**CITI 2.0 Development Guide**

1. Development environment

Visual Studio 2017

.Net core 2.1

SQL Server 2016

Microsoft SQL Server Management Studio 17

1. Required NuGet packages and plugins

2.1 NuGet:

**Bootstrap:**

Install-Package bootstrap

Install-Package bootstrap.sass

**SixLabors.ImageSharp:**

Install-Package SixLabors.ImageSharp -Version 1.0.0-beta0007

2.2 Plugins:

**JCrop :** <http://code.ciaoca.com/jquery/jcrop>

**Bootstrap-fileinput:** <http://plugins.krajee.com/file-input>

**bootstrap-suggest-plugin:**  <https://github.com/lodev09/bootstrap-suggest>

1. How to use on first time run

3.1 Create two database : CMSCore and CMSCoreIdentity.

3.2 Create table for database(script in the “database”

folder).

3.3 Run the project, Enter this URL to create a user:

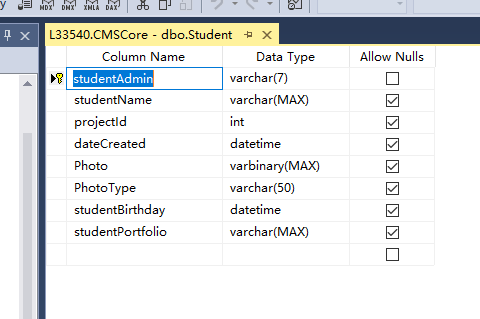
<http://localhost:60905/Login/Create>

3.4 The default username is “admin” and the password is “Sun123!”.

1. How to create a new set of features

First, you need to create a table in the database and create a module corresponding to the table. Next we have "Student" as an example.

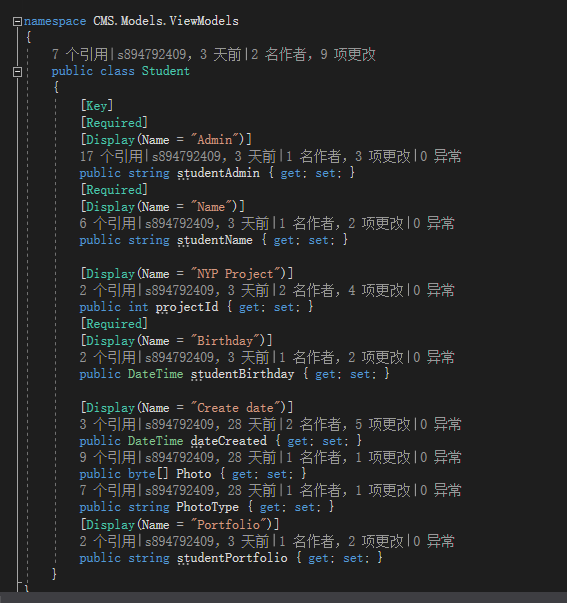
* 1. Create database table



* 1. Create Model

“[Display(Name = "Admin")]”is used to set the name displayed on the view

“[Required]”is used to set it as a required item, which is convenient for verification when submitting a form.

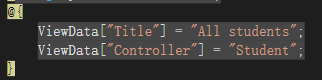


* 1. Create controllers and views

When creating controllers and views you have to make sure their names are the same. For example, the view corresponding to the StudentController is student, and the view corresponding to the StudentsController is students.

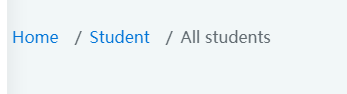
* 1. View file considerations

In the view you can see that there are some ViewBags that are used to display the title and directory.



ViewData["Title"] is use to set title and displayed directory name

ViewData["Controller"]is use to set displayed controller name

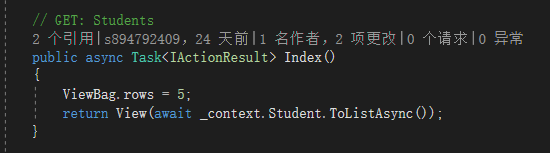




These basic view files are in Views-Shared.

