הקוד שלך הוא כלי שורת פקודה (CLI) ב-#C עם שני פקודות עיקריות:

1. **"bundle"** - מאחד קבצי קוד לקובץ בודד.
2. **"create-rsp"** - יוצר קובץ תגובה (response file) עם פרמטרים למאגר הפקודות, כדי להקל על הרצת פקודות חוזרות.

**הסבר כללי על פעולת הקוד:**

* כלי ה-CLI משתמש ב-System.CommandLine כדי להגדיר פקודות, אפשרויות וארגומנטים. כל פקודה מקבלת אפשרויות לביצוע משימות שונות, כמו הוספת שם מחבר, סידור קבצים לפי סוג, הסרת שורות ריקות ועוד.
* עבור פקודת "bundle", הכלי מאתר קבצים בספריה הנוכחית עם סיומות קבצים שתואמות לשפות התכנות שנבחרו, ומעתיק את תכניהם לקובץ מאוחד. הוא יכול לכלול את נתיב קובץ המקור כתגובה וגם לסדר את הקבצים לפי סדר אלפביתי או לפי סוג קובץ.
* עבור פקודת "create-rsp", הכלי שואל את המשתמש על ערכים שייכללו בקובץ תגובה, וכותב את הקובץ עם פרמטרים מובנים שאפשר להריץ בקלות לאחר מכן.

**קוד עם הערות:**

csharp

Copy code

using System.CommandLine; **// ייבוא ספרייה לפקודות CLI**

using System.Text;

**// הגדרת אפשרות קלט עבור הפלט המאוחד (קובץ האיחוד)**

var bundleOption = new Option<FileInfo>("--output", "File path and name");

**// הגדרת אפשרות קלט לרשימת שפות הקוד. יש לספק רשימה או "all" לכל הקבצים**

var languageOption = new Option<string[]>("--language", "List of programming languages (or 'all' for all files)")

{

IsRequired = true,

ArgumentHelpName = "language"

};

**// הגדרת אפשרות בוליאנית להכללת נתיב קבצים בתור תגובה בתוכן המאוחד**

var includeSourceOption = new Option<bool>("--include-source", "Include source code path and name as a comment in the bundle")

{

IsRequired = false,

ArgumentHelpName = "include-source"

};

**// הגדרת אפשרות לסידור הקבצים לפי סדר אלפביתי או לפי סוג קובץ**

var sortOption = new Option<string>("--sort", "Sort order for copying code files (alphabetical, type)")

{

Arity = ArgumentArity.ZeroOrOne,

ArgumentHelpName = "sort"

};

**// אפשרות להסרת שורות ריקות מהקבצים לפני איחוד**

var removeEmptyLinesOption = new Option<bool>("--remove-empty-lines", "Remove empty lines from the source code before copying")

{

IsRequired = false,

ArgumentHelpName = "remove-empty-lines"

};

**// אפשרות להוספת שם מחבר**

var authorOption = new Option<string>("--author", "Name of the file author")

{

IsRequired = true,

ArgumentHelpName = "author"

};

**// פקודת bundle: מאגדת את הקבצים לפי הקלט שהתקבל**

var bundleCommand = new Command("bundle", "Bundle code files to a single file");

**// הוספת האפשרויות לפקודת bundle**

bundleCommand.AddOption(bundleOption);

bundleCommand.AddOption(languageOption);

bundleCommand.AddOption(includeSourceOption);

bundleCommand.AddOption(sortOption);

bundleCommand.AddOption(removeEmptyLinesOption);

**// הגדרת הפעולה שתתבצע עם פקודת bundle**

bundleCommand.SetHandler((FileInfo output, string[] languages, bool includeSource, string sortOrder, bool removeEmptyLines) =>

{

try

{

var filesToBundle = new List<string>(); **// יצירת רשימה לשמירת הקבצים המאוחדים**

var currentDirectory = Directory.GetCurrentDirectory(); **// קבלת הנתיב הנוכחי**

**// קבלת סיומות קבצים לפי שפות שנבחרו**

string[] validExtensions;

if (languages.Contains("all"))

{

validExtensions = new[] { "\*.cs", "\*.java", "\*.py", "\*.js", "\*.cpp", "\*.h" }; **// סיומות כלליות לשפות נפוצות**

}

else

{

validExtensions = languages.Select(lang => GetFileExtension(lang)).ToArray();

}

**// איסוף הקבצים עם סיומות תואמות בספריה הנוכחית**

foreach (var ext in validExtensions)

{

filesToBundle.AddRange(Directory.GetFiles(currentDirectory, ext, SearchOption.TopDirectoryOnly)

.Where(file => !file.Contains("bin") && !file.Contains("debug")));

}

using (var fileStream = File.Create(output.FullName)) **// יצירת קובץ הפלט**

using (var writer = new StreamWriter(fileStream))

{

if (includeSource) **// הכללת תגובה בקובץ המאוחד**

{

writer.WriteLine($"// Source files included: {string.Join(", ", filesToBundle)}");

}

**אלפבתי או טייפ// סידור הקבצים לפי הסדר המבוקש**

if (sortOrder == "alphabetical")

{

filesToBundle = filesToBundle.OrderBy(f => Path.GetFileName(f)).ToList();

}

else if (sortOrder == "type")

{

filesToBundle = filesToBundle.OrderBy(f => Path.GetExtension(f)).ToList();

}

**// הוספת תוכן כל קובץ למאוחד**

foreach (var file in filesToBundle)

