontSize="17"
                    HorizontalAlignment="Center"
                    Content="Post(HTTP)"
                    Command="{Binding OnClickButton}"
                    CommandParameter="HTTP"
                    Style="{StaticResource MaterialDesignFlatButton}" />
            </StackPanel>
        </StackPanel>
    </Grid>
</UserControl>
```

```
 1  using Prism.Commands;
 2  using Prism.Events;
 3  using Prism.Mvvm;
 4  using Prism.Regions;
 5  using System;
 6  using System.Windows;
 7  using SampleMvvmPrism.Views;
 8  using SampleMvvmPrism.Models.Config;
 9  using SampleMvvmPrism.Models.Http;
10  using SampleMvvmPrism.Services.Interface;
11  using SampleMvvmPrism.Services.Imple;
12
13
14  namespace SampleMvvmPrism.ViewModels
15  {
16      public class LearningViewModel : BindableBase, INavigationAware
17      {
18          private readonly IRegionManager regionManager;
19          private readonly IEventAggregator eventAggregator;
20          private readonly IMainService mainService;
21          private readonly ILearningService learningService;
22          private AppSettingsModel appSettings;
23
24          private string ip = string.Empty;
25          public string Ip
26          {
27              get { return this.ip; }
28              set
29              {
30                  this.appSettings.SystemNetwork.Ip = value;
31                  SetProperty(ref this.ip, value);
32              }
33          }
34          private int port = 0;
35          public int Port
36          {
37              get { return this.port; }
38              set
39              {
40                  this.appSettings.SystemNetwork.Port = value;
41                  SetProperty(ref this.port, value);
42              }
43          }
44          public DelegateCommand<string> OnClickButton { get; set; }
45          public LearningConfigModel ConfigModel { get; set; }
46
47          public LearningViewModel(IRegionManager regionManager,
              IEventAggregator eventAggregator, IMainService mainService,
              ILearningService learningService)
48          {
49              this.regionManager = regionManager;
50              this.eventAggregator = eventAggregator;
51              this.mainService = mainService;
```

```
52              this.learningService = learningService;
53              this.OnClickButton = new DelegateCommand<string>
                  (this.OnEventButton);
54          }
55
56          private async void OnEventButton(string message)
57          {
58              if (message == "CONFIG")
59              {
60                  try
61                  {
62                      this.ConfigModel =
                    this.learningService.ReadConfigLearning();
63                      LearningResponseModel responseModel = await
                    this.learningService.PostHttpLearningConfig
                    (this.ConfigModel);
64                      MessageBox.Show(responseModel.ToString());
65                  }
66                  catch (Exception ex)
67                  {
68                      MessageBox.Show(ex.Message);
69                  };
70              }
71              else if (message == "HTTP")
72              {
73                  LearningRequestModel requestModel = new
                    LearningRequestModel();
74                  try
75                  {
76                      LearningResponseModel responseModel = await
                    this.learningService.PostHttpLearning(requestModel);
77                      MessageBox.Show(responseModel.ToString());
78                  }
79                  catch (Exception ex)
80                  {
81                      MessageBox.Show(ex.Message);
82                  };
83              }
84              else
85              {
86                  throw new NotImplementedException();
87              }
88          }
89
90          /// <summary>Viewを表示した後呼び出されます。</summary>
91          /// <param name="navigationContext">Navigation Requestの情報を表
              すNavigationContext。</param>
92          public void OnNavigatedTo(NavigationContext navigationContext)
93          {
94              this.appSettings = this.mainService.GetAppSettings();
95              this.Ip = this.appSettings.SystemNetwork.Ip;
96              this.Port = this.appSettings.SystemNetwork.Port;
97          }
```

```
 98
 99          /// <summary>表示するViewを判別します。</summary>
100          /// <param name="navigationContext">Navigation Requestの情報を表
               すNavigationContext。</param>
101          /// <returns>表示するViewかどうかを表すbool。</returns>
102          public bool IsNavigationTarget(NavigationContext
               navigationContext)
103          {
104              return true;
105          }
106
107          /// <summary>別のViewに切り替わる前に呼び出されます。</summary>
108          /// <param name="navigationContext">Navigation Requestの情報を表
               すNavigationContext。</param>
109          public void OnNavigatedFrom(NavigationContext
               navigationContext)
110          {
111              return;
112          }
113      }
114  }
115
116
```

