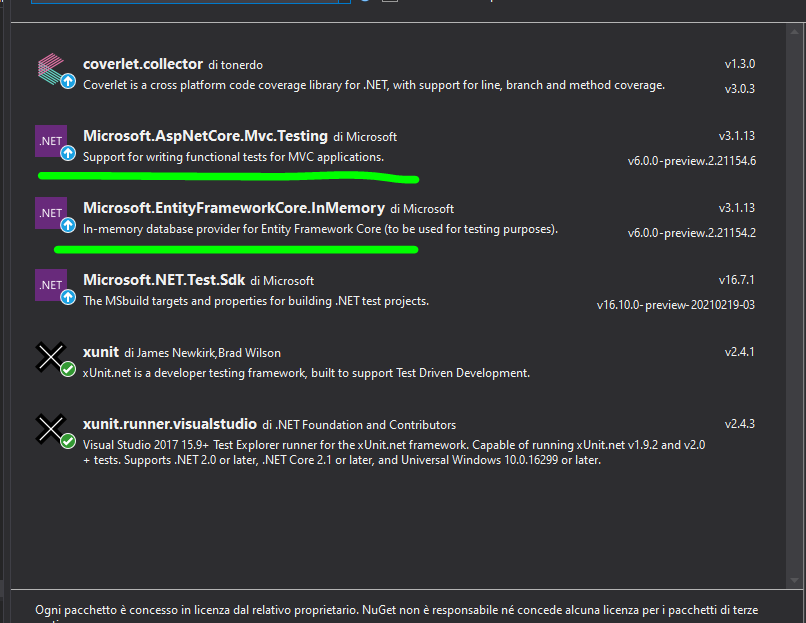
**Integration Tests**

<https://code-maze.com/integration-testing-asp-net-core-mvc/>

Aggiungere un progetto Xunit

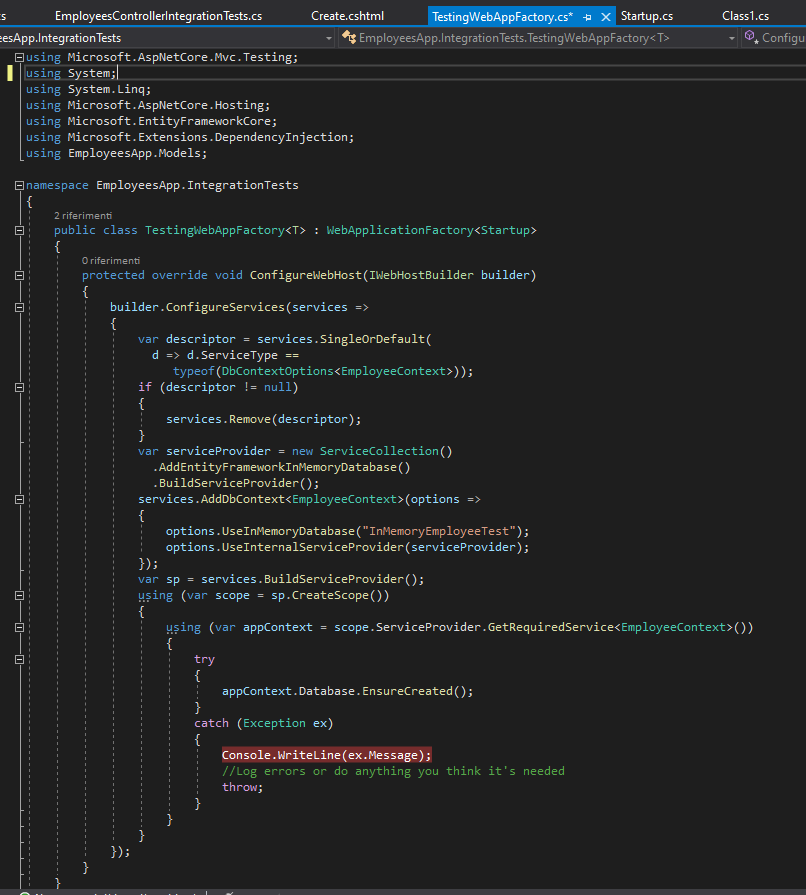
Rinominare la classe di default in EmployeesControllerIntegrationTests

