Configuration Guide for Code Morphing Using ts-morph

This guide explains how to configure the project to perform code morphing automation using the **MergeConfig** interface. The configuration is critical for specifying paths, logic criteria, and additional details that guide the ts-morph operations.

Overview of Configuration

The configuration is defined in the **MergeConfig** interface and implemented in a config object. This setup specifies:

- Required imports and providers for the target file.
- The criteria for extracting validation logic from the template file.
- File paths for the source (template) and target (user) components.
- Class and method-level details for merging.

Configuration Variables in MergeConfig:

Variable	Type	Description
requiredImports	Array of Objects	Specifies the imports required in the target file.
requiredProviders	Array of Strings	Specifies Angular providers required in the target file.
validationLogicCriteria	String	A keyword to identify the validation logic in the template file.
templateFilePath	String	Path to the template file containing validation logic.
userFilePath	String	Path to the target Angular component to be modified.
userClassName	String	Name of the class in the target file.
newMethodName	String	Name of the method to add in the target file for validation logic.

newMethodCall	String	Code snippet to call the new method (usually in ngOnInit).
mergeProperties	Boolean (Optional)	Indicates whether to merge class properties from the template file.
mergeConstructor	Boolean (Optional)	Indicates whether to merge constructors from the template file.

Example of MergeConfig Interface:

```
export interface MergeConfig {
    requiredImports: {
        moduleSpecifier: string;
        namedImports: string[];
    }[];
    requiredProviders: string[];
    validationLogicCriteria: string;
    templateFilePath: string;
    userFilePath: string;
    userClassName: string;
    newMethodName: string;
    newMethodCall: string;
    mergeProperties?: boolean;
    mergeConstructor?: boolean;
}
```

Explanation of Each Configuration Variable

1. requiredImports

- Example:

```
requiredImports: [
  {
    moduleSpecifier: '@magic-xpa/angular',
    namedImports: ['TaskBaseMagicComponent', 'magicProviders']
  }]
```

2. requiredProviders

- Lists the providers that must be included in the @Component decorator of the target file.
- Example:

```
requiredProviders: ['[...magicProviders]']
```

3. validationLogicCriteria

- Specifies a keyword to locate the validation logic in the template file.
- Example:

```
validationLogicCriteria: 'this.fb.group'
```

4. templateFilePath

- Path to the file containing the template logic (source file).
- Example:

```
templateFilePath: 'C:/path/to/Registration.component.ts'
```

5. userFilePath

- Path to the file that will be updated (target file).
- Example:

```
userFilePath: 'C:/path/to/TestLoadProgram.component.ts'
```

6. userClassName

- Name of the class in the target file that will be modified.
- Example:

```
userClassName: 'TestLoadProgram'
```

7. newMethodName

- Name of the new method to be added to the target class.
- Example:

```
newMethodName: 'initializeRegistrationForm'
```

8. newMethodCall

- Code snippet to call the new method, usually added in ngOnInit.
- Example:

```
newMethodCall: 'this.initializeRegistrationForm();'
```

9. mergeProperties

- Indicates whether class properties from the template file should be merged.
- Default: false.

10. mergeConstructor

- Indicates whether constructors from the template file should be merged.
- Default: false.

How to Configure for Code Morphing

1. **Define the Configuration**:
   - Open the config.ts file.
   - Fill in the required paths, criteria, and other details as shown in the example.

   Example:

```
export const config: MergeConfig = {
    requiredImports: [
        {
            moduleSpecifier: '@magic-xpa/angular',
            namedImports: ['TaskBaseMagicComponent', 'magicProviders']
        }
    ],
    requiredProviders: ['[...magicProviders]'],
    validationLogicCriteria: 'this.fb.group',
    templateFilePath: 'C:/Users/Desktop/Task/Morphing/Registration
Component/Registration.component.ts',
    userFilePath: 'C:/Users/Desktop/Task/Morphing/Angular Sample
Component/src/app/magic/TestLoadModule/TestLoadProgram/TestLoadProgram.com
ponent.ts',
    userClassName: 'TestLoadProgram',
    newMethodName: 'initializeRegistrationForm',
    newMethodCall: 'this.initializeRegistrationForm();',
    mergeProperties: false,
    mergeConstructor: false
};
```

2. **Save the Configuration**:
    - Ensure paths are correct and accessible.
    - Verify that the validationLogicCriteria is consistent with the template file's logic.
3. **Run the Script**:
    - Execute the script to perform the morphing:

    node path/to/main/script.js
4. **Verify Output**:
    - Open the userFilePath file to confirm:
        - The new method is added.
        - Required imports and providers are present.
        - Validation logic is properly integrated.

By following this guide, you can efficiently configure and execute code morphing automation using ts-morph.