

JetBrains - Apprendre à créer des prompts d'IA personnalisés dans les IDE ReSharper et Rider

Par Rachel Appel - Delphine Massenhove (traducteur)

Date de publication : 16 mars 2024

Pour réagir au contenu de cet article, un espace de dialogue vous est proposé sur le forum.
Commentez

Introduction.....	3
I - Prompts : comment favoriser la pertinence des réponses de l'IA.....	3
II - Créer un prompt basique.....	3
III - Sauvegarder des prompts personnalisés.....	8
IV - Conclusion.....	8

Introduction

Le recours à l'IA s'est rapidement répandu dans le secteur du développement de logiciels, d'où une demande croissante d'informations sur les bonnes pratiques en vue de tirer le meilleur parti de cette technologie pour la réalisation de tâches quotidiennes. Dans cet article, nous allons voir comment **écrire des prompts personnalisés à utiliser avec l'AI Assistant de JetBrains dans les IDE ReSharper et Rider**, qui vous permettront de tirer le meilleur parti de l'IA.

I - Prompts : comment favoriser la pertinence des réponses de l'IA

Les prompts sont des requêtes ou instructions écrites envoyées à une IA en vue d'obtenir une réponse ou un résultat. L'IA utilise le traitement du langage naturel (NLP) dans les prompts afin qu'ils génèrent des interactions semblables à des conversations entre deux personnes. Les outils d'IA acceptent vos requêtes et les transmettent à un LLM (Large Language Model) qui vous répond.

En tant que développeurs, nous savons que les **ordinateurs prennent nos instructions au pied de la lettre**. Lorsque notre code ne correspond pas aux besoins de l'utilisateur, cela crée un bug. L'ordinateur n'est pas en cause, il fait ce que l'on lui demande, c'est le programmeur qui est fautif. De la même façon, **lorsque vous créez un prompt, vous devez être précis**. Pour que l'IA puisse effectuer correctement la tâche souhaitée, il est nécessaire que **les requêtes et commandes soient formulées de manière claire et pertinente dans les prompts**.

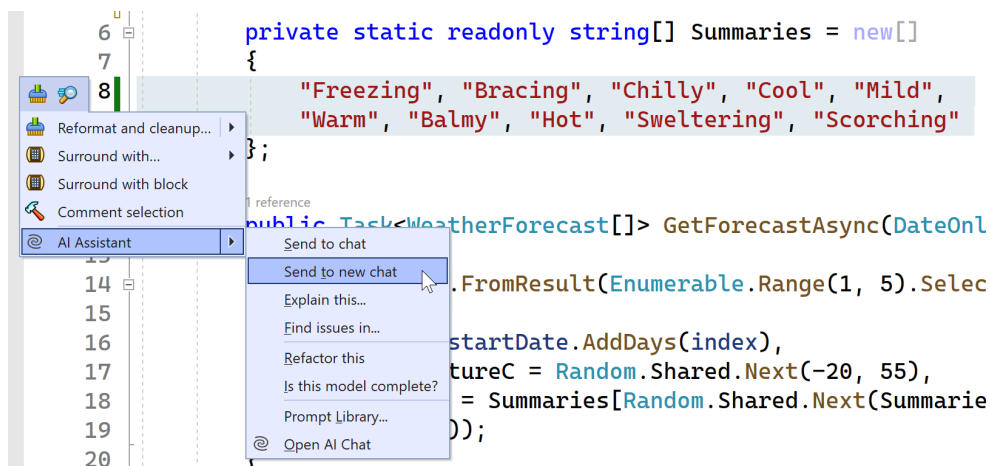
Les outils d'IA s'appuyant fortement sur le contexte pour accomplir leurs tâches, il est essentiel d'y prêter une attention particulière. Le contexte consiste tout d'abord dans le projet en cours : l'ensemble des fichiers et des ressources que votre projet contient. À cela s'ajoute l'historique du chat et les connaissances générales de l'IA concernant le problème à traiter. Vous pouvez également inclure d'autres informations contextuelles dans les prompts. Par exemple, vous pouvez répéter une commande précédente ou utiliser des informations provenant d'une partie précédente de la conversation. **La technique « d'interactivité des prompts », basée sur un processus d'affinement et d'amélioration itératif des prompts, permet d'établir un échange en mode conversationnel avec l'IA.**

Imaginons que souhaitez apprendre à jouer au tennis en vue de participer à des compétitions alors que vous n'avez encore jamais pris un seul cours. Vous pourriez demander « Comment participer à un tournoi national de tennis ? » puisque c'est votre objectif final, mais dans ce cas, l'IA fera probablement la même déduction qu'un humain : ce type de question laisse entendre que vous savez jouer au tennis et avez déjà participé à des tournois locaux. Les données renvoyées vous expliqueront donc comment vous inscrire à un tournoi national, mais pas comment acquérir les compétences pour jouer à ce niveau. Vous obtenez cette réponse car vous avez omis de mentionner des éléments de contexte importants. Reformuler votre requête en disant « Je veux participer à des compétitions de tennis. Je n'ai jamais joué au tennis auparavant. Que dois-je faire pour atteindre un niveau me permettant de participer à un tournoi national ? » devrait produire de meilleurs résultats, car vous fournissez des informations contextuelles plus pertinentes. **L'IA étant entraînée à partir de données basées sur des interactions humaines**, elle applique la même logique que les humains à de nombreux égards.

