



**راهنمای جامع چهارچوب ترکیبی ماهی  
آموزش ساخت صفحات وب و API**

نویسنده مهدی خلیل زاده

در ابتدا، از پدر و مادر عزیزم تشکر می‌کنم که همیشه با عشق و حمایت بی‌قید و شرط خود، به من قدرت تفکر، آزادی ایده‌پردازی و فضایی برای رشد دادند. قدردان زحمات و فداکاری‌های آن‌ها هستم و این کتاب را به آن‌ها تقدیم می‌کنم.

فهرست مطالب

[فصل ۱: ماهی چیست 8](#_Toc187660793)

[چرا ماهی ؟ 8](#_Toc187660794)

[آشنایی با فریمورک ماهی 8](#_Toc187660795)

[ماهی چیست ؟ 8](#_Toc187660796)

[ویژگی های کلیدی ماهی چیست ؟ 8](#_Toc187660797)

[چرا Lua در توسعه وب؟ 8](#_Toc187660798)

[مزایای ماهی برای توسعه‌دهندگان 9](#_Toc187660799)

[استفاده از Lua به‌جای زبان‌های پیچیده‌تر 9](#_Toc187660800)

[سادگی و قابلیت شخصی‌سازی 9](#_Toc187660801)

[سرعت و عملکرد 9](#_Toc187660802)

[سبک و کم حجم 10](#_Toc187660803)

[مناسب برای پروژه‌های خاص و اختصاصی 10](#_Toc187660804)

[پشتیبانی از APIهای سریع و قابل گسترش 10](#_Toc187660805)

[یادگیری سریع 10](#_Toc187660806)

[فصل 2: نصب و راه اندازی 11](#_Toc187660807)

[پیش‌نیازها 11](#_Toc187660808)

[.NET 8.0 یا بالاتر 11](#_Toc187660809)

[دانلود و نصب ماهی 11](#_Toc187660810)

[نصب ماهی در ویندوز 11](#_Toc187660811)

[نصب ماهی در لینوکس 12](#_Toc187660812)

[اجرای اولین پروژه 12](#_Toc187660813)

[فصل 3: ساختار پروژه در ماهی 13](#_Toc187660814)

[فصل ۴: مفاهیم پایه ماهی 14](#_Toc187660815)

[فصل ۵: مدیریت درخواست‌های HTTP 14](#_Toc187660816)

[فصل ۶: کار با کنترلرها 14](#_Toc187660817)

[فصل ۷: ساخت API با ماهی 14](#_Toc187660818)

[فصل ۸: پروژه‌های عملی 14](#_Toc187660819)

[فصل ۹: اشکال‌زدایی و بهینه‌سازی 14](#_Toc187660820)

[فصل ۱۰: منابع و مراجع 14](#_Toc187660821)

فهرست مطالب

فصل ۱: ماهی چیست

* آشنایی با فریمورک ماهی
* چرا Lua در توسعه وب؟
* مزایای ماهی برای توسعه‌دهندگان

فصل ۲: نصب و راه‌اندازی

* پیش‌نیازها
* دانلود و نصب ماهی
* اجرای اولین پروژه

فصل ۳: ساختار پروژه در ماهی

* فایل‌ها و پوشه‌های اصلی
* نحوه مدیریت منابع ثابت
* تنظیمات اولیه پروژه

فصل ۴: مفاهیم پایه ماهی

* معرفی زبان Lua
* متغیرها، انواع داده و عملگرها
* توابع و ساختارهای کنترلی
* ماژول‌ها در Lua

فصل ۵: مدیریت درخواست‌های HTTP

* ساختار درخواست‌ها و پاسخ‌ها در Mahi
* تعریف مسیرها (Routes)
* استفاده از ماژول‌های HTTP

فصل ۶: کار با کنترلرها

* معرفی کنترلرها
* نوشتن کنترلرهای ساده
* ارسال و دریافت داده‌ها

فصل ۷: ساخت API با ماهی

* تعریف مسیرهای API
* ارسال پاسخ به صورت JSON
* مدیریت خطاها در API

فصل ۸: پروژه‌های عملی

* پروژه اول: ساخت یک وبلاگ ساده
* پروژه دوم: ایجاد یک سیستم مدیریت وظایف (To-Do List)
* پروژه سوم: طراحی API احراز هویت کاربران

فصل ۹: اشکال‌زدایی و بهینه‌سازی

* ابزارهای اشکال‌زدایی در Mahi
* بهبود عملکرد و امنیت
* نکات پیشرفته برای مدیریت پروژه‌های بزرگ

