# Table of Contents

# Welcome to the ToothpicExchange manual.

This HTML website contains the relevant information to setup, to use, and to further develop the ToothpicExchange system which has been created for Toothpic/OralEye Ltd.

This documentation has been created with DocFx (console version 2.23). It contains static information alongside dynamic content generated from ToothpicExchange's source code. The source code generated content utilises C# XML Documentation Comments

## How to use this website:

There are two main sections.

The system overview is best for setting up and configuring the plugin. The API documentation may seem daunting at first, however it is only necessary for development of the source code.

### System Overview

This section contains general information about the system. Including:

1. General Introduction
2. How to Setup the Plugin
3. How to use the Plugin
4. How to setup the developer environment

The information in this section is static content contained in the project's markdown files (.md)

### API Documentation

This section contains a comprehensive reference of the source code. It details the classes, their methods, properties and more. It can be used in tandem with the source code, and it can be updated as the source code changes.

Much of this content is created from the XML tages within the source code. In order to update this content, the XML tags should be edited in the source code and the project rebuilt.

With DocFx.Console configured in Visual Studio, the site will update itself everytime the project is built.

The HTML files are output to a `_site` folder within the project.

Much of the content contained in this section is also visible within Visual Studio's *IntelliSense*. This is extremely convenient as the relevant documentation appears in Visual Studio as you type, there is no need to open a separate reference.

It should be noted that DocFx currently do not support publishing of `private` defined methods or attributes. Thus, these will not appear on the HTML site however they are still documented in Visual Studio through XML comments, and they will show up in IntelliSense.

# Introduction

ToothpicExchange is a system which allows the exchange of patient demographic information between Toothpic's API and a custom plugin for OpenDental

## Information

- Written in C# (7.2) .NET (4.5) and Windows Forms
- Written with OpenDental 17.2 source code and plugin framework

## Plugin Installation

Click here for instructions on how to install the OpenDental plugin.

## Dependencies

- Microsoft .NET version 4.5
- C# version 7.2
- RestSharp version 105.2.3
- OpenDental 17.2
- (Visual Studio 2017 or other IDE)
- Json-Server 0.12.1 for test environment
- DocFx.console version 2.23 for documentation

## Folder Contents

```
- Project source code
- Documentation
- Compiled DLL ready to go
- Screenshots
- OpenDental source code
```

# Plugin Setup

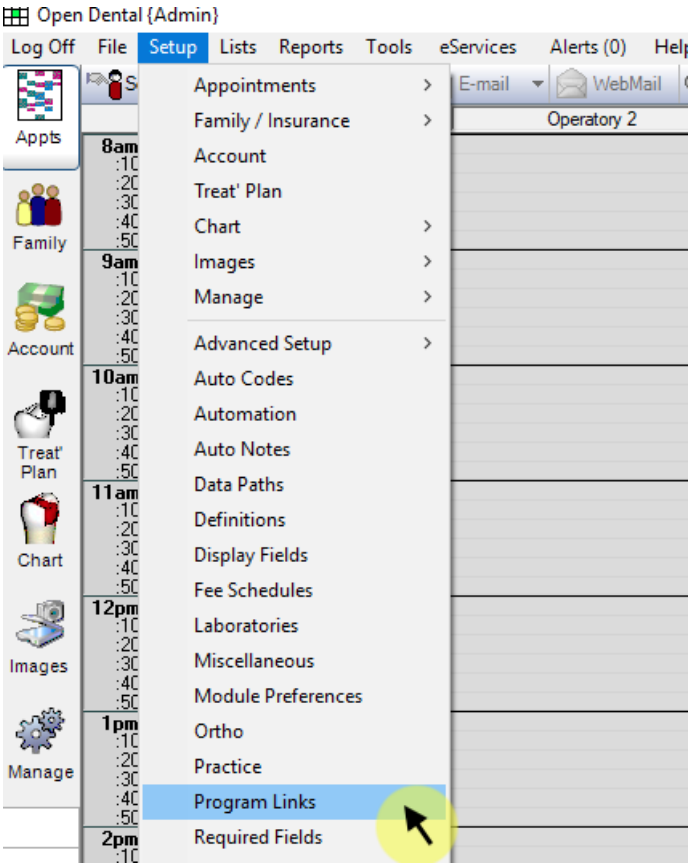
## Source Code

The source code is included with the project. This can be altered, however in order to install the plugin in OpenDental, it must be compiled as a `.dll` file. The file *must* be called `ToothpicExchange.dll`. If no modifications are required, a pre-compiled version is included.

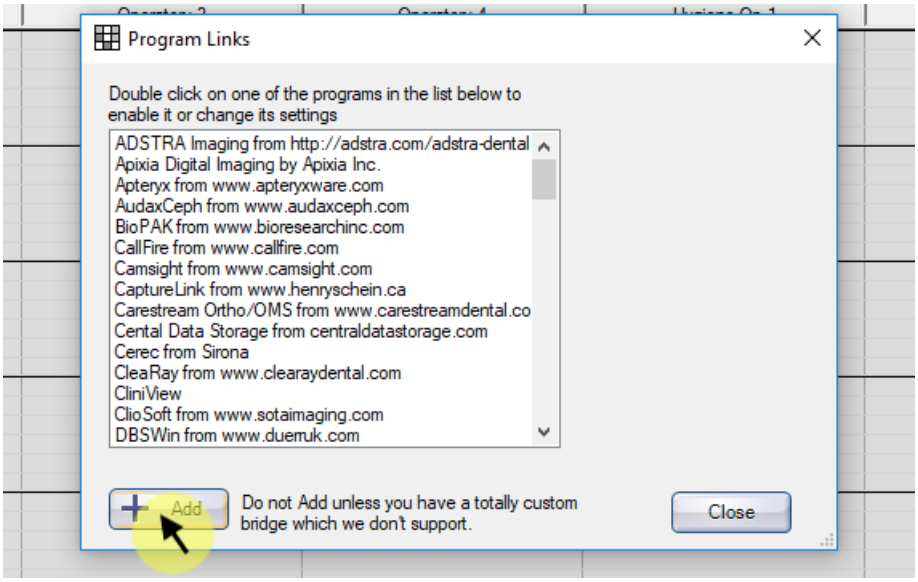
## Setup in Open Dental

1. If not already compiled, the sourcecode must be compiled into a file called `ToothpicExchange.dll`
2. ToothpicExchange.dll must be copied to OpenDental's folder e.g `C:\Program Files\Open Dental\`

