

Qu'est-ce qu'un bot ?

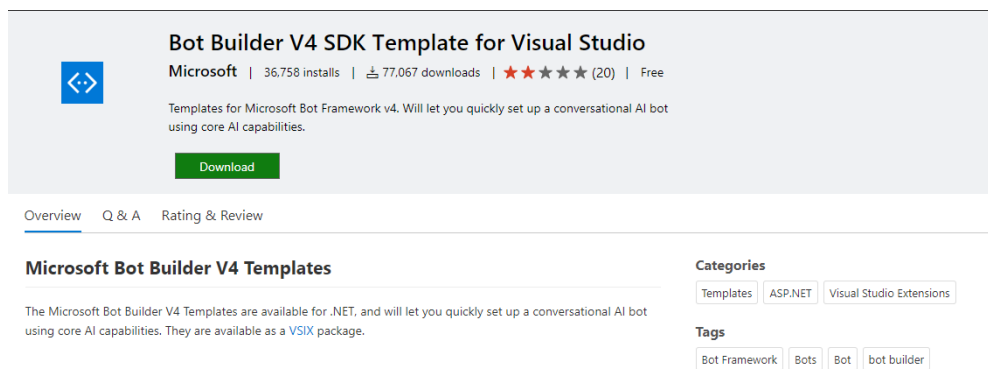
Les bots offrent une expérience qui donne l'impression aux utilisateurs d'avoir moins à faire à un ordinateur, et plus à une personne, ou du moins à un robot intelligent. Les bots peuvent être utilisés pour faire passer des tâches répétitives et simples, comme la réservation de table ou le recueil d'informations de profil, vers des systèmes automatisés qui ne nécessitent plus d'intervention humaine directe. Les utilisateurs conversent avec le bot à l'aide de texte, de cartes interactives et avec la voix. Une interaction avec un bot peut se composer d'une question-réponse rapide, ou il peut s'agir d'une conversation plus sophistiquée fournissant un accès à des services de manière plus intelligente.

Les bots sont comparables à des applications web modernes. Ils résident sur Internet et utilisent des API pour envoyer et recevoir des messages. Le contenu d'un bot varie considérablement selon son type. Les logiciels de bots modernes s'appuient sur une pile de technologies et d'outils qui offrent des expériences de plus en plus complexes sur un large éventail de plateformes. Toutefois, un bot simple peut se contenter de recevoir un message et de répondre à l'utilisateur avec très peu de code.

Les bots peuvent effectuer les mêmes opérations que d'autres types de logiciels : lire et écrire des fichiers, utiliser des bases de données et des API, et réaliser des tâches de calcul classiques. Ce qui rend les bots uniques, c'est leur utilisation de mécanismes généralement réservés à la communication entre humains.

Créer votre bot avec bot Builder

Azure Bot Service propose un ensemble intégré d'outils et de services permettant de faciliter ce processus. Choisissez votre environnement de développement ou vos outils de ligne de commande favoris pour créer votre bot. Des kits de développement logiciel (SDK) existent pour C#, JavaScript et Typescript. (Les kits de développement logiciel pour Java et Python sont en cours de développement.) Nous proposons des outils pour différents stades de développement de bot, afin de vous aider à concevoir et créer des bots.