```
 1  using CsvHelper;
 2  using Microsoft.Win32;
 3  using SampleMvvmPrism.Models.Config;
 4  using SampleMvvmPrism.Models.Csv;
 5  using SampleMvvmPrism.Models.Http;
 6  using SampleMvvmPrism.Services.Interface;
 7  using SampleMvvmPrism.Services.Utils;
 8  using System;
 9  using System.Collections.Generic;
10  using System.Globalization;
11  using System.IO;
12  using System.Linq;
13  using System.Text.Json;
14  using System.Threading;
15  using System.Threading.Tasks;
16
17
18  namespace SampleMvvmPrism.Services.Imple
19  {
20      public class MainService : IMainService
21      {
22          private readonly AppSettingsModel appSettings;
23          private readonly RestUtils restUtils;
24          private readonly FileUtils fileUtils;
25          private readonly SemaphoreSlim semaphore = new SemaphoreSlim(1,
                1);
26
27          public MainService(AppSettingsModel appSettings)
28          {
29              this.appSettings = appSettings;
30              restUtils = new RestUtils();
31              fileUtils = new FileUtils();
32          }
33
34          public AppSettingsModel GetAppSettings()
35          {
36              return appSettings;
37          }
38
39          /*
40          public MeasurementConfigModel ReadConfigMeasurement()
41          {
42              string filePath = fileUtils.OpenFile();
43              MeasurementConfigModel configModel =
                  fileUtils.ReadJson<MeasurementConfigModel>(filePath);
44              return configModel;
45          }
46
47          public LearningConfigModel ReadConfigLearning()
48          {
49              string filePath = fileUtils.OpenFile();
50              LearningConfigModel configModel =
                  fileUtils.ReadJson<LearningConfigModel>(filePath);
```

```
51              return configModel;
52          }
53
54          public PredictionConfigModel ReadConfigPrediction()
55          {
56              string filePath = fileUtils.OpenFile();
57              PredictionConfigModel configModel =                          ↵
                  fileUtils.ReadJson<PredictionConfigModel>(filePath);
58              return configModel;
59          }
60
61          public List<MeasurementCsvModel> ReadCsvMeasurement()
62          {
63              string filePath = fileUtils.OpenFile();
64              IEnumerable<MeasurementCsvModel> datasetModels =             ↵
                  fileUtils.ReadCsv<MeasurementCsvModel>(filePath);
65              return datasetModels.ToList();
66          }
67
68          public List<PredictionCsvModel> ReadCsvPrediction()
69          {
70              string filePath = fileUtils.OpenFile();
71              IEnumerable<PredictionCsvModel> datasetModels =              ↵
                  fileUtils.ReadCsv<PredictionCsvModel>(filePath);
72              return datasetModels.ToList();
73          }
74
75          public void WriteCsvPrediction(List<PredictionResultsCsvModel>  ↵
              csvModels)
76          {
77              string filePath = fileUtils.SaveFile();
78              fileUtils.WriteCsv(filePath, csvModels);
79          }
80
81          public List<MeasurementRequestModel> FormatDatasetMeasurement   ↵
              (IEnumerable<MeasurementCsvModel> csvModels)
82          {
83              List<MeasurementRequestModel> requestModels = new           ↵
                  List<MeasurementRequestModel>();
84              foreach (MeasurementCsvModel csvModel in csvModels)
85              {
86                  requestModels.Add(new MeasurementRequestModel()
87                  {
88                      Id = csvModel.Id,
89                      Name = csvModel.Name,
90                      Value = csvModel.Value
91                  });
92              }
93              return requestModels;
94          }
95
96          public List<PredictionRequestModel> FormatDatasetPrediction     ↵
              (IEnumerable<PredictionCsvModel> csvModels)
```

```
 97          {
 98              List<PredictionRequestModel> requestModels = new
                   List<PredictionRequestModel>();
 99              foreach (PredictionCsvModel csvModel in csvModels)
100              {
101                  requestModels.Add(new PredictionRequestModel()
102                  {
103                      Id = csvModel.Id,
104                      Name = csvModel.Name,
105                      Value = csvModel.Value
106                  });
107              }
108              return requestModels;
109          }
110
111          public List<PredictionResultsCsvModel>
               FormatDatasetPredictionResults
               (IEnumerable<PredictionResponseModel> respounceModels)
112          {
113              List<PredictionResultsCsvModel> requestModels = new
                   List<PredictionResultsCsvModel>();
114              foreach (PredictionResponseModel respounceModel in
                   respounceModels)
115              {
116                  requestModels.Add(new PredictionResultsCsvModel()
117                  {
118                      Id = respounceModel.Id,
119                      Name = respounceModel.Name,
120                      Value = respounceModel.Value
121                  });
122              }
123              return requestModels;
124          }
125
126          public async Task<List<MeasurementResponseModel>>
               PostHttpMeasurement(List<MeasurementRequestModel>
               requestModels)
127          {
128              await semaphore.WaitAsync();
129              try
130              {
131                  List<MeasurementResponseModel> responseModels = new
                       List<MeasurementResponseModel>();
132                  foreach (MeasurementRequestModel requestModel in
                       requestModels)
133                  {
134                      string url = $"http://
                       {appSettings.SystemNetwork.Ip}:
                       {appSettings.SystemNetwork.Port}/measurement";
135                      MeasurementResponseModel responseModel = await
                       restUtils.ModelsPost<MeasurementRequestModel,
                       MeasurementResponseModel>(url, requestModel);
136                      responseModels.Add(responseModel);
```