3. In the OpenDental main menu, click `Setup` > `Program Links`.



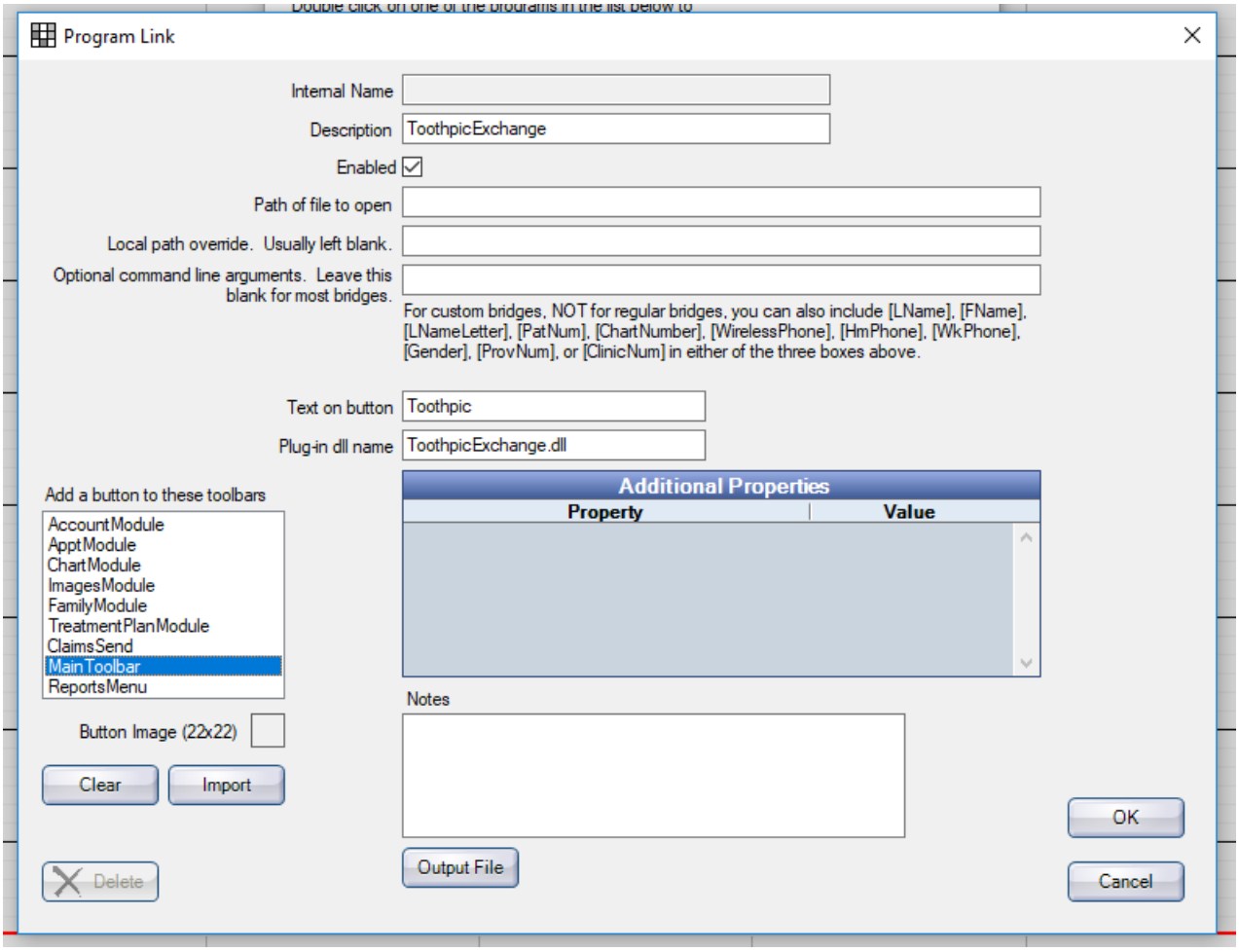

4. A new window will open with a list of Program Links already established, click the `Add` button in the bottom left hand corner.




5. A new window will open to enter the plugin settings. Input the following:

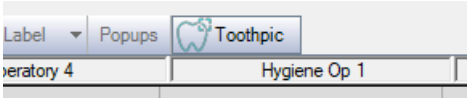
```
Description:        ToothpicExchange
Enabled:            [Checked]
Text on Button:     Toothpic
Plug-in dll name:   ToothpicExchange.dll
```

Ensure the option `Main Toolbar` is selected in the section `Add a button to these toolbars`.




6. Save the settings.

7. The Toothpic plugin will now be added to the Main Toolbar in OpenDental.



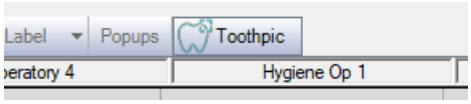

For more information, consult the OpenDental plugin framework page.

# Using the Plugin

The plugin is relatively straightforward. It is accessed through the OpenDental toolbar which is accessible in every module. The plugin offers functionality to import or export patient information to/from OpenDental and Toothpic.