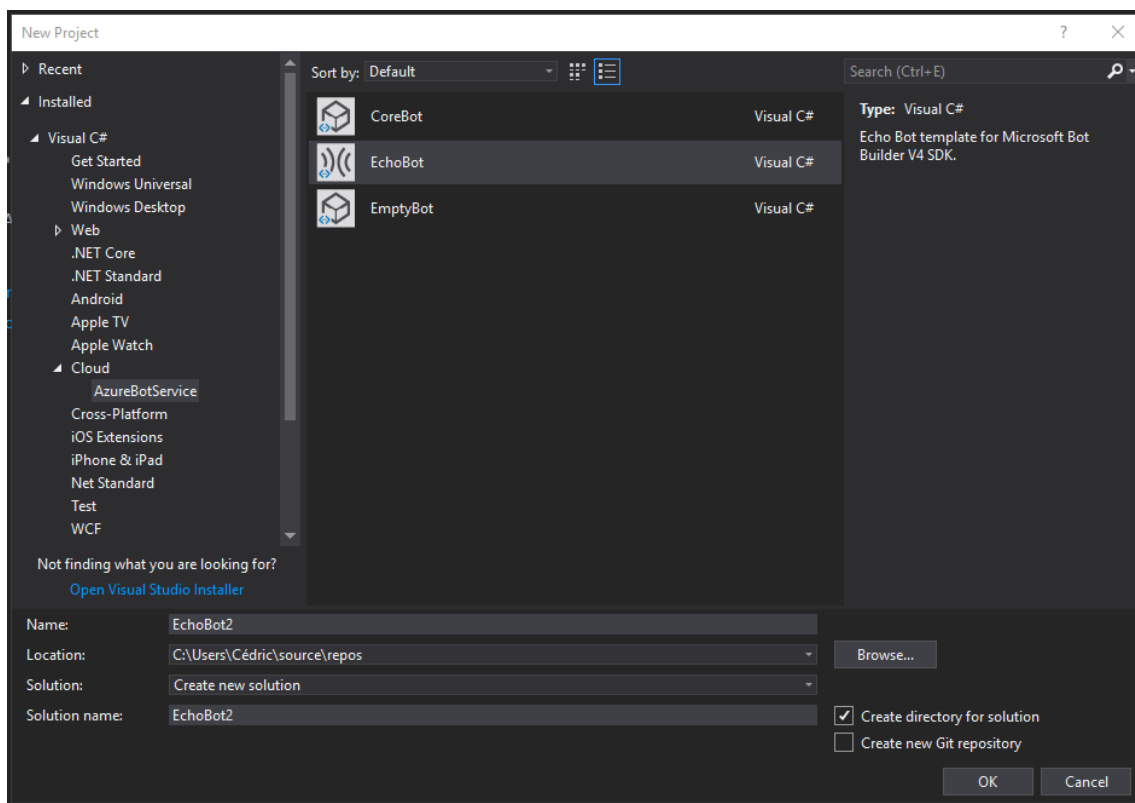
The screenshot shows the Visual Studio Marketplace page for the "Bot Builder V4 SDK Template for Visual Studio" by Microsoft. It includes the Microsoft logo, a download button, and a description: "Templates for Microsoft Bot Framework v4. Will let you quickly set up a conversational AI bot using core AI capabilities." Below the main card, there are tabs for "Overview", "Q & A", and "Rating & Review". To the right, there are "Categories" (Templates, ASP.NET, Visual Studio Extensions) and "Tags" (Bot Framework, Bots, Bot, bot builder).

1. Installer Bot Builder V4 SDK Template for Visual Studio

<https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4>

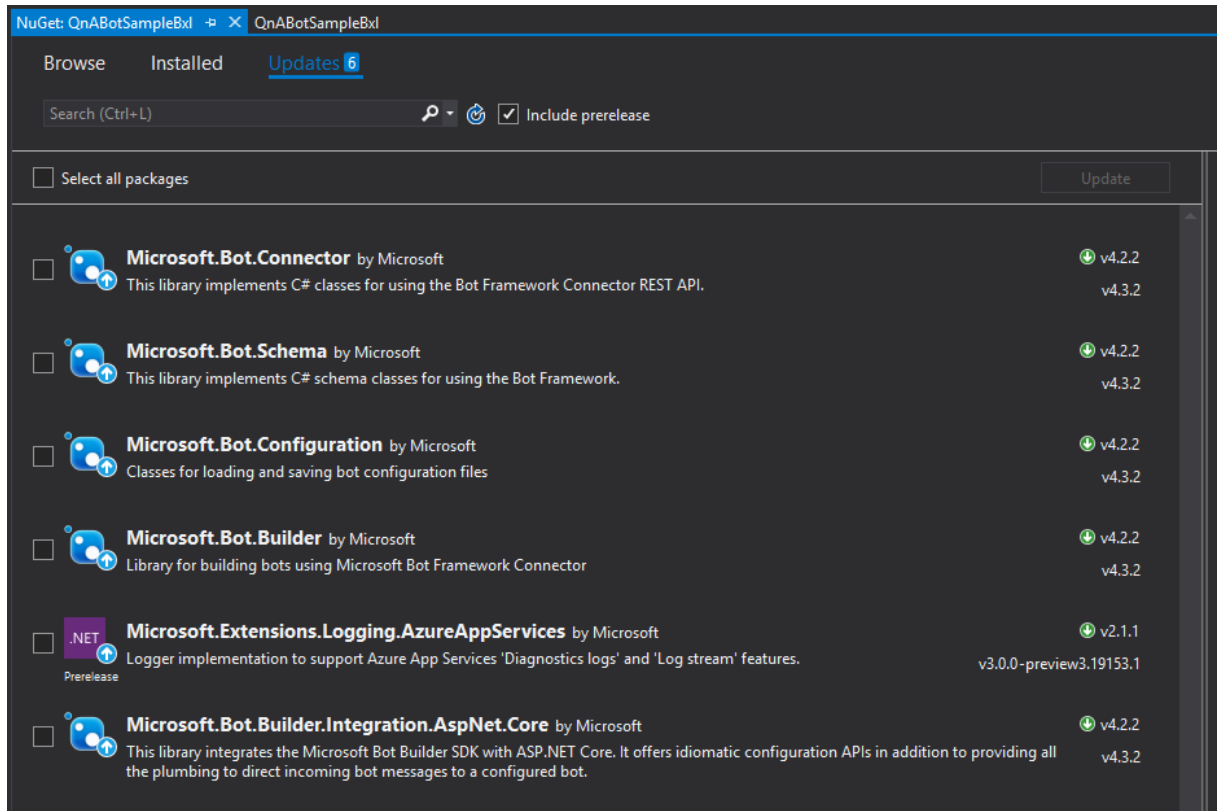


2. Créez un nouveau projet de type EchoBot



3. Mettez les package nuget à jour et compilez





4. Exécutez le projet et test du bot.

Maintenant, nous allons tester notre bot pour vérifier s'il fonctionne.

