Challenge5 Braindump

Alex Fletcher edited this page 1 hour ago · 1 revision

Edit New page

Challenge 5 - Braindump

AI can help us again. Not only with coding but also in solving our problems. We do not need to write a complicated parser. We can use a LLM to parse the files for us! We can use Semantic Kernel to help us to implement this within our .NET code.

Semantic Kernel

There is already a pull request that contains the file parser I wrote last night. It already includes the Semantic Kernel Libaries and the only thing that needs to be done is write the right prompts for the AI and we have the solution.

I found this amazing BlogPost by some dude called Duncan Roosma, that describes all we need to do. I don't know if it is a coincidence but he is doing the exact thing we need, but then for cars. We need it for our concerts. Check out this post of Duncan about using Semantic Kernel

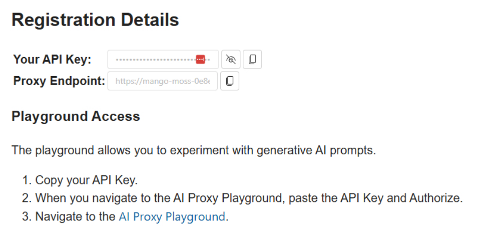
I used it as my inspiration. Below, I describe the rough outline what we need to do

Merge my PR

First of all, you need to merge my PR that contains a boilerplate solution for this. It adds the EventImporter project to our solution, where we can add the Semantic Kernel logic.

Getting access to OpenAI

To get a token from Azure OpenAI we use our own central proxy at https://openai.globoticket.com/event/2bbe-5922. You can go to it and authenticate with your GitHub Account. The 'Login in with GitHub' is on the top right. After registering you can scroll down and find the url and API token you can use to connect to our API:



Registration Details

Your API Key: ·························

Proxy Endpoint: https://mango-moss-0e8

Playground Access

The playground allows you to experiment with generative AI prompts.

1. Copy your API Key.
2. When you navigate to the AI Proxy Playground, paste the API Key and Authorize.
3. Navigate to the AI Proxy Playground.

To be able to use the key, make a Codespace secret where you add this OpenAI Key. Add a codespace secret called `OPENAIKEY`. Check out how to do this here

Pseudocode

I already created quite a lot of code. This is what it does

- Get the full files with all events (`Parse` method) from storage account
- `Chunk` the files with a LLM to split the file into separate events
- Process the separate chunks to create a CreateEventRequest object (`ParseEvent`). I suggest to convert the event text with the LLM to a Json object and serialize that

Parse

We need to parse the files. I added the files we need to parse already in our storage account. In the file `SemanticKernelSettings.cs` you see a reference to these files

```
"https://gdexcdn.azureedge.net/data/email.txt"
"https://gdexcdn.azureedge.net/data/edifact.txt"
"https://gdexcdn.azureedge.net/data/unparsable.txt"
```

No work for you here.

Chunking

We need to create chunks of our files that we can feed into the model. Since we never know how long the files will be we need to break it up. I already created the `Chunk` function.

LLM's are not deterministic. If it does not give the right results, I suggest you follow Duncans Blogpost and use GitHub Copilot to help you experiment to get the prompts just ok.

No work for you here unless the LLM has the wrong response.

Parse Events

Now that we have our individual events, we need to get them in our database. When you build the solution, it breaks at `ParseEvents`. We need to implement this.

You can use the the same code as used in the `Chunk` method, only you need to adjust the prompt. Adjust it in such a way, that it parses an event text and outputs a JSON string.

You will need to find the C# class called `EventFileParser`. It contains a method called `ParseEvents`. This method breaks up the file we are reading from storage and then gives us chunks that can be processed by the LLM.

The only thing you need to do is come up with the right prompts for the System prompt and voila, our unparsable file will turn into a json string we can process.

A good apprach is a few shot prompt, where we first tell the LLM what we expect and also build history of what the LLM returns. This is explained here in this blogpost. In Semantic Kernel we call this a ChatHistory.

It consists of 3 parts

- A System prompt
- An User prompt
- An Assistant prompt

Example of system prompt

```
    You are tasked with converting a user's description of a music event into a structured JSON format.
    Only the description provided in the latest user input should be processed into the output. Ignore all previous ir
    Follow this template:
    {
        "ContextId": "the id of the question asked by the user",
        "Artist": {
            "Name": "extracted artist name",
            "Genre": "extracted genre, if available",
        },
        "Name": "extracted event name",
        "Venue": "extracted event location",
        "Date": "date in YYYY-MM-DD format",
        "Description": "concise event description",
        "Price": extracted price as integer converted to dollar
    }
```

The sample below, is an example of what it should expect in the User prompt

```
On the cusp of midnight, when the world slips into the unseen wavelengths of the 10th of January 2024, at the cr
```

This is an example of a Assistant prompt

```
    {
        "Artist": {
            "Name": "BTS",
            "Genre": null
        },
        "Name": "Reverie Rendezvous",
        "Venue": "Qudos Bank Arena",
        "Date": "2024-01-10",
        "Description": "an unseen spectrum beneath the celestial garnish",
        "Price": 144
```

+ Add a custom sidebar

Clone this wiki locally

`https://github.com/globaldevopsexper`

```
    }
```

And you see pure magic! Set a breakpoint or `Console.Writeline`, to see this happen!

