**Zig**

Here's a **complete guide** on how to create **.exe** (executable) and **.lib** (library) projects using **Zig**. It covers **project structure, file contents, compilation, and execution commands**.

**📌 Zig Project Setup: Creating Executables and Libraries**

**1️⃣ Installing Zig**

Before starting, make sure you have Zig installed. Check by running:

zig version

If Zig is not recognized, follow the [installation guide](https://ziglang.org/download/).

**2️⃣ Creating a Zig Executable (.exe) Project**

**Project Structure**

my\_exe\_project/

│── src/

│ └── main.zig

│── build.zig

│── zig-cache/

│── zig-out/

└── README.md

**📌 File Breakdown**

1. **src/main.zig** → The main entry point for the application.
2. **build.zig** → Zig's build script to configure the project.
3. **zig-cache/ & zig-out/** → Auto-generated directories for compiled files.
4. **README.md** → Project documentation.

**📝 Step 1: Writing the Main Program (main.zig)**

const std = @import("std");

pub fn main() void {

const stdout = std.io.getStdOut().writer();

stdout.print("Hello, Zig!\n", .{}) catch {};

}

**🛠️ Step 2: Writing the Build Script (build.zig)**

This script tells Zig how to compile the executable.

const std = @import("std");

pub fn build(b: \*std.Build) void {

const exe = b.addExecutable(.{

.name = "my\_exe",

.root\_source\_file = .{ .path = "src/main.zig" },

});

b.installArtifact(exe);

}

**🔍 Explanation**

* b.addExecutable(...) → Creates an executable named **my\_exe**.
* .root\_source\_file = "src/main.zig" → Specifies the main Zig file.
* b.installArtifact(exe) → Ensures the compiled executable is placed in zig-out/bin.

**🚀 Step 3: Compiling and Running the Executable**

1. **Build the project**:
2. zig build
   * Creates zig-out/bin/my\_exe.exe.
3. **Run the executable**:
4. zig-out/bin/my\_exe.exe
   * Outputs:
   * Hello, Zig!
5. **Compile and run in one step**:
6. zig run src/main.zig
   * This compiles and immediately runs the program.
7. **Compile with optimizations** (e.g., release mode for better performance):
8. zig build -Drelease-fast

**3️⃣ Creating a Zig Library (.lib or .a)**

**Project Structure**

my\_lib\_project/

│── src/

│ ├── math\_lib.zig

│ └── main.zig

│── build.zig

│── zig-cache/

│── zig-out/

└── README.md

**📝 Step 1: Writing a Simple Library (math\_lib.zig)**

pub fn add(a: i32, b: i32) i32 {

return a + b;

}

pub fn subtract(a: i32, b: i32) i32 {

return a - b;

}

**📝 Step 2: Writing a Build Script for the Library (build.zig)**

const std = @import("std");

pub fn build(b: \*std.Build) void {

const lib = b.addStaticLibrary(.{

.name = "math\_lib",

.root\_source\_file = .{ .path = "src/math\_lib.zig" },

});

b.installArtifact(lib);

}

**🔍 Explanation**

* b.addStaticLibrary(...) → Creates a **static library** named math\_lib.
* b.installArtifact(lib) → Places the compiled library in zig-out/lib.

**🚀 Step 3: Compiling the Library**

1. **Build the library**:
2. zig build
   * Creates zig-out/lib/math\_lib.a (or .lib on Windows).

**4️⃣ Using the Library in an Executable**

**📝 Step 1: Writing a Program (main.zig) That Uses the Library**

const std = @import("std");

const math = @import("math\_lib.zig");

pub fn main() void {

const stdout = std.io.getStdOut().writer();

const result = math.add(10, 5);

stdout.print("10 + 5 = {}\n", .{result}) catch {};

}

**🚀 Step 2: Compiling the Executable with the Library**

1. **Manually compile with zig build**:
2. zig build
3. **Run the program**:
4. zig-out/bin/my\_exe.exe
   * Outputs:
   * 10 + 5 = 15

**Running a Standalone Zig File (One-Liner Compilation)**

If you want to run a single **Zig source file** as a console application **without a project setup**, you can directly execute:

zig run myfile.zig

Or compile it manually:

zig build-exe myfile.zig -o myfile.exe

Then run:

myfile.exe

**Key Zig Build Commands**

| **Command** | **Description** |
| --- | --- |
| zig build | Builds the project using build.zig. |
| zig build -Drelease-fast | Compiles in release mode (optimized). |
| zig build-exe src/main.zig -o main.exe | Compiles a single Zig file into an executable. |
| zig run src/main.zig | Compiles and runs the program in one step. |
| zig build-lib src/math\_lib.zig -o math\_lib.a | Compiles a Zig file into a static library. |

**🎯 Summary**

* **Executables (.exe)**: Use build.zig with b.addExecutable().
* **Libraries (.lib/.a)**: Use build.zig with b.addStaticLibrary().
* **Standalone Files**: Use zig run or zig build-exe.
* **Run Compiled Files**: Run from zig-out/bin/.

This guide gives you everything you need to **build Zig applications** and **libraries** efficiently. 🚀 Let me know if you need more details!