فصل ۱۰: منابع و مراجع

* مستندات رسمی فریمورک Mahi
* منابع یادگیری زبان Lua
* لینک‌های مفید و انجمن‌های آنلاین

# فصل ۱: ماهی چیست

## چرا ماهی ؟

نام "ماهی" الهام گرفته از سادگی، سبکی و روان بودن این فریمورک است. همان‌طور که ماهی در آب به راحتی و آزادانه حرکت می‌کند، این فریمورک هم با استفاده از Lua و ساختار ساده و آزادش، فرایند توسعه وب را روان‌تر و سریع‌تر و آسان‌تر می‌کند.

## آشنایی با فریمورک ماهی

### ماهی چیست ؟

ماهی یک فریمورک وب و مفسر سبک و انعطاف‌پذیر است که با زبان C# توسعه داده شده و به شما این امکان را می‌دهد تا صفحات وب، APIها، و ماژول‌های HTTP را با استفاده از زبان Lua طراحی و مدیریت کنید.

### ویژگی های کلیدی ماهی چیست ؟

قابلیت‌های وب:

پشتیبانی از مدیریت درخواست‌های HTTP، ساخت API و ماژول‌های HTTP .

ساختار ساده و کاربردی:

برخلاف فریمورک‌های سنتی MVC، در ماهی خبری از مدل و ویو نیست، و تمرکز بیشتر بر کنترلرها و ماژول‌هاست.

## چرا Lua در توسعه وب؟

ماهی به کمک [NLua](https://github.com/nlua/nlua) به‌ عنوان یک مفسر داخلی Lua عمل می‌کند، بنابراین شما می‌توانید کد Lua را به‌طور مستقیم اجرا کرده و از آن برای کنترل رفتار صفحات وب استفاده کنید همچنین این زبان بسیار سبک، ساده و سریع است که در ماهی از آن برای تعریف منطق برنامه و طراحی صفحات وب استفاده می‌شود.

## مزایای ماهی برای توسعه‌دهندگان

**ماهی** در مقایسه با فریمورک‌های دیگر مانند  **ASP.NET** و **PHP** مزایای خاصی دارد که به ویژگی‌های منحصر به فرد آن و فلسفه طراحی‌اش مربوط می‌شود. در ادامه به مزایای ماهی نسبت به این فریمورک‌ها می‌پردازیم:

### استفاده از Lua به‌جای زبان‌های پیچیده‌تر

شما می‌توانید به‌سادگی از قدرت C# برای توسعه قسمت‌های دیگر برنامه و از Lua برای منطق کسب‌وکار استفاده کنید، که این ترکیب باعث **انعطاف‌پذیری بیشتر** می‌شود.

### سادگی و قابلیت شخصی‌سازی

برخلاف ASP.NET که معماری MVC را برای ساختاردهی به برنامه‌ها استفاده می‌کند، ماهی فقط از کنترلرها و ماژول‌ها استفاده می‌کند، که این باعث می‌شود کدنویسی و پیکربندی آن برای پروژه‌های کوچک و متوسط **ساده‌تر و سریع‌تر** باشد.

### سرعت و عملکرد

زبان Lua از سرعت بالایی برخوردار است و به همین دلیل **در پروژه‌هایی که نیاز به پردازش سریع دارند، مناسب‌تر** است. همچنین، چون ماهی تنها از Lua برای منطق برنامه استفاده می‌کند، **سرعت بارگذاری و پردازش درخواست‌ها** نسبت به فریمورک‌های سنگین‌تر مانند ASP.NET یا PHP می‌تواند بهتر باشد.

### سبک و کم حجم

رخلاف ASP.NET که به‌طور پیش‌فرض وابستگی‌های زیادی به کتابخانه‌ها و چارچوب‌های دیگر دارد، ماهی فریمورکی سبک و کم‌حجم است که تنها به یک مفسر Lua و .NET نیاز دارد. این ویژگی باعث می‌شود که پروژه‌های مبتنی بر ماهی **کمتر منابع مصرف کنند** و **راه‌اندازی آن سریع‌تر** باشد.

### مناسب برای پروژه‌های خاص و اختصاصی

ماهی برای پروژه‌های خاص یا اختصاصی که ممکن است نیاز به سادگی و **کنترل دقیق بر روی کد** داشته باشند، ایده‌آل است. این ویژگی در صورتی که نیاز به یک **فریمورک بسیار سبک و شخصی‌سازی‌شده** باشد، خیلی مفید خواهد بود.