Pour se faire nous allons avoir besoin du bot emulator.

<https://github.com/Microsoft/BotFramework-Emulator/releases/tag/v4.3.3>

5. Nous allons ajouter une base de connaissance.

Pour se faire nous allons créer un service que nous viendrons injecter par injection de dépendance.

- Créez un répertoire Services
- Ajouter une interface IQnaMakerServices qui contiendra l'instance du QnAMaker

```
public interface IQnaMakerServices
{
    QnAMaker QnAMaker { get; }
}
```

- Il faut pour ajouter le package nuget du QnAMaker.

```
Microsoft.Bot.Builder.AI.QnA
```

- Ajoutez une nouvelle class QnaMakerServices qui contiendra la définition du service. Cet object va initialiser le QnAMaker que notre bot utilisera.



```

public class QnaMakerServices : IQnaMakerServices
{
    private const string Name = "QnABot";

    public QnaMakerServices(BotConfiguration config)
    {
        QnaMaker = Init(config);
    }

    public QnaMaker QnaMaker { get; }

    private QnaMaker Init(BotConfiguration config)
    {
        var service = config.Services.FirstOrDefault(s => s.Type ==
ServiceTypes.QnA && s.Name == Name);

        var qna = (QnaMakerService)service;
        if (qna == null)
        {
            throw new InvalidOperationException("The QnA service is not
configured correctly in your '.bot' file.");
        }

        if (string.IsNullOrEmpty(qna.KbId))
        {
            throw new InvalidOperationException("The QnA KnowledgeBaseId
('kbId') is required to run this sample. Please update your '.bot' file.");
        }

        if (string.IsNullOrEmpty(qna.EndpointKey))
        {
            throw new InvalidOperationException("The QnA EndpointKey
('endpointKey') is required to run this sample. Please update your '.bot' file.");
        }

        if (string.IsNullOrEmpty(qna.Hostname))
        {
            throw new InvalidOperationException("The QnA Host ('hostname') is
required to run this sample. Please update your '.bot' file.");
        }

        var qnaEndpoint = new QnaMakerEndpoint()
        {
            KnowledgeBaseId = qna.KbId,
            EndpointKey = qna.EndpointKey,
            Host = qna.Hostname,
        };

        var qnaMaker = new QnaMaker(qnaEndpoint);

        return qnaMaker;
    }
}

```

e. Ajoutez l'instance `QnaMakerServices` dans le conteur IOC.



Pour cela dans le Startup.cs dans la méthode configureServices ajoutez en singleton le service.

```
services.AddSingleton<IQnaMakerServices, QnaMakerServices>();
```

- f. Votre service est maintenant accessible
Par injection via le constructeur de votre bot, vous allez pouvoir récupérer le qnaMakerServices. Et le conserver comme variable private.
- g. Maintenant dans la méthode OnTurnAsync, vous allez pouvoir l'utiliser.
Si le type de l'activity est bien message :

```
var response = await
_qnaMakerServices.QnAMaker.GetAnswersAsync(turnContext);
    if (response != null && response.Length > 0)
    {
        await
turnContext.SendActivityAsync(response[0].Answer);
    }
    else
    {
        var msg = @"No QnA Maker answers were found. This
example uses a QnA Maker Knowledge Base that focuses on smart light
bulbs.

                To see QnA Maker in action, ask the bot
questions like 'Why won't it turn on?' or 'I need help.';

        await turnContext.SendActivityAsync(msg);
    }
```

- h. Ajouter dans le fichier config .bot le point d'entrée du service qna

```
{
    "type": "qna",
    "endpointKey": "cfbbaee1-0973-48cc-b366-a1313758c3db",
    "hostname": "https://labbx1.azurewebsites.net/qnamaker",
    "id": "117",
    "kbId": "cdcd7b37-4d3f-4a10-ba31-2c46ac7e8ea3",
    "name": "QnABot",
    "subscriptionKey": ""
}
```

6. Testez votre bot.

(Optional)

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-tutorial-dispatch?view=azure-bot-service-4.0&tabs=csharp>



Api

<https://westus.dev.cognitive.microsoft.com/docs/services/5a93fcf85b4ccd136866eb37/operations/5ac266295b4ccd1554da75ff>