* 1. Show all data

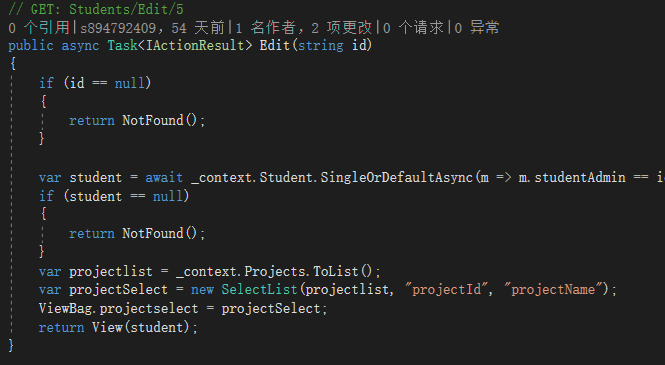
You need to query all the data in the Controller and return to the view, which is displayed by the view. The .net core is very simple to operate on the database, the code is as follows.



View code you can see the file in the project.

* 1. Edit data

The principle of Edit data is to transfer the id of the data to the controller when the user clicks the button, then find the data in the controller and return to the edit view, then display the data in the edit view, and change it in the user. Update the data when you click Submit.

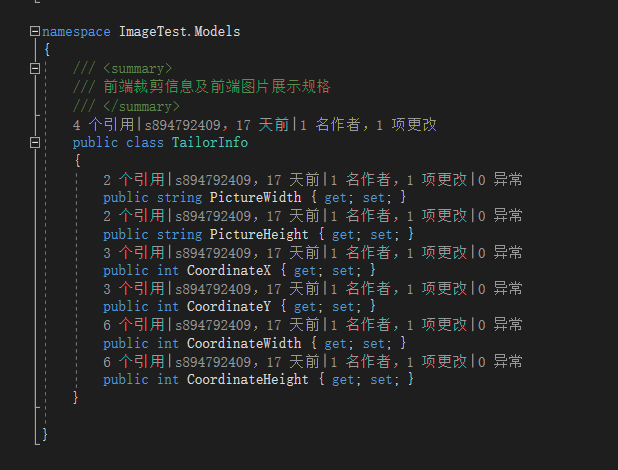


Because we need to crop the image and display the thumbnails, we need to use the Ajax method to upload. The principle of cropping the image is to use the Jcrop plugin to select the area to be cropped and then submit the data with bootstrap-fileinput, send the location data and images to the controller at commit time, and use the third-party class library SixLabors.ImageSharp to crop in the controller. And then return a thumbnail of the base64 type. Because this is not necessarily the user's final choice, we also need to submit images and crop data when the user submits the data, so we need to use Ajax upload.

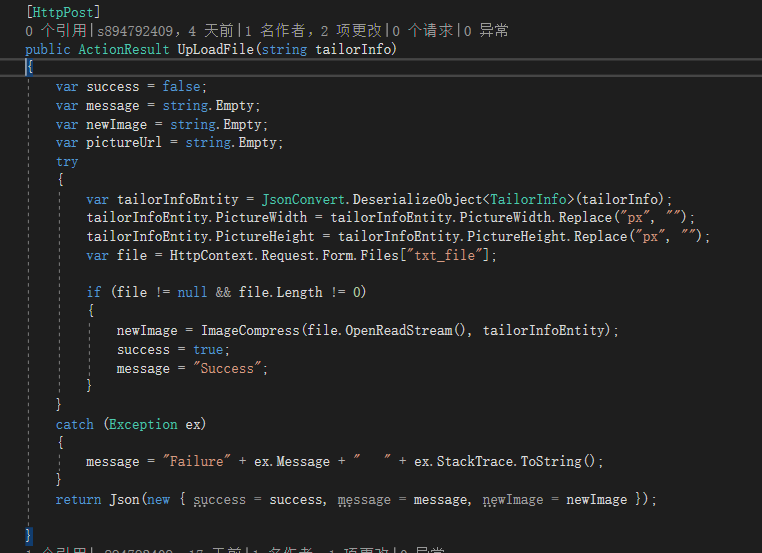
The script for crop images and the uploaded scripts are more code-oriented. Please look directly at the code in the project. Below we focus on the use of SixLabors.ImageSharp to crop the image and save the image in the Controller. If you want to learn more you can read my blog: <https://blog.csdn.net/s894792409/article/details/102952736> .

