Welcome!

Welcome to the Sisk Framework! This project was initially created to explore the capabilities of the native .NET HttpListener and has gradually evolved into a more commercially oriented framework as I began applying it to personal and commercial projects.

What is Sisk?

Sisk is a lightweight, agnostic, simple, and robust web development framework. Its core idea is to create a service that runs on the internet, following the patterns you define. Moreover, Sisk adapts to your preferred working style, rather than the other way around.

Due to its explicit nature, Sisk's behavior is predictable. The main differentiator from ASP.NET is that Sisk can be up and running with very few lines of code, avoiding unnecessary configurations and requiring minimal setup to get your server operational. Additionally, it does not require any additional .NET SDK packages for development, as the base package of .NET 6 is sufficient to start working with Sisk.

Sisk's philosophy emphasizes simplicity and provides all the essential tools for building cloud applications without relying on specific or proprietary technologies from other companies. However, the Sisk environment allows for the implementation of both proprietary and non-proprietary technologies, enabling their use, development, and support within the framework. Furthermore, the term "framework" refers to the collection of methods, tools, and libraries that make Sisk a complete web development ecosystem. Contributions must adhere to this philosophy and maintain readable, maintainable code that anyone with a basic understanding can read, maintain, and compile.

Sisk has a strict policy of transparency in its code. All technologies used to build Sisk **must** be open source, traceable, maintainable, and compilable, allowing anyone to edit, view, and create their own version of Sisk.

What is Sisk for?

With Sisk, you can create RESTful applications, Web sockets, file servers, GraphQL APIs, Entity Framework integrations, and more—essentially, whatever you need. Sisk is an extremely modular and sustainable framework. Its current development is intense, with much more to be added, but the focus remains on keeping it simple, easy to maintain, and enjoyable for developers working on projects of any size.

Sisk has also been tested in low-performance environments, such as machines with less than 1GB of RAM, handling over 20,000 requests per second. The code, from the moment it arrives on the server to the response sent back, is extremely concise, with very few steps before reaching the client.

One of the pillars of developing with Sisk is its compatibility with any machine that supports .NET, including those that do not require Native AOT. Additional implementations are also provided within the Sisk ecosystem, such as porting projects to other machines using configuration files and a view engine based on LISP, among others, served with packages beyond the core Sisk package. By design, Sisk is built to work with routers, but you are not obligated to use them. Sisk provides all the necessary infrastructure to create a secure application without obfuscating your code.

There's no need for excessive ceremony, fluff, or spending hours on tedious documentation. Sisk is simple and elegant in its syntax, facilitating the development of fast and complex systems.

But why not just use ASP.NET?

ASP.NET is a great and well-established web framework, and many features present in Sisk were inspired by it. However, Sisk focuses on simpler and more performant development, eliminating the need to install additional components in your system, project, or editor. Sisk was designed to be straightforward and robust, enabling you to create anything you desire.

Moreover, its development model allows you to choose how you want to work. You can handle requests in a simple, efficient, explicit, and fast manner. A basic understanding of HTTP is required if you want to manage everything manually, but even then, Sisk can greatly simplify the process with the functions it provides in its core package.

Getting started with Sisk is easy. Those with prior web development experience typically learn Sisk in one or two days. The Sisk documentation is a valuable resource, serving not just as a specification but as a complete manual with examples and support.

You can get started with Sisk here.

Let's get started with Sisk.

Getting started

Welcome to the Sisk documentation!

Finally, what is the Sisk Framework? It is an open-source lightweight library built with .NET, designed to be minimalist, flexible, and abstract. It allows developers to create internet services quickly, with little or no necessary configuration. Sisk makes it possible for your existing application to have a managed HTTP module, complete and disposable or complete.

Sisk's values include code transparency, modularity, performance, and scalability, and can handle various types of applications, such as Restful, JSON-RPC, Web-sockets, and more.

It's main features includes:

Resource	Description
Routing	A path router that supports prefixes, custom methods, path variables, value converters and more.
Request Handlers	Also known as *middlewares*, provides an interface to build your own request-handlers that work with the request before or after an action.
Compression	Compress your response contents easily with Sisk.
Web sockets	Provides routes that accept complete web-sockets, for reading and writing to the client.
Server-sent events	Provides the sending of server events to clients that support the SSE protocol.
Logging	Simplified logging. Log errors, access, define rotating logs by size, multiple output streams for the same log, and more.
Multi-host	Have an HTTP server for multiple ports, and each port with its own router, and each router with its own application.
Server handlers	Extend your own implementation of the HTTP server. Customize with extensions, improvements, and new features.

First steps

Sisk can run in any .NET environment. In this guide, we will teach you how to create a Sisk application using .NET. If you haven't installed it yet, please download the SDK from here.

In this tutorial, we will cover how to create a project structure, receive a request, obtain a URL parameter, and send a response. This guide will focus on building a simple server using C#. You can also use your favorite programming language.

> **NOTE**
>
> You may be interested in a quickstart project. Check this repository for more information.

Creating a Project

Let's name our project "My Sisk Application." Once you have .NET set up, you can create your project with the following command:

```
dotnet new console -n my-sisk-application
```

Next, navigate to your project directory and install Sisk using the .NET utility tool:

```
cd my-sisk-application
dotnet add package Sisk.HttpServer
```

You can find additional ways to install Sisk in your project here.

Now, let's create an instance of our HTTP server. For this example, we will configure it to listen on port 5000.

Building the HTTP Server

Sisk allows you to build your application step by step manually, as it routes to the HttpServer object. However, this may not be very convenient for most projects. Therefore, we can use the builder method, which makes it easier to get our app up and running.

Program.cs C#

```
class Program
{
    static async Task Main(string[] args)
    {
        using var app = HttpServer.CreateBuilder()
            .UseListeningPort("http://localhost:5000/")
            .Build();

        app.Router.MapGet("/", request =>
        {
            return new HttpResponse()
            {
                Status = 200,
                Content = new StringContent("Hello, world!")
            };
        });

        await app.StartAsync();
    }
}
```

It's important to understand each vital component of Sisk. Later in this document, you will learn more about how Sisk works.

Manual (advanced) setup

You can learn how each Sisk mechanism works in this section of the documentation, which explains the behavior and relationships between the HttpServer, Router, ListeningPort, and other components.

Namespace Sisk

Namespaces

Sisk.BasicAuth

Sisk.Cadente

Sisk.Core

Sisk.Documenting

Sisk.Helpers

Sisk.IniConfiguration

Sisk.JsonRPC

Sisk.ModelContextProtocol

Sisk.Ssl