Aggiungiamo i seguenti pacchetti che ci permettono di creare un client e di tenere in memoria un db provider:



Importare la dll del progetto da testare

Creare la classe testApplicationFactory

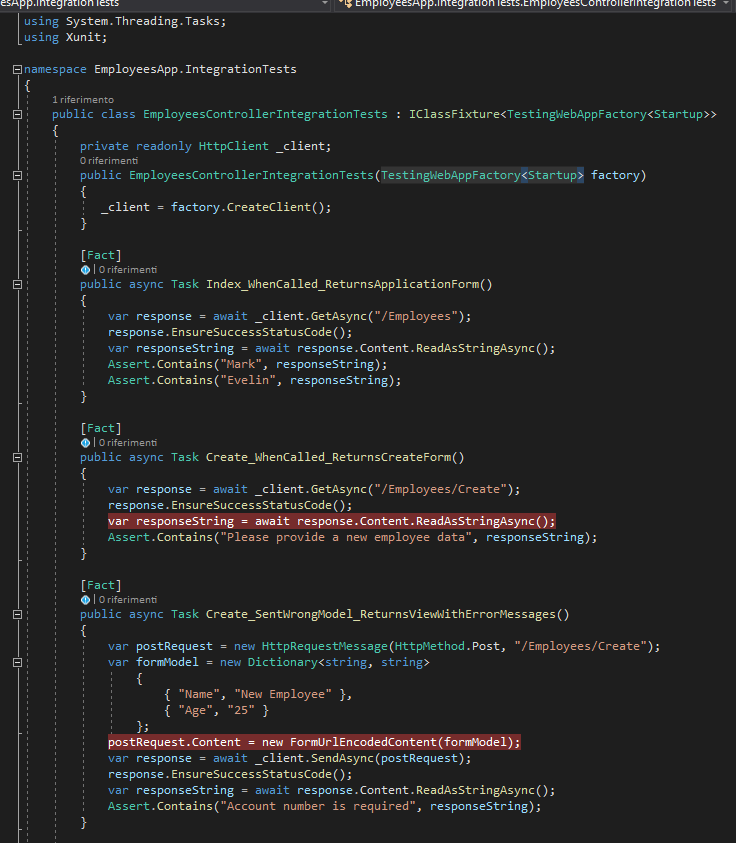


Questa classe va a fare un override del ConfigureWebHost:

rimuove l’EmployeeContext ed aggiunge un inmemory context (ossia un contest in memoria fittizio che andrà a scrivere e leggere i dati non sul db reale)

Quindi andiamo a creare una classe per effettuare i test, questa classe deve ereditare da una interfaccia generica IClassFixture che in questo caso userà un oggetto WebApplicationFactory (TestingWebAppFactory eredita da questo):

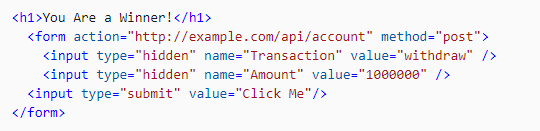
Quindi il nostro oggetto TestingWebAppFactory<Startup> verrà iniettato nel costruttore con la DI e mediante createClient verrà creato il client http.



Anti Forgery Token.

E’ un sistema per impedire l’invio di form fraudolenti da altri domini verso l’applicazione vulnerabile.

Supponiamo uno si logghi nella sua intranet bancaria, ottiene un cookies di autenticazione, poi a sua insaputa visita il sito fraudolento dove vi è un form così:



Se preme submit invierà una richiesta in post al sito vulnerabile ordinando di effettuare l’azione di trasferimento denaro.

Un form coperto da token di antifalsificazione invia un codice generato ogni volta in un cookies nell’header ed uno in un campo nascosto del form.

Al submit del form questi due cookies devono essere reinviati in post affinche la richiesta sia validata.

L’anti falsificazione funziona poiché il sito fraudolento non può accedere ai codici: la pagina dannosa può inviare richieste ma non leggere risposte.