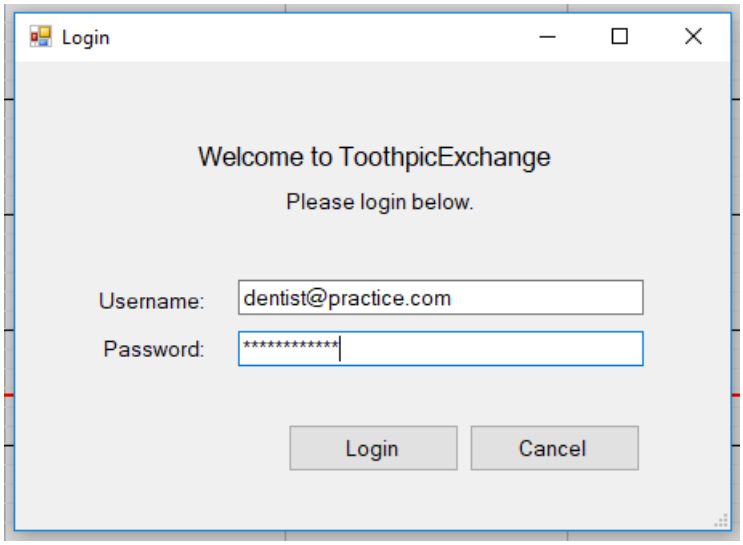
## Opening the Plugin

Once the plugin has been installed, simply click the Toothpic button in the main toolbar of OpenDental.



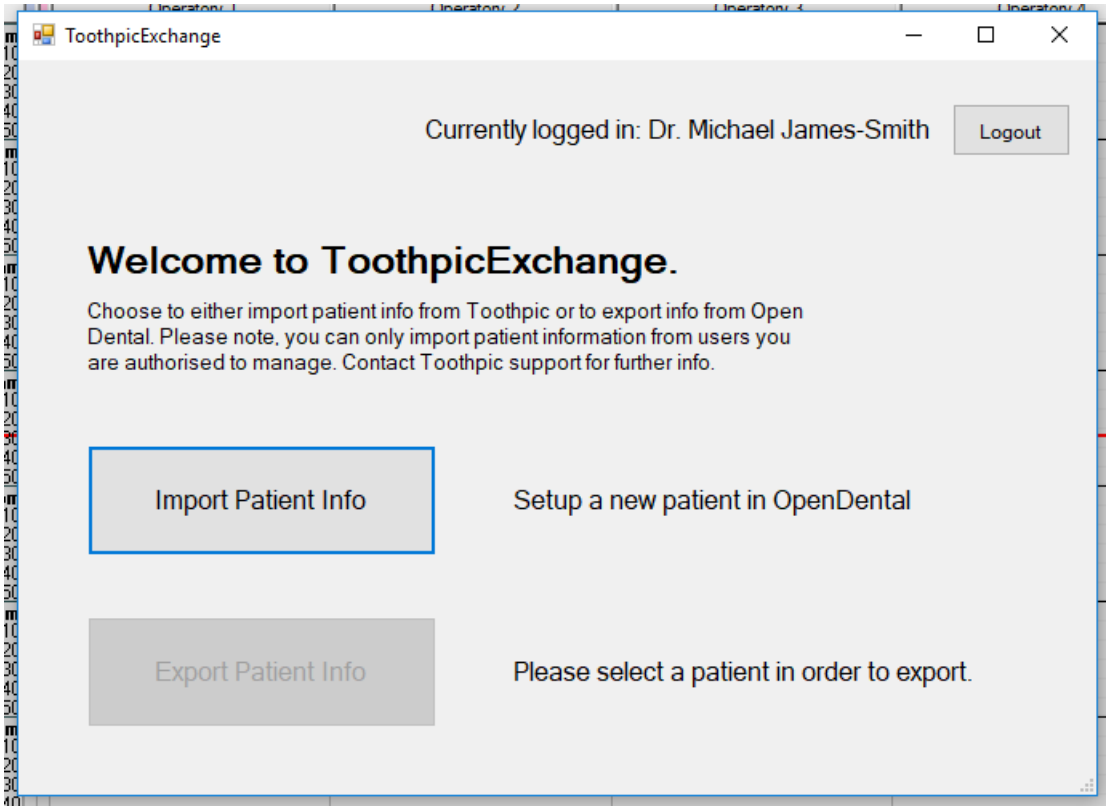

## Logging in

The plugin will prompt the practice to login with their Toothpic `username` and `password`.



