GETTING STARTED WITH THE PAY AT TABLE API SAMPLE

Contents

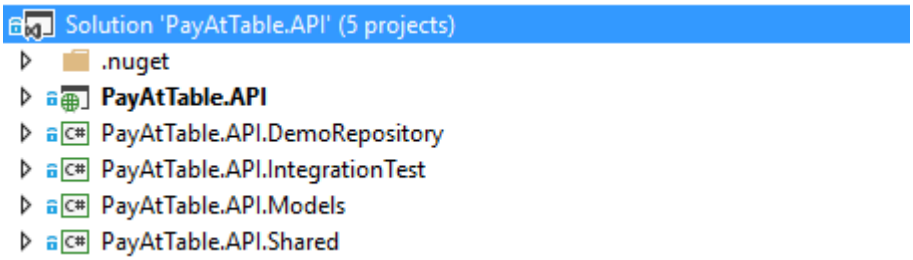
Overview

The sample is written in ASP.NET WEBAPI2. It requires Visual Studio 2013 (the Community edition is available for free to small teams).

You can download the source from https://github.com/pceftpos/pay-at-table

If using this project in a production environment, it is suggested that POS code should be contained within the custom data repository implementations. This will allow you to take advantage of future updates such as OWIN self-hosting and ASP.NET MVC 6 + DNX for OSX/Linux support.

Projects



Controller	Method
PayAtTable.API	Main project which contains the API implementation.
PayAtTable.API.DemoRepository	A sample data repository implementation.
PayAtTable.API.IntegrationTest	Standalone integration test
PayAtTable.API.Models	Contains object definitions and data repository interfaces
PayAtTable.API.Shared	Common code shared between all other projects

Controllers

The controllers define the API entry points in the main PayAtTable.API project. Each controller encompasses a logical group of methods from the API specification.

Controller	Method
TablesController.cs	GET ~/pat/tables
OrdersController.cs	GET ~/pat/tables/{table-id}/orders GET ~/pat/orders/{order-id} GET ~/pat/orders/{order-id}/receipt
SettingsController.cs	GET ~/pat/settings
TendersController.cs	POST ~/pat/tenders PUT ~/pat/tenders/{tender-id}
EFTPOSController.cs	POST ~/pat/eftpos/commands

Data Repositories

Each controller accepts one or more data repository interfaces as constructor parameters. The sample code utilises dependency injection to construct implementations of these data repository interfaces.

For example, when the OrdersController is constructed the server will attempt to find an instance of IOrdersRepository in the IoC container. If one is not found it will be constructed.

```
public OrdersController(IOrdersRepository ordersRepository)
{
        _ordersRepository = ordersRepository;
}
```

CODE 1 ORDERSCONTROLLER.CS

The data repository implementations are registered in the ***WebApiConfig.Register(...)*** method. The default behaviour is to attempt to load the assemblies from the ***DataRepositoryAssembly*** setting.

```
// Find assembly from path defined in web.config
var path = PayAtTable.API.Properties.Settings.Default.DataRepositoryAssembly;
var mappedPath = System.Web.Hosting.HostingEnvironment.MapPath(path);
Assembly assembly = Assembly.LoadFrom(mappedPath);
```

CODE 2 WEBAPICONFIG.CS

The ***DataRepositoryAssembly*** setting is located in the Web.config. The path must be relative to the base directory of the website.

By default, the project it using the ***DemoRepository.dll*** file. You can change the behaviour of the Pay at Table server by building a custom implementation of the DemoRepository and updating the ***DataRepositoryAssembly*** setting to point to your own dll.

```
<applicationSettings>
        <PayAtTable.API.Properties.Settings>
                <setting name="DataRepositoryAssembly" serializeAs="String">
                        <value>~\bin\PayAtTable.API.DemoRepository.dll</value>
                </setting>
        </PayAtTable.API.Properties.Settings>
</applicationSettings>
```

CODE 3 WEB.CONFIG

Logging

The Pay at Table project utilises the Common.Logging framework. Custom repository assemblies can participate in the log managed by the Pay at Table project by importing the Common.Logging assembly and making a call to the LogManager.

```
protected static readonly Common.Logging.ILog log = Common.Logging.LogManager.GetLogger(
        System.Reflection.MethodBase.GetCurrentMethod().DeclaringType);

public void TestLog()
{
        log.InfoEx("Log test...");
}
```

CODE 4 ORDERSREPOSITORYDEMO.CS

The default logging provider is Log4Net, which is configured in the ***log4net*** section of the projects ***web.config*** file.

Integration Testing

An integration test client is included in the project which tests an API implementation.

Available parameters:

Parameter	Description
baseUri	The base URI of the server. e.g. http://localhost/api/
key	The key parameter to include in each request if API key authentication is enabled.
debugTrace	Indicates if debug tracing should be enabled.
delayEFTPOSCommands	Indicates if the test client should delay for 1 second between EFTPOS command requests.
help	Show the available commands

Example

.\PATAPI.IntegrationTest.exe -baseUri="http://localhost/api" --debugTrace

Deployment

TODO