L'une des meilleures façons de développer le contexte est de mettre l'IA au défi. Demandez lui « Pourquoi penses-tu cela ? » ou « Sur quoi bases-tu ta réponse ? » pour voir quels sont les résultats. Quoi qu'il en soit, il est recommandé de toujours traiter les réponses de l'IA avec précaution. Certains LLM ont tendance à fournir des réponses longues, mais leur demander de rester concis peut aider à obtenir des retours plus clairs. D'autre part, vous pouvez aussi corriger et clarifier vos prompts en fonction des réponses de l'IA. Si elle vous a mal compris, vous pouvez simplement lui indiquer pourquoi et poursuivre.

II - Créer un prompt basique

Plusieurs approches sont possibles pour créer un prompt. Ouvrez l'AI Assistant de ReSharper pour commencer la conversation. Vous pouvez aussi appeler l'AI Assistant à partir d'un bloc de code mis en évidence, comme ci-dessous :



Send to new chat crée un contexte pour le chat et envoie le code mis en évidence au chat de l'**AI Assistant**. Ajoutez vos informations au prompt et envoyez-le. Lorsque vous avez obtenu une réponse, vous pouvez utiliser le prompting interactif afin de finaliser l'exécution de votre tâche.

Dans l'exemple suivant, l'**AI Assistant** a été sollicité pour créer à la fois un JSON valide et des classes C# à partir de la chaîne de valeurs. Comme vous pouvez le voir, la réponse est assez basique. Le JSON consiste en un simple tableau et la classe contient un objet List pour contenir ces valeurs.

Creating JSON Data and Corresponding C# Class for Temperature Descriptions

Your query

Use the following values to create valid JSON for temperature data as well as C# classes:

```
C#
"Freezing", "Bracing", "Chilly", "Cool", "Mild",
"Warm", "Balmy", "Hot", "Sweltering", "Scorching"
```

AI Assistant

Sure, let's start with creating a JSON representation for the temperature data.

```
Json
{
  "temperatures": [
    "Freezing",
    "Bracing",
    "Chilly",
    "Cool",
    "Mild",
    "Warm",
    "Balmy",
    "Hot",
    "Sweltering",
    "Scorching"
  ]
}
```

This JSON object has a single property `temperatures` which is an array of strings.

Now, let's create a corresponding C# class for this JSON structure. We'll need a class `TemperatureData` with a property `Temperatures` which is a list of strings.

```
C#
public class TemperatureData
{
  public List<string> Temperatures { get; set; }
}
```

You can then use this class with a JSON deserialization library (like `Newtonsoft.Json` or `System.Text.Json`) to parse the JSON into a `TemperatureData` object.

En général, nous souhaitons que nos modèles ressemblent le plus possible aux structures de données avec lesquelles nous travaillons. Avec le prompting interactif, nous pouvons ajouter des instructions pour rendre notre classe plus compatible avec Entity Framework, comme dans l'exemple ci-dessous :

Your query

Change the class so that it works with EF Core data annotations for common validation of individual temperatures

AI Assistant

Sure, to make the `TemperatureData` class work with EF Core and data annotations, you need to add the necessary using directives and modify the class as follows:

```
C#
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
[Table(name: "TemperatureData")]
public class TemperatureData
{
    [Key]
    public int Id { get; set; }
    [Required]
    [StringLength(50)]
    public List<string> Temperatures { get; set; }
}
```

In this updated class:

- The `Table` attribute is used to specify the name of the database table that this class maps to.
- The `Key` attribute denotes that the `Id` property is the primary key of the table.
- The `Required` attribute indicates that the `Temperatures` property must always have a value.
- The `StringLength` attribute limits the length of each string in the `Temperatures` list to a maximum of 50 characters.

Please note that EF Core does not support collections of primitive types like `List<string>`. You might need to create another class for `Temperature` and use `List<Temperature>` instead. Here is an example:

```
C#
public class TemperatureData
{
    [Key]
    public int Id { get; set; }
    public List<Temperature> Temperatures { get; set; }
}
public class Temperature
{
    [Key]
    public int Id { get; set; }
    [Required]
    [StringLength(50)]
    public string Name { get; set; }
}
```

In this case, each `Temperature` object has an `Id` and a `Name` (the actual temperature description). The `TemperatureData` class now has a list of `Temperature` objects.