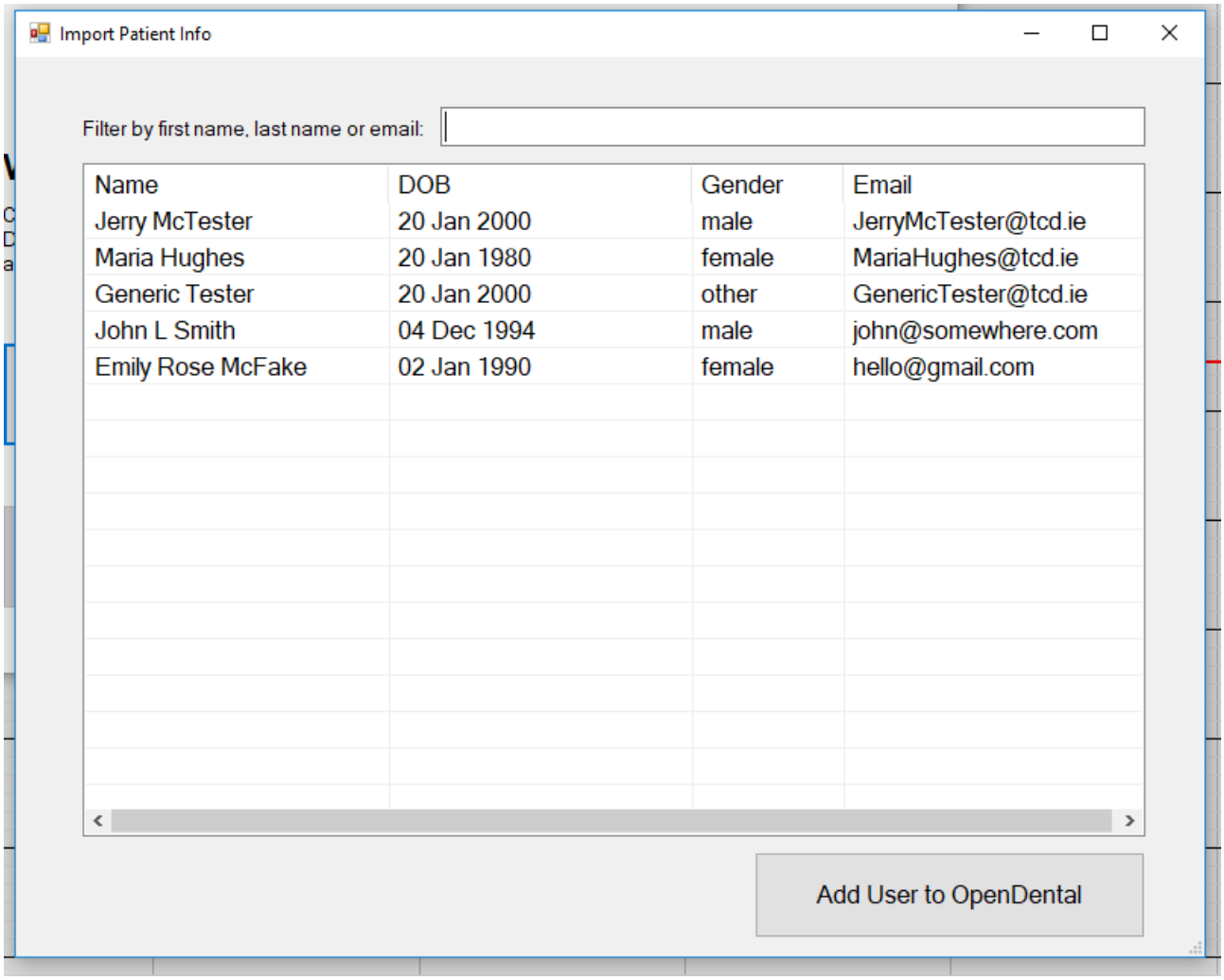

## Homepage

Once the practice has been authenticated, ToothpicExchange will launch. This is where the practice can either import or export patient information.

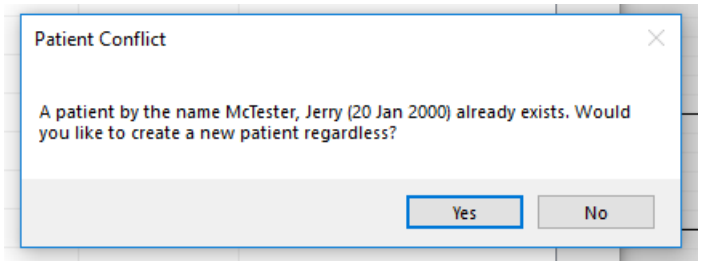



## Importing Info

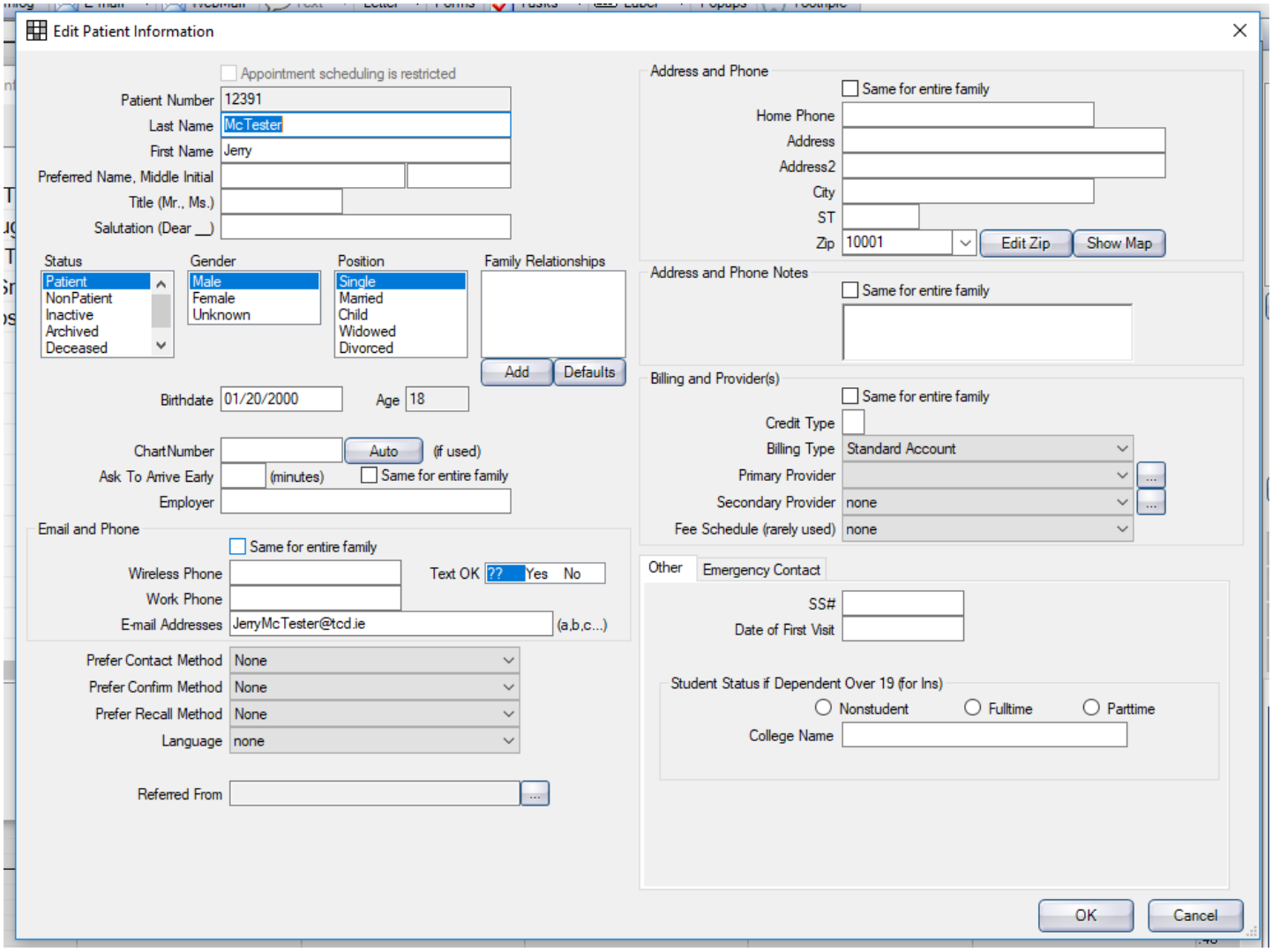
In order to import patient information, simply click the `Import Patient Info` button to begin. A new window will open with a list of users available to import. These users are loaded from Toothpic, and thus only users who the practice are authorised to view will appear on the list. There is a search box at the top of the form to filter users by first name, last name or by email address.