{

var fileContent = File.ReadAllLines(file);

if (removeEmptyLines) **// הסרת שורות ריקות**

{

fileContent = fileContent.Where(line => !string.IsNullOrWhiteSpace(line)).ToArray();

}

writer.WriteLine($"// File: {Path.GetFileName(file)}"); **// הוספת שם הקובץ כתגובה**

writer.WriteLine(string.Join(Environment.NewLine, fileContent));

}

Console.WriteLine("File was created successfully.");

}

}

catch (DirectoryNotFoundException)

{

Console.WriteLine("Error: File path is invalid.");

}

}, bundleOption, languageOption, includeSourceOption, sortOption, removeEmptyLinesOption);

**// פקודת create-rsp: יוצרת קובץ תגובה עם ערכי הפרמטרים**

var createRspCommand = new Command("create-rsp", "Create a response file with the specified options");

**// הוספת האפשרויות לפקודת create-rsp**

createRspCommand.AddOption(authorOption);

createRspCommand.AddOption(bundleOption);

createRspCommand.AddOption(languageOption);

createRspCommand.AddOption(includeSourceOption);

createRspCommand.AddOption(sortOption);

createRspCommand.AddOption(removeEmptyLinesOption);

**// הגדרת הפעולה שתתבצע עם פקודת create-rsp**

createRspCommand.SetHandler((string author, FileInfo output, string[] languages, bool includeSource, string sortOrder, bool removeEmptyLines) =>

{

var responseFileContent = new StringBuilder();

**// שאילת פרמטרים מהמשתמש**

string outputDirectory = PromptUser("Enter the output directory:");

string languagesInput = PromptUser("Enter the value for --language (comma-separated, e.g., 'java,csharp'):");

bool includeSourceInput = PromptUser("Should it include source path? (true/false):") == "true";

string sortOrderInput = PromptUser("Enter the sort order (alphabetical/type):");

bool removeEmptyLinesInput = PromptUser("Should it remove empty lines? (true/false):") == "true";

**// בדיקת קלט תקין**

if (string.IsNullOrWhiteSpace(outputDirectory) || string.IsNullOrWhiteSpace(sortOrderInput))

{

Console.WriteLine("Invalid input. Please provide valid values.");

return;

}

**// כתיבת תוכן קובץ התגובה**

responseFileContent.AppendLine($"# Author: {author}");

responseFileContent.AppendLine($"--output {outputDirectory}");

responseFileContent.AppendLine($"--language {languagesInput}");

responseFileContent.AppendLine($"--include-source {includeSourceInput}");

responseFileContent.AppendLine($"--sort {sortOrderInput}");

responseFileContent.AppendLine($"--remove-empty-lines {removeEmptyLinesInput}");

string responseFilePath = $"{author.ToLower()}.rsp";

File.WriteAllText(responseFilePath, responseFileContent.ToString()); **// יצירת קובץ התגובה**

Console.WriteLine($"Response file '{responseFilePath}' created. You can run it using: dotnet @\"{responseFilePath}\"");

}, authorOption, bundleOption, languageOption, includeSourceOption, sortOption, removeEmptyLinesOption);

**// יצירת פקודה ראשית המכילה את bundle ו-create-rsp**

var rootCommand = new RootCommand("CLI Tool for bundling code files");

rootCommand.AddCommand(bundleCommand);

rootCommand.AddCommand(createRspCommand);

await rootCommand.InvokeAsync(args); **// הפעלת הפקודה**

**// פונקציה לשאילת קלט מהמשתמש**

static string PromptUser(string question)

{

Console.WriteLine(question);

string? input = Console.ReadLine();

if (string.IsNullOrEmpty(input))

{

throw new ArgumentException("Input cannot be empty.");

}

return input;

}

**// פונקציה הממירה שם שפה לסיומת הקובץ המתאימה**

static string GetFileExtension(string language)

{

return language.ToLower() switch

{

"csharp" => "\*.cs", // אם השפה היא #C, הסיומת היא ".cs"

"java" => "\*.java", // אם השפה היא Java, הסיומת היא ".java"

"python" => "\*.py", // אם השפה היא Python, הסיומת היא ".py"

"javascript" => "\*.js", // אם השפה היא JavaScript, הסיומת היא ".js"

"cpp" => "\*.cpp", // אם השפה היא ++C, הסיומת היא ".cpp"

"c" => "\*.c", // אם השפה היא C, הסיומת היא ".c"

"html" => "\*.html", // אם השפה היא HTML, הסיומת היא ".html"

\_ => throw new ArgumentException($"Unsupported language: {language}"), // אם השפה לא נתמכת, נזרקת שגיאה

};

}

**הסבר על פונקציות עזר:**

* PromptUser: פונקציה שמציגה שאלה למשתמש וממתינה לתשובה מהקונסול. אם התשובה ריקה, נזרקת שגיאה. פונקציה זו עוזרת בהפיכת הקוד לדינמי ומקבלת קלט לשימוש מאוחר יותר בקובץ התגובה.
* GetFileExtension: פונקציה שמקבלת את שם השפה (לדוגמה: "csharp") וממירה אותה לסיומת הקובץ המתאימה (לדוגמה: "\*.cs"). היא משתמשת ב-switch כדי לזהות את השפה, וכך הכלי יכול להתמודד עם סוגים שונים של קבצי קוד.

**סיכום:**

הקוד שלך בונה כלי

CLI

רב-עוצמה שמסייע באיחוד קבצים לסוגים שונים וביצירת קובצי תגובה להרצה עתידית, עם אפשרויות גמישות לסידור ואיחוד הקוד.