```
137                 }
138                 return responseModels;
139             }
140             finally
141             {
142                 semaphore.Release();
143             }
144         }
145 
146         public async Task<MeasurementResponseModel>                    ↵
              PostHttpMeasurementConfig(MeasurementConfigModel configModel)
147         {
148             await semaphore.WaitAsync();
149             try
150             {
151                 string url = $"http://{appSettings.SystemNetwork.Ip}:  ↵
                    {appSettings.SystemNetwork.Port}/measurement/config";
152                 MeasurementResponseModel responseModel = await         ↵
                    restUtils.ModelsPost<MeasurementConfigModel,           ↵
                    MeasurementResponseModel>(url, configModel);
153                 return responseModel;
154             }
155             finally
156             {
157                 semaphore.Release();
158             }
159         }
160 
161         public async Task<LearningResponseModel> PostHttpLearning      ↵
              (LearningRequestModel requestModel)
162         {
163             await semaphore.WaitAsync();
164             try
165             {
166                 string url = $"http://{appSettings.SystemNetwork.Ip}:  ↵
                    {appSettings.SystemNetwork.Port}/learning";
167                 LearningResponseModel responseModel = await            ↵
                    restUtils.ModelsPost<LearningRequestModel,             ↵
                    LearningResponseModel>(url, requestModel);
168                 return responseModel;
169             }
170             finally
171             {
172                 semaphore.Release();
173             }
174         }
175 
176         public async Task<LearningResponseModel> PostHttpLearningConfig ↵
              (LearningConfigModel configModel)
177         {
178             await semaphore.WaitAsync();
179             try
180             {
```

```
181                 string url = $"http://{appSettings.SystemNetwork.Ip}:
                      {appSettings.SystemNetwork.Port}/learning/config";
182                 LearningResponseModel responseModel = await
                      restUtils.ModelsPost<LearningConfigModel,
                      LearningResponseModel>(url, configModel);
183                 return responseModel;
184             }
185             finally
186             {
187                 semaphore.Release();
188             }
189         }
190
191         public async Task<List<PredictionResponseModel>>
              PostHttpPrediction(List<PredictionRequestModel>
              requestModels)
192         {
193             await semaphore.WaitAsync();
194             try
195             {
196                 List<PredictionResponseModel> responseModels = new
                      List<PredictionResponseModel>();
197                 foreach (PredictionRequestModel requestModel in
                      requestModels)
198                 {
199                     string url = $"http://
                      {appSettings.SystemNetwork.Ip}:
                      {appSettings.SystemNetwork.Port}/prediction";
200                     PredictionResponseModel responseModel = await
                      restUtils.ModelsPost<PredictionRequestModel,
                      PredictionResponseModel>(url, requestModel);
201                     responseModels.Add(responseModel);
202                 }
203                 return responseModels;
204             }
205             finally
206             {
207                 semaphore.Release();
208             }
209         }
210
211         public async Task<PredictionResponseModel>
              PostHttpPredictionConfig(PredictionConfigModel configModel)
212         {
213             await semaphore.WaitAsync();
214             try
215             {
216                 string url = $"http://{appSettings.SystemNetwork.Ip}:
                      {appSettings.SystemNetwork.Port}/prediction/config";
217                 PredictionResponseModel responseModel = await
                      restUtils.ModelsPost<PredictionConfigModel,
                      PredictionResponseModel>(url, configModel);
218                 return responseModel;
```

```
219                 }
220                 finally
221                 {
222                     semaphore.Release();
223                 }
224             }
225             */
226         }
227     }
228
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Moq;

using SampleMvvmPrism.Services.Interface;
using SampleMvvmPrism.Models.Http;
using SampleMvvmPrism.Models.Config;
using SampleMvvmPrism.Models.Csv;


namespace XTest.Services.Imple
{
    public class MainServiceMock
    {
        public Mock<IMainService> mock;
        public Dictionary<string, object> parameters;
        public Dictionary<string, object> returns;

        public IMainService Object { get { return this.mock.Object; } }

        public MainServiceMock()
        {
            this.parameters = new Dictionary<string, object>();
            this.returns = new Dictionary<string, object>();

            this.mock = new Mock<IMainService>();
            this.mock.Setup(it => it.GetAppSettings())
                .Callback(() => this.parameters["GetAppSettings"] = new Object())
                .Returns(() => (AppSettingsModel)this.GetReturn("GetAppSettings"));
        }

        public object GetParam(string targetFunc)
        {
            if (this.parameters.ContainsKey(targetFunc))
            {
                return this.parameters[targetFunc];
            }
            else
            {
                throw new Exception($"TargetFuncHasNotCalled [{targetFunc}]:メソッド呼び出し情報がありません。");
            }
        }

        public object SetReturn(string targetFunc, object value)
        {
            return this.returns[targetFunc] = value;
        }
```