Comme le chat IA le suggère, les anciennes versions d'EF Core ne prennent pas en charge les collections de types primitifs, contrairement à **EF Core 8**. Ceci est un bon exemple d'**interaction entre votre expertise et le prompting**. En omettant de bien examiner les résultats fournis par l'IA, vous prenez le risque que quelqu'un essaye d'appliquer ce code à une base de code plus ancienne où il ne fonctionnerait pas. **Vérifiez toujours attentivement les réponses obtenues, quel que soit l'outil d'IA que vous utilisez.**

Vous pouvez poursuivre avec les prompts pour demander la modification du code JSON afin de le rendre plus proche du modèle. L'IA répond avec une liste de températures dans un tableau de données de température.

Your query ...
 Create JSON to match the C# classes

AI Assistant ...

Sure, here is the JSON representation that matches the updated C# classes:

Json

```
{
  "temperatureData": {
    "id": 1,
    "temperatures": [
      {
        "id": 1,
        "name": "Freezing"
      },
      {
        "id": 2,
        "name": "Bracing"
      },
      {
        "id": 3,
        "name": "Chilly"
      },
      {
        "id": 4,
        "name": "Cool"
      },
      {
        "id": 5,
        "name": "Mild"
      },
      {
        "id": 6,
        "name": "Warm"
      },
      {
        "id": 7,
        "name": "Balmy"
      },
      {
        "id": 8,
        "name": "Hot"
      },
      {
        "id": 9,
        "name": "Sweltering"
      },
      {
        "id": 10,
        "name": "Scorching"
      }
    ]
  }
}
```

This JSON object represents a `TemperatureData` object with an `Id` and a list of `Temperature` objects. Each `Temperature` object has an `Id` and a `Name`.

Vous pouvez continuer de modifier les prompts en fonction des résultats jusqu'à obtenir exactement ce que vous souhaitez.

Remarque : la **fenêtre d'outils AI Assistant** n'apporte aucune modifications au contenu de l'éditeur. Lorsque vous utilisez des actions qui produisent du code, ce code s'affiche dans la fenêtre de l'AI Assistant. et vous devez créer les fichiers requis et copier la sortie.

III - Sauvegarder des prompts personnalisés

Maintenant que vous avez créé des prompts de qualité, vous pouvez les enregistrer afin de pouvoir les réutiliser. Ils peuvent aussi être utiles pour créer du contexte pour des requêtes avancées. ReSharper et Rider comportent tous les deux une bibliothèque de prompts pour stocker vos prompts personnalisés. **Le stockage de ces prompts dans votre bibliothèque vous permet de conserver les « macros d'IA » pour des tâches courantes**, comme :

- Instructions d'intégration ;
- Explications du code ;
- Déplacement et transformation de données ou de code ;
- Écrire ou clarifier des messages de commits ;
- Et bien plus...

Plusieurs des actions de l'IDE proposent une option d'enregistrement de prompts dans la bibliothèque. Par exemple, l'utilisation de l'**AI Assistant** avec Alt+Entrée révèle des prompts qui sont immédiatement disponibles, comme :

- Explain this...
- Find issues in...
- Refactor this...
- Is this model complete?

Vous pouvez personnaliser ces prompts et en ajouter les nouvelles versions dans la bibliothèque ou créer votre propre bibliothèque à partir de prompts que vous avez créés. Pour créer un prompt, utilisez le jeton `$SELECTION` pour représenter le bloc de code que l'AI Assistant doit utiliser comme espace réservé pour la partie de la requête qui est envoyée au LLM.

Pendant la création de votre prompt personnalisé, vous pouvez utiliser l'IA pour l'améliorer. C'est un peu comme une « Promptception ». **Lorsque votre équipe dispose d'un bon ensemble de prompts, vous pouvez automatiser certaines tâches qui seraient normalement effectuées manuellement**, telles que la création de classes basées sur JSON ou un schéma de base de données, la génération de tests ou tout autre travail dont vous souhaitez vous décharger.

IV - Conclusion

Aujourd'hui, il est essentiel pour les développeurs d'intégrer l'IA dans leurs processus de travail au quotidien, tant au niveau individuel que pour les équipes. L'AI Assistant de JetBrains offre une solution simple et rapide pour utiliser l'IA de façon efficace et précise dans le code. Gardez en tête que la pertinence des prompts et l'examen des réponses fournies sont deux éléments cruciaux pour tirer pleinement parti de l'IA. Si vous créez des prompts trop vagues ou non compréhensibles par l'humain, il est fort probable que l'IA ne puisse pas les comprendre non plus et vous n'obtiendrez pas les résultats attendus. De même, si l'IA renvoie des informations incorrectes, vous devez être en mesure de le détecter et de comprendre pourquoi. Continuez de développer vos compétences pour profiter des avantages de l'IA et essayez l'**AI Assistant de JetBrains** pour découvrir tout ce qu'il peut vous apporter.