Select the user to import, and press `Add User to OpenDental`. If a user with the same name and date of birth already exists in OpenDental, the software will warn about a patient conflict. The practice may review this information and choose to proceed if they are not duplicates.

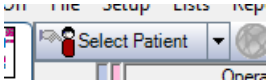



The patient edit form in OpenDental will now launch, and the practice may insert additional information about the patient. To save the information, click `OK`, otherwise `Cancel`.

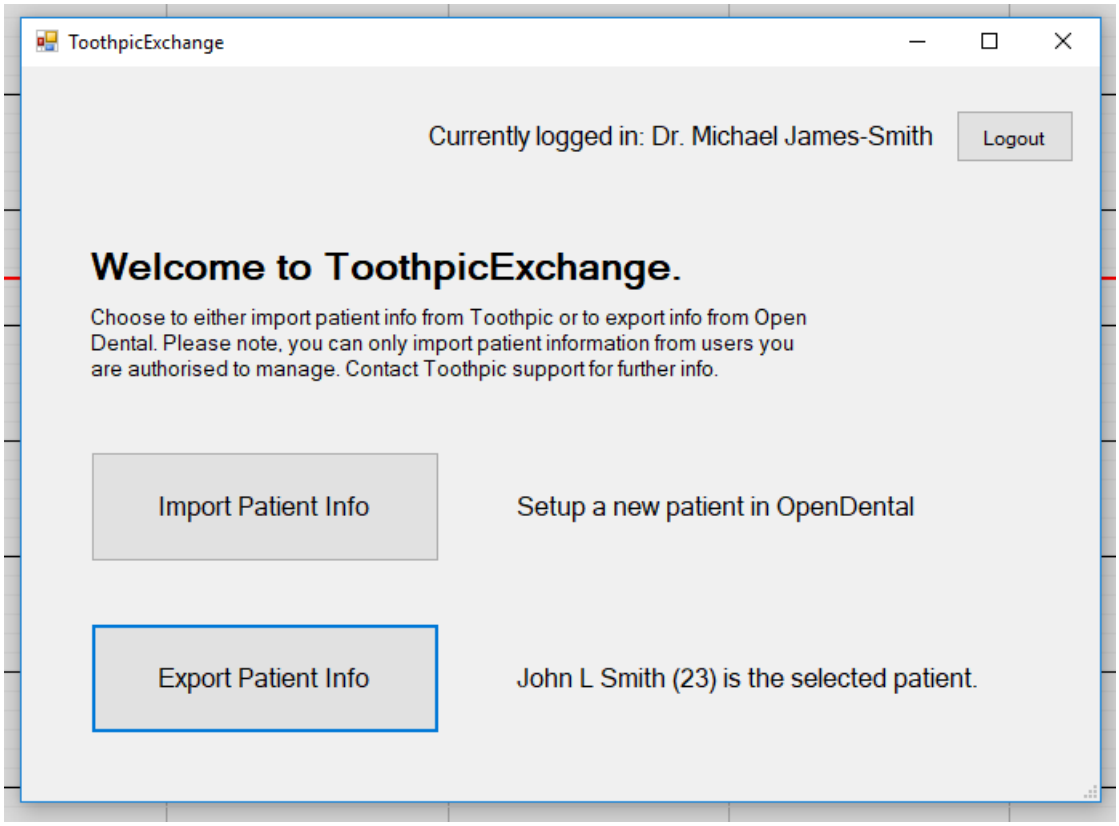



# Exporting Info

Return to the homepage of the ToothpicExchange plugin. A user must be **selected** in OpenDental before proceeding with an export. To select a patient, the practice must click the `Select Patient` button in the top left hand corner.




Once a patient has been selected, the option to `Export Patient Info` will now become available. Click it to export.




This will send the patient to Toothpic's API and will advise the practice whether this was successful or not.

# Setting up the environment

The technical environment to set up for the plugin can be quite complex. It is fairly straightforward to load the plugin directly to an instance of OpenDental (e.g. the trial version). These instructions outline that process. However, in order to setup a development environment you must follow the steps below.

## Visual Studio Installation

It is advisable to install Visual Studio IDE 2017 with a minimum .NET 4.5 and C# 7.2

Once this is done, the root `ToothpicExchange folder (the folder directly containing the` ToothpicExchange.sln` file) should be copied or set up in a new folder. This new folder will contain all of the plugin source code.

## OpenDental Developer Installation

The OpenDental source code must now be downloaded. This is a two step process.

1. The first step involves downloading the trial version of OpenDental and installing it. This installs all of OpenDental's additional dependencies and sets up the MySQL server locally. Follow the OpenDental instructions for advice on how to install OpenDental locally.