```
51
52          private object GetReturn(string targetFunc)
53          {
54              if (this.returns.ContainsKey(targetFunc))
55              {
56                  return this.returns[targetFunc];
57              }
58              else
59              {
60                  throw new Exception($"TargetFuncHasNotSetReturn
                      [{targetFunc}]:メソッドのリターン値が設定されていません。");
61              }
62          }
63      }
64  }
65
```

```
1  using System.Windows;
2
3  namespace SampleMvvmPrism.Views
4  {
5      /// <summary>
6      /// Interaction logic for MainWindow.xaml
7      /// </summary>
8      public partial class MainWindow : Window
9      {
10         public MainWindow()
11         {
12             InitializeComponent();
13         }
14     }
15 }
16
```

```
<Window x:Class="SampleMvvmPrism.Views.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:prism="http://prismlibrary.com/"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/
          themes"
        prism:ViewModelLocator.AutoWireViewModel="True"
        TextElement.Foreground="{DynamicResource MaterialDesignBody}"
        Background="{DynamicResource MaterialDesignPaper}"
        TextElement.FontWeight="Medium"
        TextElement.FontSize="14"
        FontFamily="pack://application:,,,/
          MaterialDesignThemes.Wpf;component/Resources/Roboto/#Roboto"
        Title="{Binding Title}"
        Height="600" Width="800"
        Style="{StaticResource MaterialDesignWindow}"
        >
    <Grid>
        <DockPanel LastChildFill="True">
            <!-- 左Drawer を表示するためのトグルボタン -->
            <materialDesign:ColorZone
                DockPanel.Dock="Top"
                Padding="8"
                Margin="4"
                materialDesign:ElevationAssist.Elevation="Dp4"
                ClipToBounds="False"
                CornerRadius="10"
                Mode="PrimaryDark">
                <StackPanel Orientation="Horizontal">
                    <TextBlock
                       Margin="16,0,0,0"
                       VerticalAlignment="Center"
                       Text="MLOps Tool" />
                </StackPanel>
            </materialDesign:ColorZone>

            <DockPanel LastChildFill="True">
                <!-- Drawer全体のデザインを記述-->
                <materialDesign:DrawerHost
                    x:Name="DrawerHost"
                    DockPanel.Dock="Left"
                    BorderBrush="{DynamicResource MaterialDesignDivider}"
                    BorderThickness="2"
                    BottomDrawerBackground="{DynamicResource
                   SecondaryHueLightBrush}"
                    BottomDrawerCornerRadius="20 20 0 0"
                    IsLeftDrawerOpen="True">

                    <!-- 左Drawerを表示させた時のメニューとメニューを選択した時の動
                   作を記述-->
                    <materialDesign:DrawerHost.LeftDrawerContent>
                        <StackPanel Margin="16" >
                            <!-- メニュー項目１ -->
```

```
                            <Button
                                Tag="0"
                                Margin="4"
                                HorizontalAlignment="Center"
                                Content="測定機能"
                                Command="{Binding ButtonClickCommand}"
                                CommandParameter="MEASUREMENT"
                                Style="{StaticResource
                    MaterialDesignFlatButton}" />
                            <!-- メニュー項目２ -->
                            <Button
                                Tag="1"
                                Margin="4"
                                HorizontalAlignment="Center"
                                Content="学習機能"
                                Command="{Binding ButtonClickCommand}"
                                CommandParameter="LEARNING"
                                Style="{StaticResource
                    MaterialDesignFlatButton}" />
                            <Button
                                Tag="2"
                                Margin="4"
                                HorizontalAlignment="Center"
                                Content="推論機能"
                                Command="{Binding ButtonClickCommand}"
                                CommandParameter="PREDICTION"
                                Style="{StaticResource
                    MaterialDesignFlatButton}" />
                        </StackPanel>
                    </materialDesign:DrawerHost.LeftDrawerContent>
                </materialDesign:DrawerHost>

                <!-- メインコンテンツ -->
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition Height="*" />
                    </Grid.RowDefinitions>

                    <ContentControl
                        Grid.Row="1"
                        prism:RegionManager.RegionName="ContentRegion" />
                </Grid>
            </DockPanel>
        </DockPanel>
    </Grid>
</Window>
```