### پشتیبانی از APIهای سریع و قابل گسترش

ماهی به‌طور طبیعی به شما این امکان را می‌دهد که APIهای RESTful بسازید. در مقابل، در ASP.NET و PHP شما ممکن است نیاز به پیچیدگی‌های بیشتری برای ساخت APIها داشته باشید (مثلاً در ASP.NET، شما باید Web API را تنظیم کنید یا در PHP از فریمورک‌هایی مانند Laravel استفاده کنید.)

### یادگیری سریع

برخلاف ASP.NET یا PHP که ممکن است برای مبتدیان پیچیده به نظر برسند و نیاز به دانشی گسترده از سیستم‌ها و الگوهای مختلف (مانند MVC) داشته باشند، ماهی به دلیل سادگی زبان Lua و معماری ساده‌تر، برای افرادی که تازه‌کار هستند **یادگیری سریع‌تری دارد**.

# فصل 2: نصب و راه اندازی

## پیش‌نیازها

### .NET 8.0 یا بالاتر

همانطور که گفتیم ماهی بر پایه C# و Dotnet توسعه داده شده است، بنابراین نیاز است که .NET 8.0 یا نسخه‌های بالاتر روی سیستم شما نصب باشد.

## دانلود و نصب ماهی

### نصب ماهی در ویندوز

برای نصب ماهی برای سیستم عامل ویندوز میشود از دو راه اقدام کرد:

* 1. دانلود نسخه از پیش کامپایل شده

برای این کار کافی است به آدرس گیت هاب مخزن ماهی به آدرس <https://github.com/mehdika2/Mahi/releases> مراجعه کنید و آخرین نسخه از پیش کامپایل شده را از قسمت Assets دانلود و از حالت فشرده خارج، سپس فایل Mahi.exe را اجرا کنید.

* 1. دانلود سورس کد برنامه و کامپایل کردن آن

در این حالت شما میبایست [گیت](https://git-scm.com/downloads/guis) را روی سیستم خود نصب کنید و با اجرا دستورات زیر، مخزن ماهی و فردین که یک مخزن مورد نیاز ماهی میباشد را نصب کنید:

git clone <https://github.com/mehdika2/mahi>

git clone <https://github.com/mehdika2/fardin>

سپس با دستورات زیر به ترتیب به محل دانلود مخزن بروید و کد ها را کامپایل کنید:

cd mahi

dotnet build

### نصب ماهی در لینوکس

توجه داشته باشید که قبل از شروع شما باید dotnet را روی لینوکس خود نصب داشته باشید، سپس برای نصب ماهی روی لینوکس کافیست با دستورات زیر اقدام به نصب برنامه گیت روی لینوکس کنید:

sudo apt-get update

sudo apt-get install git

و از اینجا به بعد باید همانند [مرحله 2 نصب ماهی در لینوکس](#_نصب_ماهی_در) اقدام به دانلود کد مخزن کنید و آن را کامپایل کنید.

## اجرای اولین پروژه

بعد از دانلود و کامپایل کردن برنامه به مسیر زیر در پوشه ماهی بروید:

bin > Debug > {.net version}   
و یک فایل با نام appconfig.yml ایجاد کنید(این فایل فایل پیکر بندی اصلی ماهی میباشد) و محتوایت زیر را در آن بریزید:

DefaultPages:

- index.htmlua

سپس یک پوشه با نام wwwapp ایجاد کنید (این پوشه حاوی تمام محتویات و فایل های منطقی برنامه تحت وب شما خواهد بود) و در آن یک فایل با نام index.htmlua ایجاد کنید(فایل های با پسوند .htmlua فایل های دارای منطق و کد های html هستند) و محتوایت زیر را در آن بریزید:

$go("Hello world!")

سپس مسیر <http://127.0.0.1:1010> را در مرورگر خود باز کنید.

\* تبریک میگوییم، شما اولین صفحه وب خود را با ماهی ایجاد کردید! \*

# فصل 3: ساختار پروژه در ماهی

## ساختار فایل پیکر بندی (Config)

همانطور که مشاهده کردید، فایل اصلی پیکربندی ماهی یک فایل با پسوند .yml و از نوع YAML میباشد که مانند Json پرکاربرد و مانند INI ساده میباشد.