2. The next step is to actually download the source code. This can only be done through OpenDental's version control system, Subversion. Please consult this webpage for instructions how to download and set up OpenDental's source code. It is advised to store the OpenDental source code in the same HEAD folder as the ToothpicExchange folder.

Now, open the `ToothpicExchange.sln` solution file to begin setting up the technical environment.

In the Visual Studio Solution Explorer (top right hand side) there should be three projects:

- OpenDental
- OpenDentBusiness
- ToothpicExchange

1. It is likely that the two OpenDental packages will be indicated as "missing". If that is the case, you must right click and remove them from the solution.

2. Next, you must right click on the solution and click `Add` > `Existing Project`

   Browse to the following and add them: `../OpenDental/OpenDental.csproj` and `../OpenDentBusiness/OpenDentBusiness.csproj`

3. Next, right click on the ToothpicExchange project in Solution Explorer and click `Add` > `Reference`.

   On the right hand side click `Projects` > `Solution` and enable both OpenDental and OpenDentBusiness.

4. Now right click on the ToothpicExchange project in Solution Explorer and click `Properties`.

   Select `Build Events` on the left hand side, and edit the "Post-Build event command line". This should reference to a batch file called CopyDllToOd.bat located in the project folder.

   You will need to alter the absolute path in the properties so that it points to the correct file. This file takes care of transferring Visual Studio's DLL build of the plugin into the required folder in the OpenDental debug bin.

5. Now, open the `CopyDllToOd.bat` file in the project folder, and edit the absolute path to match the OpenDental `bin/debug` folder.

6. Right click on the OpenDental project in Solution Explorer and set it as the startup project.

7. Finally, try build the project and see if it runs. It could take a while initially.

For troubleshooting, and further info, consult the OpenDental plugin setup page

# Setting up JSON-Server

1. Setup a folder to contain the server files and navigate to it in the Command Prompt.
2. Json-server runs on Node.js, so you will need to install the Node.js package manager (npm).

   Navigate here to download and install npm
3. Now you can utilise npm to install json-server. Run the following in the Command Prompt:

   ```
   npm install -g json-server
   ```
4. Create a `db.json` file, there is a template included in the project folder.
5. Run the following command:

   ```
   json-server db.json
   ```

   This will start a new local server operating on port 3000. It will respond to GET requests for any objects contained in the `db.json` file, and it will add any new objects to the `db.json` with a POST request.

For further information, consult the json-server github.

# Producing Documentation

Navigate to the DocFx website for information on how to download and setup the latest version of DocFx.

This project utilises the DocFx.console variant which can be downloaded in Visual Studio through the NuGet Package Manager.

There are `docfx.json` and `docfxPDF.json` files in the project directory. These files are used to specify the DocFx configuration. It is recommended these are not changed, unless you are familiar with the DocFx environment

# Configuring the Plugin

There are a few minor configuration items to tackle before deploying the plugin.

### API Base Url

In the `APISettings.settings` file contained within the project folder, there is a property labelled `ToothpicAPIBaseURI`. This value is the API's base URL.

This property is referenced in `ToothpicAPI.cs`. The `APISettings.settings` file is compiled when the project is built, so this setting must be configured before building and deploying the plugin.

### API Endpoints

In `ToothpicAPI.cs` there are two private variables `authEndpoint` and `userEndpoint`. These can be changed as necessary.

*Currently there is no other configuration required.*

# Namespace ToothpicExchange

Classes

### AuthForm

This class contains the form for users to authenticate (login).

### ExportPatientInfo

This class is responsible for exporting the patient in OpenDental to a user in Toothpic's API.

### ImportPatientInfo

This class is responsible for importing users from Toothpic's API into patients in OpenDental. It takes an API object in order to make authenticated requests.

### LaunchToothpicExchange

This class is the entry point for ToothpicExchange. When a user clicks the Toothpic button from the menu bar, this class is initialised from Plugin.cs Both import and export functions are launched from within this form.

### Plugin

This is a plugin class from OpenDental.

### ToothpicAPI

ToothpicAPI is the class responsible for handling Toothpic's REST API. All of its attributed are private. It holds the API's URL and the UserAuth object.

### ToothpicUser

This class contains the user model as defined by Toothpic's API. It contains methods to convert a Toothpic User to an OpenDental Patient, and vice versa.

Enums

### ExportResult

This Enum is to allow ExportPatientInfo report back with the degree of success it had. This is beacause ExportPatientInfo does not have a UI, it reports back to the homepage with these enums.

# Class AuthForm

This class contains the form for users to authenticate (login).

Inheritance

System.Object
System.MarshalByRefObject
System.ComponentModel.Component
System.Windows.Forms.Control
System.Windows.Forms.ScrollableControl
System.Windows.Forms.ContainerControl
System.Windows.Forms.Form
AuthForm

Implements

System.IDisposable

Namespace: ToothpicExchange

Assembly: ToothpicExchange.dll

Syntax

```
public class AuthForm : Form, IDropTarget, ISynchronizeInvoke, IWin32Window, IBindableComponent,
IArrangedElement, IComponent, IDisposable, IContainerControl
```

Remarks

The user enters the username/password which is used in an API call to authenticate an API object. If the authentication is successful, then that API object is made available for future authenticated API calls.

## Constructors

### AuthForm()

This method initialises the form.

Declaration

```
public AuthForm()
```

## Properties

### Call

The API object which has been authenticated and ready to use for API calls. It is public so it can pass the object on to the main plugin.

Declaration

```
public ToothpicAPI Call { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| ToothpicAPI | |

## Methods

### Dispose(Boolean)

Clean up any resources being used.

Declaration

```
protected override void Dispose(bool disposing)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| System.Boolean | disposing | true if managed resources should be disposed; otherwise, false. |

Overrides

System.Windows.Forms.Form.Dispose(System.Boolean)

## Implements

System.IDisposable

# Class ExportPatientInfo

This class is responsible for exporting the patient in OpenDental to a user in Toothpic's API.