If you want to add the events to the catalog service, make sure you start the service locally as well, or point to the service online by pointing to the right urls.

Plugins

Oh yeah, before I forget, we of course want all tickets to be in USD currency. I looked at the files and I see all kinds of currencies in there. For this you can use a so called Plugin and register this with the Kernel. The moment it needs to do a conversion, it will call this function and you can implement the way you want to convert it. I found this free Currency converter API that we can use and call it andget USD values back. This is the one I mean: https://freecurrencyapi.com/.

In Duncan's Post he mentions something like ToolCallBehavior to enable this. I already created the plugin scaffolding.

Just create an `HttpClient` and call the api and register the plugin with the kernel. Easy Peasy ;-).

Oh and you can always ask Copilot to tell you how to write the code ...

Inserting Smart Components into the existing form

I saw this great new feature! NET Smart Components.. It allows us to use AI enabled components in the UI. I think the Smart Paste would really be helpful.

We can use the API key and endpoint we created earlier. Here is descriobed how to configure this for SmartComponents

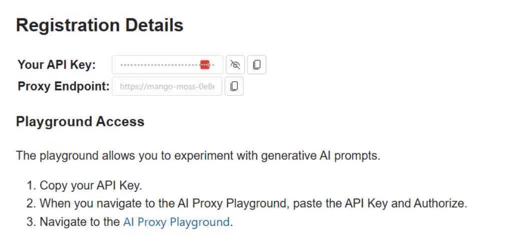
In the FrontEnd project, we can easily add this to the `EventCatalog` edit form. Here is some description how to add this

Maybe you can try if it works by using these sample emails

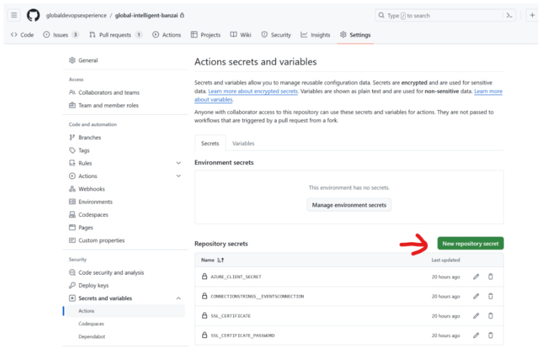
```
Hey!
Guess who's coming to perform next month? It's Joni Mitchell! I am so thrilled. The event's called Mirage Moments and
I can already picture how magical the stadium is going to look filled with twilight and timeless tunes. Joni's songs a
The price though, is a cool 414! That's slightly pinching my pocket but when I think about the priceless memories of s
I wonder if we could pull together a group to make it even more fun. Let me know your thoughts! It would be such a far
Catch you later!

Hey!
I had to share this with you - our favorite violinist, Anne-Sophie Mutter, has a concert coming up! Honestly, I'm as g
The event's named 'Twilight Tunes', rather mystical, don't you think? It does seem fitting, given she makes melodies t
Anyway, the ticket's going for 217 bucks - a steal if you ask me. Should we take the plunge? Do think about it, but nc
Talk soon.
```

OpenAI Enablement

I already mentioned it shortly above but.. do not forget to get the keys from OpenAI.

> Note To get a token from Azure OpenAI we use our own central proxy at https://openai.globoticket.com/event/2bbe-5922. You can go to it and authenticate with your GitHub Account. The 'Login in with GitHub' is on the top right. After registering you can scroll down and find the url and API token you can use to connect to our API:
>
> 
>
> ### Registration Details
>
> **Your API Key:** [redacted]
>
> **Proxy Endpoint:** https://mango-moss-0e8
>
> #### Playground Access
>
> The playground allows you to experiment with generative AI prompts.
>
> 1. Copy your API Key.
> 2. When you navigate to the AI Proxy Playground, paste the API Key and Authorize.
> 3. Navigate to the AI Proxy Playground.

The you need to copy the API key and put this in the code space environment variable OPENAIKEY. This is done by creating a new repository secret in the settings of the repo as shown in the below image.



Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. Learn more about encrypted secrets. Variables are shown as plain text and are used for **non-sensitive** data. Learn more about variables.

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

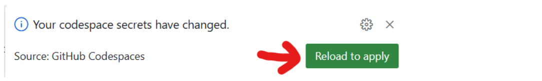
Environment secrets

This environment has no secrets.

Repository secrets

Name	Last updated
AZURE_CLIENT_SECRET	20 hours ago
CONNECTIONSTRINGS__EVENTSCONNECTION	20 hours ago
SSL_CERTIFICATE	20 hours ago
SSL_CERTIFICATE_PASSWORD	20 hours ago

After saving the variable, you will see the following pop-up apear in your code space



Your codespace secrets have changed.

Source: GitHub Codespaces

make sure you click the reload and apply, so you have the secret in your working environment. Don't worry, the reload will not lose any data, the codespace will restart and you have everything available as it was before the restart.

You can run the application from the command-line by typing: `dotnet run eventImporter.csproj` This will show you the output of the results in the console.

+ Add a custom footer

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information