

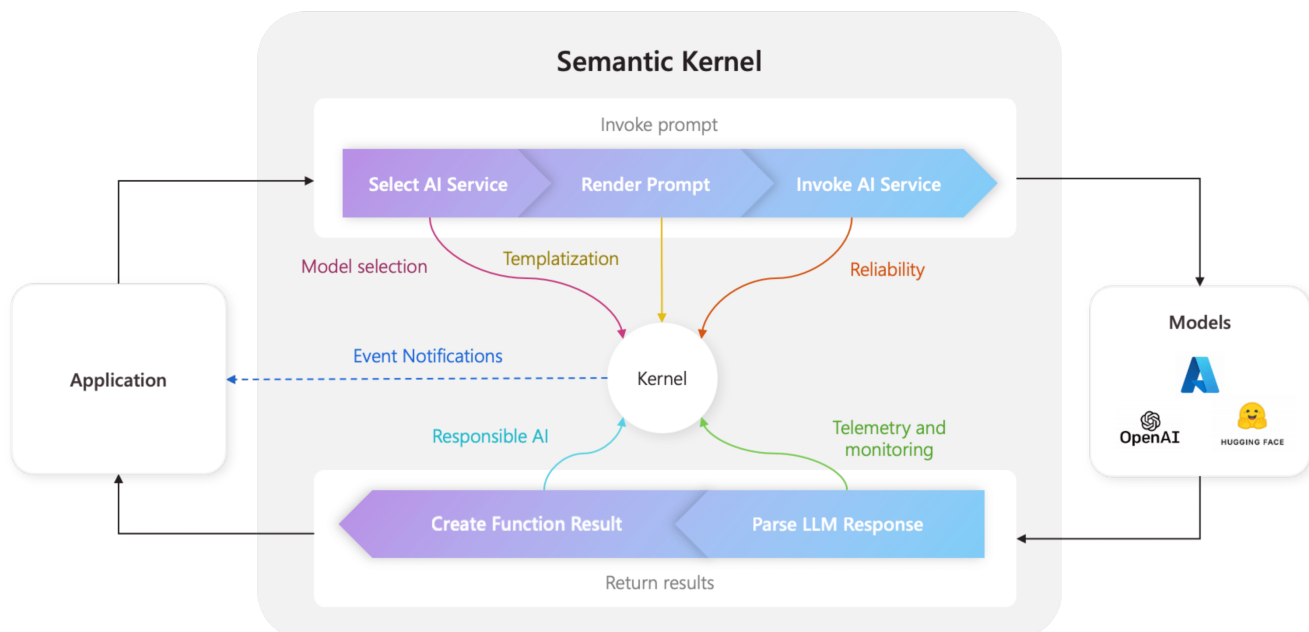
Understanding the kernel

Article • 06/24/2024

The kernel is the central component of Semantic Kernel. At its simplest, the kernel is a Dependency Injection container that manages all of the services and plugins necessary to run your AI application. If you provide all of your services and plugins to the kernel, they will then be seamlessly used by the AI as needed.

The kernel is at the center of your agents

Because the kernel has all of the services and plugins necessary to run both native code and AI services, it is used by nearly every component within the Semantic Kernel SDK to power your agents. This means that if you run any prompt or code in Semantic Kernel, the kernel will always be available to retrieve the necessary services and plugins.




This is extremely powerful, because it means you as a developer have a single place where you can configure, and most importantly monitor, your AI agents. Take for example, when you invoke a prompt from the kernel. When you do so, the kernel will...

1. Select the best AI service to run the prompt.
2. Build the prompt using the provided prompt template.
3. Send the prompt to the AI service.
4. Receive and parse the response.
5. And finally return the response from the LLM to your application.

Throughout this entire process, you can create events and middleware that are triggered at each of these steps. This means you can perform actions like logging, provide status updates to users, and most importantly responsible AI. All from a single place.

Build a kernel with services and plugins

Before building a kernel, you should first understand the two types of components that exist:

 Expand table

| Components | Description |
|------------|--|
| 1 Services | These consist of both AI services (e.g., chat completion) and other services (e.g., logging and HTTP clients) that are necessary to run your application. This was modelled after the Service Provider pattern in .NET so that we could support dependency ingestion across all languages. |
| 2 Plugins | These are the components that are used by your AI services and prompt templates to perform work. AI services, for example, can use plugins to retrieve data from a database or call an external API to perform actions. |

Import the necessary packages:

Python

```
from semantic_kernel import Kernel
from semantic_kernel.connectors.ai.open_ai import AzureChatCompletion
from semantic_kernel.core_plugins.time_plugin import TimePlugin
```

Next, you can create a kernel.

Python

```
# Initialize the kernel
kernel = Kernel()
```

Finally, you can add the necessary services and plugins. Below is an example of how you can add an Azure OpenAI chat completion, a logger, and a time plugin.

Python

```
# Add the Azure OpenAI chat completion service
kernel.add_service(AzureChatCompletion(model_id, endpoint, api_key))

# Add a plugin (the LightsPlugin class is defined below)
kernel.add_plugin(
    TimePlugin(),
    plugin_name="TimePlugin",
)
```

Next steps

Now that you understand the kernel, you can learn about all the different AI services that you can add to it.

[Learn about AI services](#)