* 1. How to crop image

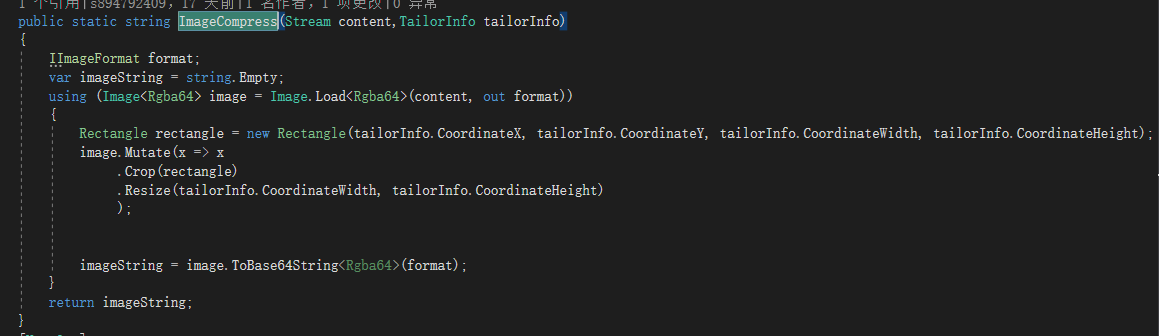
First, convert the uploaded crop data into modules. The code for this module is as follows, it contains data for all cropped images.



Convert the Json format crop data into module:



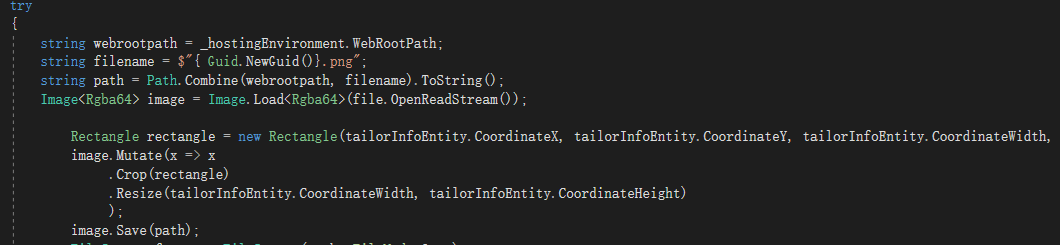
Crop image:



* 1. How to save image to the database

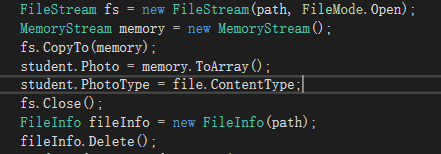
The process of saving a picture is similar to the process of processing a picture. There are two methods for storing pictures. One is to store the byte data of the picture, and the other is to store the picture path. Here, the byte data of the picture is stored.

First crop the image and store it:



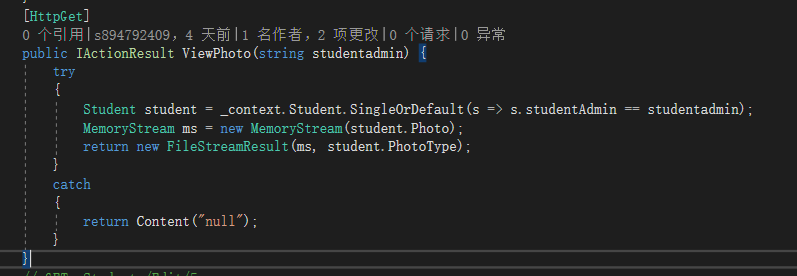
Second, read the image you just saved and convert it to bytes.

Finally save the byte data and image type to the module, don't forget to delete the photo after completion to avoid redundancy.



* 1. How to display images from the database

Since the images we store in the database are byte-type, we need to process the data as it is displayed. At this point, the image types we stored previously come in handy. We create a new function in the controller that is used to process the image.



Here we are the student admin sent through the front-end interface to find the corresponding data, and then convert the data into a MemoryStream, and finally combine the MemoryStream object with the image type and return it to the front-end interface.

Now we still need the final step, which is to call this function from the front end. We need to put an <img> tag in the front-end interface, set its src to the address of the function, and use the get method to pass the student admin.



* 1. Create data and delete data

Create data and delete data is very similar to previous edit data. You can develop other create data and delete data after understand the edit data.

1. API

All apis of this project are in the Controller->API directory. It uses GET, PUT, DELETE, POST four methods, GET is used to get data, PUT is used to update data, DELETE is used to delete data, POST is used to create data. . The creation of the API is relatively simple compared to the Controller, so there is no more explanation here. You can refer to the API file and the Controller to create the API.