```
1  using Prism.Commands;
2  using Prism.Events;
3  using Prism.Mvvm;
4  using Prism.Regions;
5  using System;
6  using System.Windows;
7  using SampleMvvmPrism.Views;
8  using SampleMvvmPrism.Services.Interface;
9
10 namespace SampleMvvmPrism.ViewModels
11 {
12     public class MainWindowViewModel : BindableBase
13     {
14         private string title = "Prism Application";
15         private string inputMessage = "";
16         private string outputMessage = "";
17         private readonly IRegionManager regionManager;
18         private readonly IEventAggregator eventAggregator;
19         private readonly IMainService mainService;
20         private NavigationParameters navigationParameters;
21
22         public DelegateCommand<string> ButtonClickCommand { get; set; }
23
24         public string Title
25         {
26             get { return this.title; }
27             set { SetProperty(ref this.title, value); }
28         }
29         public string InputMessage
30         {
31             get { return this.inputMessage; }
32             set { SetProperty(ref this.inputMessage, value); }
33         }
34         public string OutputMessage
35         {
36             get { return this.outputMessage; }
37             set { SetProperty(ref this.outputMessage, value); }
38         }
39
40         public MainWindowViewModel(IRegionManager regionManager,
             IEventAggregator eventAggregator, IMainService mainService)
41         {
42             this.regionManager = regionManager;
43             this.eventAggregator = eventAggregator;
44             this.mainService = mainService;
45             this.navigationParameters = new NavigationParameters()
               {{ nameof(IMainService), this.mainService }};
46             ButtonClickCommand = new DelegateCommand<string>(OnEvent);
47         }
48
49         private void OnEvent(string message)
50         {
51             if (message == "MEASUREMENT")
```

```
52              {
53                  // MessageBox.Show($"測定画面を開始します");
54                  regionManager.RequestNavigate("ContentRegion", nameof
                      (MeasurementView), this.navigationParameters);
55                  this.OutputMessage = $"{message} : Start";
56              }
57              else if (message == "LEARNING")
58              {
59                  // MessageBox.Show($"学習画面を開始します");
60                  regionManager.RequestNavigate("ContentRegion", nameof
                      (LearningView), this.navigationParameters);
61                  this.OutputMessage = $"{message} : Start";
62              }
63              else if (message == "PREDICTION")
64              {
65                  // MessageBox.Show($"推論画面を開始します");
66                  regionManager.RequestNavigate("ContentRegion", nameof
                      (PredictionView), this.navigationParameters);
67                  this.OutputMessage = $"{message} : Start";
68              }
69              else
70              {
71                  throw new NotImplementedException();
72              }
73          }
74
75          /// <summary>Viewを表示した後呼び出されます。</summary>
76          /// <param name="navigationContext">Navigation Requestの情報を表す
              NavigationContext。</param>
77          public void OnNavigatedTo(NavigationContext navigationContext)
78          {
79          }
80
81          /// <summary>表示するViewを判別します。</summary>
82          /// <param name="navigationContext">Navigation Requestの情報を表す
              NavigationContext。</param>
83          /// <returns>表示するViewかどうかを表すbool。</returns>
84          public bool IsNavigationTarget(NavigationContext
              navigationContext)
85          {
86              return true;
87          }
88
89          /// <summary>別のViewに切り替わる前に呼び出されます。</summary>
90          /// <param name="navigationContext">Navigation Requestの情報を表す
              NavigationContext。</param>
91          public void OnNavigatedFrom(NavigationContext navigationContext)
92          {
93              return;
94          }
95      }
96  }
97
```

```
1  using Moq;
2  using Prism.Events;
3  using Prism.Regions;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10
11 namespace XTest.Prism
12 {
13     public class RegionManagerMock
14     {
15         public Mock<IRegionManager> mock;
16         public Dictionary<string, object> parameters;
17         public Dictionary<string, object> returns;
18
19         public IRegionManager Object { get { return
             this.mock.Object; } }
20
21         public RegionManagerMock()
22         {
23             this.parameters = new Dictionary<string, object>();
24             this.returns = new Dictionary<string, object>();
25
26             this.mock = new Mock<IRegionManager>();
27             this.mock.Setup(it => it.RequestNavigate(It.IsAny<string>(),
                  It.IsAny<string>(), It.IsAny<NavigationParameters>()))
28                  .Callback<string, string, NavigationParameters>((s1, s2,
                     navi1) => this.parameters["RequestNavigate"] = (s1,
                    s2, navi1));
29         }
30
31         public object GetParam(string targetFunc)
32         {
33             if (this.parameters.ContainsKey(targetFunc))
34             {
35                 return this.parameters[targetFunc];
36             }
37             else
38             {
39                 throw new Exception($"TargetFuncHasNotCalled
                     [{targetFunc}]:メソッド呼び出し情報がありません。");
40             }
41         }
42
43         public object SetReturn(string targetFunc, object value)
44         {
45             return this.returns[targetFunc] = value;
46         }
47
48         private object GetReturn(string targetFunc)
```