Inheritance

System.Object
ExportPatientInfo

Name: ToothpicExchange
Assembly: ToothpicExchange.dll

Syntax

```
public class ExportPatientInfo
```

Remarks

It takes a Patient object, converts into a ToothpicUser object, and advises whether the user was POSTed successfully or not to Toothpic's API. An API object is required to make a POST request, otherwise it will fail.

## Constructors

### ExportPatientInfo(Patient)

This method initialises the class with the current selected patient. !important: This constructor is generally used for debug only, it is advised ExportPatientInfo(pat, call) is used instead.

Declaration

```
public ExportPatientInfo(Patient pat)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| OpenDentBusiness.Patient | pat | Usually the current patient within OpenDental |

### ExportPatientInfo(Patient, ToothpicAPI)

This method initialises the class with the current selected patient and an authenticated API call ready to go.

Declaration

```
public ExportPatientInfo(Patient pat, ToothpicAPI call)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| OpenDentBusiness.Patient | pat | Usually the current patient within OpenDental. |
| ToothpicAPI | call | An authenticated API object. |

## Methods

### PostToToothpic()

Implements ToothpicAPI to POST a ToothpicUser.

Declaration

```
public ExportResult PostToToothpic()
```

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| ExportResult | An enum ExportResult explaining whether the POST was successful or not. |

### WriteHL7ADT()

This method is used to construct a HL7 v2 ADT message to file. It is included for reference.

Declaration

```
public bool WriteHL7ADT()
```

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| System.Boolean | A boolean whether the file was successfully written or not |

# Enum ExportResult

This Enum is to allow ExportPatientInfo report back with the degree of success it had. This is beacause ExportPatientInfo does not have a UI, it reports back to the homepage with these enums.

Name: ToothpicExchange

Assembly: ToothpicExchange.dll

## Syntax

```
public enum ExportResult
```

## Fields

| NAME | DESCRIPTION |
| --- | --- |
| AuthenticationFailure | Authentication failure. |
| DuplicateUser | If a duplicate user already exists in Toothpic's API |
| Failure | Generic failure. |
| Null | Null object. |
| Success | Successful export. |

# Class ImportPatientInfo

This class is responsible for importing users from Toothpic's API into patients in OpenDental. It takes an API object in order to make authenticated requests.

Inheritance

System.Object
System.MarshalByRefObject
System.ComponentModel.Component
System.Windows.Forms.Control
System.Windows.Forms.ScrollableControl
System.Windows.Forms.ContainerControl
System.Windows.Forms.Form
ImportPatientInfo

Implements

System.IDisposable

Name: ToothpicExchange

Assembly: ToothpicExchange.dll

Syntax

```
public class ImportPatientInfo : Form, IDropTarget, ISynchronizeInvoke, IWin32Window, IBindableComponent,
IArrangedElement, IComponent, IDisposable, IContainerControl
```

Remarks

It querys the API to get a list of all ToothpicUsers and adds them to a ListView box. The user is selected and the import process begins. A new patient is created in OpenDental, then that patient is opened in a new FormPatientEdit window whereby the patient's information can be altered. If the FormPatientEdit window is cancelled then the user is not added.

## Constructors

### ImportPatientInfo(ToothpicAPI)

This method initialises the form.

Declaration

```
public ImportPatientInfo(ToothpicAPI call)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| ToothpicAPI | call | |

## Methods

### Dispose(Boolean)

Clean up any resources being used.

Declaration

```
protected override void Dispose(bool disposing)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| System.Boolean | disposing | true if managed resources should be disposed; otherwise, false. |

Overrides

System.Windows.Forms.Form.Dispose(System.Boolean)

## Implements

System.IDisposable

# Class LaunchToothpicExchange

This class is the entry point for ToothpicExchange. When a user clicks the Toothpic button from the menu bar, this class is initialised from Plugin.cs Both import and export functions are launched from within this form.

Inheritance

System.Object
System.MarshalByRefObject
System.ComponentModel.Component
System.Windows.Forms.Control
System.Windows.Forms.ScrollableControl
System.Windows.Forms.ContainerControl
System.Windows.Forms.Form
LaunchToothpicExchange

Implements

System.IDisposable

Name: ToothpicExchange

Assembly: ToothpicExchange.dll

Syntax

```
public class LaunchToothpicExchange : Form, IDropTarget, ISynchronizeInvoke, IWin32Window, IBindableComponent,
IArrangedElement, IComponent, IDisposable, IContainerControl
```

## Constructors

### LaunchToothpicExchange()

This method initialises the form.

Declaration

```
public LaunchToothpicExchange()
```

## Fields

### PatNum

The current patient Number taken from OpenDental's Patients.GetPat(PatNum)

Declaration

```
public long PatNum
```

Field Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.Int64 | |

## Methods

### Dispose(Boolean)

Clean up any resources being used.

Declaration

```
protected override void Dispose(bool disposing)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| System.Boolean | disposing | true if managed resources should be disposed; otherwise, false. |

Overrides

System.Windows.Forms.Form.Dispose(System.Boolean)

## Implements

System.IDisposable

# Class Plugin

This is a plugin class from OpenDental.

Inheritance

System.Object
OpenDentBusiness.PluginBase
Plugin

Inherited Members

OpenDentBusiness.PluginBase.HookMethod(System.Object, System.String, System.Object[])
OpenDentBusiness.PluginBase.HookAddCode(System.Object, System.String, System.Object[])
OpenDentBusiness.PluginBase.HookException(System.Exception)

Namespace: TypothpicExchange
Assembly: mTbolgythpicExchange.dll

Syntax

```
public class Plugin : PluginBase
```

## Properties

### Host

This is an OpenDental method.

Declaration

```
public override Form Host { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.Windows.Forms.Form | |

Overrides

OpenDentBusiness.PluginBase.Host

## Methods

### LaunchToolbarButton(Int64)

This is an OpenDental function to launch the main form when the toolbar button is clicked.