فایل پیکربندی این اجازه را به شما میدهد تا رفتار ماهی را هرطور که لازم است تایین کنید، این فایل شامل بخش های زیر میباشد:

#### BaseDirectory (پوشه اصلی)

این کلید برای تعریف پوشه اصلی برنامه تنظیم میشود که یک رشته به عنوان مقدار میگیرد.

مقدار پیش فرض: wwwapp

#### DefaultPages (صفحات پیش فرض)

#### این کلید برای تعریف صفحات پیش‌فرض استفاده می‌شود و مقادیری از نوع آرایه‌ای از رشته‌ها می‌پذیرد. زمانی که مسیر / درخواست شود، فریمورک به ترتیب از اولین مقدار آرایه شروع کرده و به دنبال فایل مربوطه می‌گردد. اگر فایل موردنظر پیدا شود(افزودن پسوند فایل ضروری میباشد):

* در صورت داشتن پسوند .html، محتویات فایل به‌طور مستقیم به کاربر ارسال میشود.
* در صورت داشتن پسوند .htmlua، ابتدا منطق برنامه اجرا شده و سپس خروجی تولید شده به کاربر پساخ داده میشود.

و در صورت پیدا نشدن فایل مربوطه، درخواست دهنده با خطای 404 مواجه میشود.

#### ExtentionRequired (ضرورت وجود .htmlua انتهای مسیر درخواست)

این کلید برای تایین این است که آیا در مسیر درخواست که یک به یک فایل منطقی اشاره دارد ، وجود پسوند فایل (.htmlua) ضرورتی دارد یا خیر و یک دو ارزشی یا Boolean به عنوان مقدار میگیرد.

مقدار پیش فرض: true

#### NotExtentionInUrl (ضرورت نبودن .htmlua انتهای مسیر درخواست)

این کلید زمانی تاثیر گذار است که مقدار ExtentionRequired برابر با false باشد و درصورتی که مقدار true به این کلید بدهید، دیگر نباید پسوند فایل منطقی (.htmlua) را در انتهای درخواست یک فایل منطقی قرار دهید.

مقدار پیش فرض: false

#### ConnectionStriongs (رشته های اتصال)

این کلید برای ذخیره سازی رشته های اتصال و استفاده ساده از آنها در کد ها میباشد، مقدار ورودی این کلید باید یک **فهرست نگاشت** یا Hash Map که هر دو مقدار آن باید رشته باشند است.

#### HttpModules (ماژول های درخواست ها)

این کلید برای تعریف ماژول های درخواست تنظیم میشود، فایل های ماژول باید در پوشه .modules در پوشه اصلی برنامه (یا همان wwwapp پیشفرض) با پسوند .lua تعریف شوند و یک نام برای ماژول به عنوان کلید و نام فایل آن را به عنوان مقدار به عنوان یک **فهرست نگاشت** برای مقدار این کلید تعریف کنید.

#### Routes (مسیر ها)

این کلید برای تعریف رفتار مسیریابی برنامه تنظیم میشود، ساختار این کلید از دیگر کلید ها پیچیده تر است و همینطور امکانات جالبتر و بیشتری در اختیار شما قرار میدهد.

این کلید فهرستی از Route به عنوان مقدار میگیرد، ساختار Route دارای کلید های زیر میباشد:

* **Type (نوع):** که دارای سه نوع Static(ثابت)، Dynamic(متغیر) و Regex میباشد.
* **URL (مسیر درخواست):** که یک رشته میباشد که تایین میکند آیا مسیر درخواست شده شامل Route مربوطه میباشد یا خیر.
* **Route (مسیر):** درصورت شامل شدن Route مربوطه، این کلید تعین میکند که پاسخ توسط کدام فایل منطقی یا کد پردازش و پاسخ داده شود.

#### ErrorPages (صفحات تعریف خطا)

#### FrobbidenPaths (مسیر های غیر مجاز)

#### DirectoryBrowsing (پیمایش پوشه ها)

## ساختار پوشه ها و فایل ها در ماهی

## مسیریابی در ماهی

## ساختار فایل HTMLUA در ماهی

# فصل ۴: مفاهیم پایه ماهی

# فصل ۵: مدیریت درخواست‌های HTTP

# فصل ۶: کار با کنترلرها

# فصل ۷: ساخت API با ماهی

# فصل ۸: پروژه‌های عملی

# فصل ۹: اشکال‌زدایی و بهینه‌سازی

# فصل ۱۰: منابع و مراجع