```
49          {
50              if (this.returns.ContainsKey(targetFunc))
51              {
52                  return this.returns[targetFunc];
53              }
54              else
55              {
56                  throw new Exception($"TargetFuncHasNotSetReturn
                      [{targetFunc}]:メソッドのリターン値が設定されていません。");
57              }
58          }
59      }
60  }
61
```

```cs
using System;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using System.Net.Http;
using System.Threading;


namespace SampleMvvmPrism.Services.Utils
{

    /// <summary>
    /// REST APIを呼び出すための基本的なユーティリティクラス
    /// </summary>
    public class RestUtils
    {
        public string lastReason = null;
        private static readonly TimeSpan TIMEOUT_MILLIS = TimeSpan.FromMilliseconds(3000);

        public RestUtils()
        {
        }

        /// <summary>APIポスト(Model型)</summary>
        /// <param name="REQ">リクエストモデルの型</param>
        /// <param name="RES">レスポンスモデルの型</param>
        /// <param name="requestModel">リクエストモデルのオブジェクト</param>
        /// <param name="url">完全なURL</param>
        /// <returns>resultModel</returns>
        public async Task<RES> ModelsPost<REQ, RES>(string url, REQ requestModel)
        {
            string data = JsonSerializer.Serialize(requestModel);
            HttpResponseMessage response = await PostJsonAsync(url, data);
            var stream = await response.Content.ReadAsStreamAsync();
            RES? resultModel = await JsonSerializer.DeserializeAsync<RES>(stream);
            if (resultModel == null) { throw new Exception("Result Model Is null"); }
            return resultModel;
        }

        /// <summary>APIポスト(Model型)</summary>
        /// <param name="REQ">リクエストモデルの型</param>
        /// <param name="RES">レスポンスモデルの型</param>
        /// <param name="requestModel">リクエストモデルのオブジェクト</param>
        /// <param name="url">完全なURL</param>
        /// <returns>resultModel</returns>
        public async Task<RES> ModelsGet<RES>(string url)
```

```
47          {
48              HttpResponseMessage response = await GetJsonAsync(url);
49              var stream = await response.Content.ReadAsStreamAsync();
50              RES? resultModel = await
                  JsonSerializer.DeserializeAsync<RES>(stream);
51              if (resultModel == null) { throw new Exception("Result
                  Model Is null"); }
52              return resultModel;
53          }
54
55          /// <summary>APIプット(Model型)</summary>
56          /// <param name="REQ">リクエストモデルの型</param>
57          /// <param name="RES">レスポンスモデルの型</param>
58          /// <param name="requestModel">リクエストモデルのオブジェクト</
              param>
59          /// <param name="url">完全なURL</param>
60          /// <returns>resultModel</returns>
61          public async Task<RES> ModelsPut<REQ, RES>(string url, REQ
              requestModel)
62          {
63              string data = JsonSerializer.Serialize(requestModel);
64              HttpResponseMessage response = await PutJsonAsync(url,
                  data);
65              var stream = await response.Content.ReadAsStreamAsync();
66              RES? resultModel = await
                  JsonSerializer.DeserializeAsync<RES>(stream);
67              if (resultModel == null) { throw new Exception("Result
                  Model Is null"); }
68              return resultModel;
69          }
70
71          /// <summary>APIポスト(Model型)</summary>
72          /// <param name="RES">レスポンスモデルの型</param>
73          /// <param name="url">完全なURL</param>
74          /// <returns>bool</returns>
75          public async Task<bool> ModelsDelete<RES>(string url)
76          {
77              HttpResponseMessage response = await DeleteAsync(url);
78              var stream = await response.Content.ReadAsStreamAsync();
79              RES? resultModel = await
                  JsonSerializer.DeserializeAsync<RES>(stream);
80              if (resultModel == null) { throw new Exception("Result
                  Model Is null"); }
81              return true;
82          }
83
84          /// <summary>APIポスト(string型)</summary>
85          /// <param name="url">完全なURL</param>
86          /// <param name="jsonData">リクエストボディーの文字列</param>
87          /// <returns>HttpResponseMessage response</returns>
88          private async Task<HttpResponseMessage> PostJsonAsync(string
              url, string jsonData)
89          {
```

