It is assumed that the TaxRate will be needed for display purposes. Otherwise the GetTaxRate method should be private.

TaxServiceController Actions

- GetTaxRate(string zip)
    - Returns the tax rate when valid US zip is passed otherwise returns Invalid Zip Code
    - Example https:/[domain]api/taxservice/GetTaxRate?zip=07747
- GetTotalTax(string zip, double orderTotal)
    - Calls GetTaxRate to get the Rate then multiplies the rate by orderTotal.
    - Example https://[domain]/api/taxservice/GetTotalTax?zip=07747&orderTotal=47.4

To Add a new Tax Service: (in the example provided to NewTaxCalcularor.cs has been added to confirm this solution is flexible)

- Create a class for the new service that implements the ITaxcalculator interface.
- Add constant for the class to the static class CalculatorList which is used to store the list of calculators so they can be referenced in code
- Add service to the Startup.cs Transient so that the service can be used in TaxServiceController when the business rules/logic dictate this calculator should be used
- Add call to the new calculator in the TaxServiceController business rules/logic dictate this calculator should be used. When referencing this controlled use the TaxCalcMapper delegate and pass the constant entered above