Query the Wage Deducted Tax API

1. Requirement
   1. Use different ways for different areas (only need to implement the dummy ways which are based on different areas, but not the actual one)
   2. Input the gross wage, output the deducted tax wage.
   3. Doesn’t need fully follow RESTful
   4. A simple user use-case: default to use the latest area if no area be submitted.
2. Web API
   1. **API Path:** /wage/DeductedTax
   2. **HTTP Method:** GET
   3. **Input:** 
      1. GrossWage: double
      2. AreaKey : string
   4. **Output**
      1. DeductedWage
      2. AreaKey
      3. AreaName
      4. Time

|  |  |  |
| --- | --- | --- |
| Item | Type | Desc |
| Code | Int | 0:Success; other values mean failure |
| Message | String | Description of API return result |
| Data | Object |  |
| DeductedWage | Double | Returned Deduted Tax Wage |
| AreaKey | String | The AreaKey of calculation |
| AreaName | String | The name of area class instance |
| Time | DateTime | Calculation finish time |

Example:

{

"code":0

,"message":"Calculation Success!"

,"data":{

"DeductedWage":850.0

,"AreaKey":"AreaA"

,”AreaName”:”Area A”

,"Time":"2018-11-07T11:34:27.5896896+08:00"

}

}

* 1. **Exception**
     1. GrossWage Error

Code=101

Message = “Please input GrossWage value for calculation.”

* + 1. AreaKey Not Exist

Code=102

* + 1. Calculation Error

Code=103

* + 1. OtherError

Code=201

* 1. **Interfaces and Classes**

IArea

ITaxCalculation

AreaFactory

Create(areaKey):IArea

TaxCalculationFactory

Create(calKey): ITaxCalculation

AreaBase

AreaA

AreaB

AreaC

CalculationA

CalculationB

CalculationC

* + 1. IArea / AreaFactory
    2. ITaxCalculation / TaxCalculationFactory
    3. Definition / AreaDefinition / ClassFactory
  1. **Lifecycle**

Yes

Yes

No

No

No

Yes

Yes

Get *AreaKey* Value

Is Empty?

TaxCalculationFactory

AreaFactory

*GrossWage* Error

*AreaKey* Not Exist

Calculation Error

*DeductedWage*

*Exception*

Calculate?

Get latest?

Create Instance?

Yes

Get *GrossWage* Value

Is Valid?

Yes

No

No

* 1. **Namespace**
     1. Abstract Business Logic

*Demo.Wage*

* + 1. Business Reallization

*Demo.Wage.AreaA*

*Demo.Wage.AreaB*

*Demo.Wage.AreaC*

* + 1. Web API

*Demo.Wage.API*

* + 1. Testing

*UnitTestProject1*

1. Result
   1. **Source Code**
   2. **TestCases**

In this demo project, the business logic is very simple. So I ignored Unit Tests and jump into Integration Test directly.

* + 1. Input for Area A and Area B, get different Deducted Tax Wage
    2. No AreaKey input (First query)
    3. No AreaKey input (Using latest area)

***Because it is difficult to implement session read/write correctly in test project, so we prefer to do it by testing on browser.***

Steps:

Step 1> Run Demo.Wage project;

Step 2> Input

url : <https://localhost:44399/wage/DeductedTax/1000/AreaA>

Expected:

1. API return Success
2. Returned DeductedWage should equal 850.0

Step 3> Input (without AreaKey)

url : <https://localhost:44399/wage/DeductedTax/2000/>

Expected:

1. API return Success
2. Returned DeductedWage should equal 1700.0
3. Returned AreaKey should equal AreaA
   * 1. AreaKey not exist
     2. GrossWage error
     3. Calculation error
     4. Loading AreaA class instance with CalculationB logic