```
90              using (var httpClientHandler = new HttpClientHandler())
91              {
92                                                                                  ↵
    httpClientHandler.ServerCertificateCustomValidationCallback = (message, ↵
     cert, chain, errors) => { return true; };
93                  using (var client = new HttpClient(httpClientHandler)   ↵
                      { Timeout = TIMEOUT_MILLIS })
94                  {
95                      HttpContent content = new StringContent(jsonData,   ↵
                    Encoding.UTF8, "application/json");
96                      HttpResponseMessage response = await                ↵
                    client.PostAsync(url, content);
97                      lastReason = response.ReasonPhrase;
98                      response.EnsureSuccessStatusCode();
99                      return response;
100                 }
101             }
102         }
103
104         /// <summary>APIポスト(string型)</summary>
105         /// <param name="url">完全なURL</param>
106         /// <returns>HttpResponseMessage response</returns>
107         private async Task<HttpResponseMessage> GetJsonAsync(string     ↵
              url)
108         {
109             using (var httpClientHandler = new HttpClientHandler())
110             {
111                                                                                 ↵
    httpClientHandler.ServerCertificateCustomValidationCallback = (message, ↵
     cert, chain, errors) => { return true; };
112                 using (var client = new HttpClient(httpClientHandler)   ↵
                      { Timeout = TIMEOUT_MILLIS })
113                 {
114                     HttpResponseMessage response = await                ↵
                    client.GetAsync(url);
115                     lastReason = response.ReasonPhrase;
116                     response.EnsureSuccessStatusCode();
117                     return response;
118                 }
119             }
120         }
121
122         /// <summary>APIポスト(string型)</summary>
123         /// <param name="url">完全なURL</param>
124         /// <param name="jsonData">リクエストボディーの文字列</param>
125         /// <returns>HttpResponseMessage response</returns>
126         private async Task<HttpResponseMessage> PutJsonAsync(string     ↵
              url, string jsonData)
127         {
128             using (var httpClientHandler = new HttpClientHandler())
129             {
130                                                                                 ↵
    httpClientHandler.ServerCertificateCustomValidationCallback = (message, ↵
```