Declaration

```
public override void LaunchToolbarButton(long patNum)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| System.Int64 | patNum | This is the current patient number from OpenDental |

Overrides

OpenDentBusiness.PluginBase.LaunchToolbarButton(System.Int64)

# Class ToothpicAPI

ToothpicAPI is the class responsible for handling Toothpic's REST API. All of its attributed are private. It holds the API's URL and the UserAuth object.

Inheritance

System.Object
ToothpicAPI

Name: ToothpicExchange

Assembly: ToothpicExchange.dll

Syntax

```
public class ToothpicAPI
```

## Constructors

### ToothpicAPI()

Contains the BaseUrl, Authentication object and an API key.

Declaration

```
public ToothpicAPI()
```

## Methods

### AuthenticateUser(String, String)

Calls the API to authenticate the user. Returns a UserAuth object if successful, otherwise returns null.

Declaration

```
public ToothpicAPI AuthenticateUser(string username, string password)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| System.String | username | The user's username (or email) |
| System.String | password | The user's password |

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| ToothpicAPI | Returns itself if authentication was successful, otherwise null. |

### CheckDuplicateUser(ToothpicUser)

Fetches a list of users and checks for duplictes.

Declaration

```
public bool CheckDuplicateUser(ToothpicUser searchUser)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| ToothpicUser | searchUser | The user to be checked. |

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| System.Boolean | |

### Execute<T>(RestRequest)

This method executes a REST request. It is from the RestSharp class http://restsharp.org See RestSharp documentation for more info. It makes a request and adds the necessary headers for authentication etc.

Declaration

```
public T Execute<T>(RestRequest request)where T : new ()
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| RestSharp.RestRequest | request | Any RestSharp request object |

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| T | Returns the object specified, normally ToothpicUser. |

Type Parameters

| NAME | DESCRIPTION |
| --- | --- |
| T | Can be any type, normally ToothpicUser |

### GetAuthenticatedUser()

Fetches the current authorised user (by their user ID). Primarily used in LaunchToothpicExchange.cs.

Declaration

```
public ToothpicUser GetAuthenticatedUser()
```

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| ToothpicUser | The currently authenticated user as a ToothpicUser object, otherwise null. |

### GetToothpicUser(String)

This method fetches a user from the Toothpic API by a specified User ID.

Declaration

```
public ToothpicUser GetToothpicUser(string userId)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| System.String | userId | A Toothpic User ID. |

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| ToothpicUser | The specified ToothpicUser. |

### GetToothpicUsers()

This method fetches a list of Toothpic users from the API.

Declaration

```
public List<ToothpicUser> GetToothpicUsers()
```

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| System.Collections.Generic.List<ToothpicUser> | A List of Toothpic Users, null otherwise. |

### IsAuthenticated()

A boolean indicating whether the API object is authenticated or not.

Declaration

```
public bool IsAuthenticated()
```

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| System.Boolean | |

### PostToothpicUser(ToothpicUser)

This method POSTs a Toothpic User to the API. The JSON formatting is defined within the method.

Declaration

```
public bool PostToothpicUser(ToothpicUser user)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| ToothpicUser | user | Any ToothpicUser. |

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| System.Boolean | A boolean whether it POSTed successfully or not |

# Class ToothpicUser

This class contains the user model as defined by Toothpic's API. It contains methods to convert a Toothpic User to an OpenDental Patient, and vice versa.

Inheritance

System.Object
ToothpicUser

Namespace: ToothpicExchange
Assembly: ToothpicExchange.dll

Syntax

```
public class ToothpicUser
```

## Constructors

### ToothpicUser()

Contains all of the user attributes. ToothpicUser(Patient pat) is recommended.

Declaration

```
public ToothpicUser()
```

### ToothpicUser(Patient)

Maps an OpenDental patient to a Toothpic user.

Declaration

```
public ToothpicUser(Patient Pat)
```

Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| OpenDentBusiness.Patient | Pat | An OpenDental patient |

## Properties

### DateOfBirth

DateOfBirth mapped to Toothpic API as "dob".

Declaration

```
[DeserializeAs(Name = "dob")]
public DateTime DateOfBirth { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.DateTime | |

### Email

Email address from Toothpic API.

Declaration

```
public string Email { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.String | |

### FirstName

FirstName mapped to Toothpic API as "first_name".

Declaration

```
public string FirstName { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.String | |

### Gender

Gender, recorded as a string from the Toothpic API ("male", "female", "other") which maps to OpenDental's enum PatientGender (male, female, unknown).

Declaration

```
public string Gender { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.String | |

### LastName

LastName mapped to Toothpic API as "last_name".

Declaration

```
public string LastName { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.String | |

### MemberNumber

Member Number from Toothpic API, not currently used in OpenDental.

Declaration

```
public string MemberNumber { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.String | |

### MiddleName

MiddleName mapped to Toothpic API as "middle_name".

Declaration

```
public string MiddleName { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.String | |

### ZipCode

ZipCode mapped to Toothpic API as "zip_code".

Declaration

```
public string ZipCode { get; set; }
```

Property Value

| TYPE | DESCRIPTION |
| --- | --- |
| System.String | |

## Methods

### CreateNewPatient()

Modified from OpenDental's Patients.CreateNewPatient !important: assumes the user information has already been validated. It creates a new patient from the current instance of ToothpicUser and adds them to the database.

Declaration

```
public Patient CreateNewPatient()
```

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| OpenDentBusiness.Patient | A new patient object |

Remarks

This is a complex method which currently conforms to OpenDental 17.2 but should be carefully monitored. This method is used in ImportPatientInfo.AddNewPatient It does the majority of the mapping from a ToothpicUser to an OpenDental patient.

### ToString()

Simply formats ToothpicUser as a string.

Declaration

```
public override string ToString()
```

Returns

| TYPE | DESCRIPTION |
| --- | --- |
| System.String | A string with each item separated by a line break. |

Overrides

System.Object.ToString()