```
    cert, chain, errors) => { return true; };
131                 using (var client = new HttpClient(httpClientHandler)
                      { Timeout = TIMEOUT_MILLIS })
132                 {
133                     HttpContent content = new StringContent(jsonData,
                    Encoding.UTF8, "application/json");
134                     HttpResponseMessage response = await
                    client.PutAsync(url, content);
135                     lastReason = response.ReasonPhrase;
136                     response.EnsureSuccessStatusCode();
137                     return response;
138                 }
139             }
140         }
141
142         /// <summary>APIデリート(string型)</summary>
143         /// <param name="url">完全なURL</param>
144         /// <returns>HttpResponseMessage response</returns>
145         private async Task<HttpResponseMessage> DeleteAsync(string url)
146         {
147             using (var httpClientHandler = new HttpClientHandler())
148             {
149
    httpClientHandler.ServerCertificateCustomValidationCallback = (message,
     cert, chain, errors) => { return true; };
150                 using (var client = new HttpClient(httpClientHandler)
                      { Timeout = TIMEOUT_MILLIS })
151                 {
152                     HttpResponseMessage response = await
                    client.DeleteAsync(url);
153                     lastReason = response.ReasonPhrase;
154                     response.EnsureSuccessStatusCode();
155                     return response;
156                 }
157             }
158         }
159     }
160 }
```

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>net8.0-windows7.0</TargetFramework>
    <UseWPF>true</UseWPF>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="CsvHelper" Version="33.0.1" />
    <PackageReference Include="MaterialDesignThemes" Version="5.1.0" />
    <PackageReference Include="Microsoft.Extensions.Configuration"
      Version="9.0.0" />
    <PackageReference Include="Microsoft.Extensions.Configuration.Binder"
      Version="9.0.0" />
    <PackageReference Include="Microsoft.Extensions.Configuration.Json"
      Version="9.0.0" />
    <PackageReference Include="Prism.Unity" Version="8.1.97" />
  </ItemGroup>
  <ItemGroup>
    <Compile Update="Views\PredictionView.xaml.cs">
      <SubType>Code</SubType>
    </Compile>
  </ItemGroup>
  <ItemGroup>
    <None Update="appsettings.json">
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
  </ItemGroup>
</Project>
```

```
1  using Xunit;
2  using Moq;
3  using System;
4  using System.Windows;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 using SampleMvvmPrism.ViewModels;
11 using SampleMvvmPrism.Models.Config;
12 using SampleMvvmPrism.Models.Http;
13
14
15 namespace XTest.ViewModels
16 {
17     public class TestLearningViewModel : AbstractTestViewModel
18     {
19         private readonly LearningViewModel targetViewModel;
20
21         public TestLearningViewModel()
22         {
23             this.targetViewModel = new LearningViewModel(
24                 base.regionManagerMock.Object,
25                 base.eventAggregatorMock.Object,
26                 base.mainServiceMock.Object,
27                 base.learningServiceMock.Object);
28         }
29
30         [Theory]
31         [InlineData("CONFIG")]
32         [InlineData("HTTP")]
33         public void TestOnEventButton1(string message)
34         {
35             switch (message)
36             {
37                 case "CONFIG":
38                     base.learningServiceMock.returns
                      ["ReadConfigLearning"] = new LearningConfigModel();
39                     base.learningServiceMock.returns
                      ["PostHttpLearningConfig"] = Task.FromResult(new
                      LearningResponseModel());
40                     this.targetViewModel.OnClickButton.Execute
                      ("CONFIG");
41                     var actual1 = (LearningConfigModel)
                      base.learningServiceMock.parameters
                      ["PostHttpLearningConfig"];
42                     var expect1 = new LearningConfigModel();
43                     Assert.Equivalent(expect1, actual1);
44                     break;
45                 case "HTTP":
46                     base.learningServiceMock.returns["PostHttpLearning"]
                       = Task.FromResult(new LearningResponseModel());
```

```
47                      this.targetViewModel.OnClickButton.Execute("HTTP");
48                      var actual2 = (LearningRequestModel)                ↵
                      base.learningServiceMock.parameters                   ↵
                      ["PostHttpLearning"];
49                      var expect2 = new LearningRequestModel();
50                      Assert.Equivalent(expect2, actual2);
51                      break;
52                  default:
53                      throw new NotImplementedException();
54              }
55          }
56      }
57  }
58
```

```cs
using Xunit;
using Moq;
using System;
using System.Windows;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using SampleMvvmPrism.ViewModels;
using SampleMvvmPrism.Models.Config;
using SampleMvvmPrism.Models.Http;
using SampleMvvmPrism.Services.Interface;
using Prism.Regions;


namespace XTest.ViewModels
{
    public class TestMainWindowViewModel : AbstractTestViewModel
    {
        private readonly MainWindowViewModel targetViewModel;

        public TestMainWindowViewModel()
        {
            this.targetViewModel = new MainWindowViewModel(
                base.regionManagerMock.Object,
                base.eventAggregatorMock.Object,
                base.mainServiceMock.Object);
        }

        [Theory]
        [InlineData("MEASUREMENT")]
        [InlineData("LEARNING")]
        [InlineData("PREDICTION")]
        public void TestOnEvent(string message)
        {
            switch (message)
            {
                case "MEASUREMENT":
                    this.targetViewModel.ButtonClickCommand.Execute("MEASUREMENT");
                    var actual1 = ((string, string, NavigationParameters))base.regionManagerMock.parameters["RequestNavigate"];
                    var expect1 = (("ContentRegion", "MeasurementView", new NavigationParameters() { { nameof(IMainService), base.mainServiceMock.Object } }));
                    Assert.Equivalent(expect1.Item2, actual1.Item2);
                    break;
                case "LEARNING":
                    this.targetViewModel.ButtonClickCommand.Execute("LEARNING");
                    var actual2 = ((string, string, NavigationParameters))
```

```
                    base.regionManagerMock.parameters["RequestNavigate"];
47                    var expect2 = (("ContentRegion", "LearningView", new ↵
                     NavigationParameters() { { nameof(IMainService),      ↵
                    base.mainServiceMock.Object } }));
48                    Assert.Equivalent(expect2.Item2, actual2.Item2);
49                    break;
50                case "PREDICTION":
51                    this.targetViewModel.ButtonClickCommand.Execute      ↵
                    ("PREDICTION");
52                    var actual3 = ((string, string,                      ↵
                    NavigationParameters))                                ↵
                    base.regionManagerMock.parameters["RequestNavigate"];
53                    var expect3 = (("ContentRegion", "PredictionView",   ↵
                    new NavigationParameters() { { nameof(IMainService),  ↵
                    base.mainServiceMock.Object } }));
54                    Assert.Equivalent(expect3.Item2, actual3.Item2);
55                    break;
56                default:
57                    throw new NotImplementedException();
58            }
59        }
60    }
61 }
62
```

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net8.0-windows7.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>

    <IsPackable>false</IsPackable>
    <IsTestProject>true</IsTestProject>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.12.0" />
    <PackageReference Include="Moq" Version="4.20.72" />
    <PackageReference Include="xunit" Version="2.9.2" />
    <PackageReference Include="xunit.runner.visualstudio" Version="3.0.0-
      pre.49">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers;
        buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="coverlet.collector" Version="6.0.2">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers;
        buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
  </ItemGroup>

  <ItemGroup>
    <Folder Include="Services\Utils\" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\SampleMvvmPrism\SampleMvvmPrism.csproj" />
  </ItemGroup>

</Project>
```