

## .NET のクラスライブラリ設計 改訂新版 目次

### 第1章 イン트로ダクション

- 1.1 良く設計されたフレームワークの品質
  - 1.1.1 良く設計されたフレームワークは単純である
  - 1.1.2 良く設計されたフレームワークの設計にはコストがかかる
  - 1.1.3 良く設計されたフレームワークはトレードオフに満ちている
  - 1.1.4 良く設計されたフレームワークは過去から拝借する
  - 1.1.5 良く設計されたフレームワークは進化するように設計される
  - 1.1.6 良く設計されたフレームワークはうまく統合されている
  - 1.1.7 良く設計されたフレームワークは一貫性がある

### 第2章 フレームワーク設計の基礎

- 2.1 漸進的なフレームワーク
- 2.2 フレームワーク設計の基本原則
  - 2.2.1 シナリオ駆動設計の原則
  - 2.2.2 参入障壁低減の原則
  - 2.2.3 自己説明的なオブジェクトモデルの原則
  - 2.2.4 レイヤー化されたアーキテクチャの原則
- まとめ

### 第3章 命名に関するガイドライン

- 3.1 大文字と小文字の使い分けの規約
  - 3.1.1 識別子の小文字と大文字の使い分けの規約
  - 3.1.2 頭字語の小文字と大文字の使い分け
  - 3.1.3 複合語や一般的な用語における小文字と大文字の使い分け
  - 3.1.4 小文字と大文字の区別
- 3.2 一般的な命名規約
  - 3.2.1 単語の選択
  - 3.2.2 略語と頭字語の使用
  - 3.2.3 言語固有の名前の回避
  - 3.2.4 既存の API の新しいバージョンの命名
- 3.3 アセンブリ、DLL、パッケージの名前
- 3.4 名前空間の名前
  - 3.4.1 名前空間や型の名前の衝突
- 3.5 クラス、構造体、インターフェイスの名前
  - 3.5.1 ジェネリック型パラメーターの名前
  - 3.5.2 一般的な型の名前
  - 3.5.3 列挙体の命名
- 3.6 型のメンバーの名前
  - 3.6.1 メソッドの名前
  - 3.6.2 プロパティの名前
  - 3.6.3 イベントの名前
  - 3.6.4 フィールドの命名
- 3.7 パラメーターの命名
  - 3.7.1 演算子のオーバーロードのパラメーターの命名

### 3.8 リソースの命名

まとめ

## 第4章 型の設計のガイドライン

### 4.1 型と名前空間

### 4.2 クラスか構造体かの選択

### 4.3 クラスかインターフェイスかの選択

### 4.4 抽象クラスの設計

### 4.5 静的クラスの設計

### 4.6 インターフェイスの設計

### 4.7 構造体の設計

### 4.8 列挙体の設計

#### 4.8.1 フラグ列挙体の設計

#### 4.8.2 列挙体への値の追加

### 4.9 ネスト型

### 4.10 型とアセンブリのメタデータ

### 4.11 厳密に型指定された文字列

まとめ

## 第5章 メンバーの設計

### 5.1 一般的なメンバー設計のガイドライン

#### 5.1.1 メンバーのオーバーロード

#### 5.1.2 インターフェイスメンバーの明示的実装

#### 5.1.3 プロパティかメソッドかの選択

### 5.2 プロパティの設計

#### 5.2.1 インデックス付きプロパティの設計

#### 5.2.2 プロパティ変更通知イベント

### 5.3 コンストラクターの設計

#### 5.3.1 型コンストラクターのガイドライン

### 5.4 イベントの設計

### 5.5 フィールドの設計

### 5.6 拡張メソッド

### 5.7 演算子のオーバーロード

#### 5.7.1 `==` 演算子のオーバーロード

#### 5.7.2 変換演算子

#### 5.7.3 比較演算子

### 5.8 パラメーターの設計

#### 5.8.1 列挙体かブール値かの選択

#### 5.8.2 引数のバリデーション

#### 5.8.3 パラメーターの渡し方

#### 5.8.4 可変数のパラメーターを持つメンバー

#### 5.8.5 ポインターパラメーター

### 5.9 メンバーのシグネチャでのタプルの使用

まとめ

## 第6章 拡張性を考慮した設計

- 6.1 拡張性メカニズム
  - 6.1.1 非シールクラス
  - 6.1.2 protected なメンバー
  - 6.1.3 イベントとコールバック
  - 6.1.4 仮想メンバー
  - 6.1.5 抽象（抽象クラスとインターフェイス）
- 6.2 基底クラス
- 6.3 シール化
- まとめ

## 第7章 例外

- 7.1 例外のフロー
- 7.2 スローする適切な例外型の選択
  - 7.2.1 エラーメッセージの設計
  - 7.2.2 例外の処理
  - 7.2.3 例外のラップ
- 7.3 標準の例外型の使用
  - 7.3.1 Exception と SystemException
  - 7.3.2 ApplicationException
  - 7.3.3 InvalidOperationException
  - 7.3.4 ArgumentException、ArgumentNullException、および ArgumentOutOfRangeException
  - 7.3.5 NullReferenceException、IndexOutOfRangeException、および AccessViolationException
  - 7.3.6 StackOverflowException
  - 7.3.7 OutOfMemoryException
  - 7.3.8 ComException、SEHException、および ExecutionEngineException
  - 7.3.9 OperationCanceledException と TaskCanceledException
  - 7.3.10 FormatException
  - 7.3.11 PlatformNotSupportedException
- 7.4 カスタム例外の設計
- 7.5 例外と性能
  - 7.5.1 Tester-Doer パターン
  - 7.5.2 Try パターン
- まとめ

## 第8章 使用法に関するガイドライン

- 8.1 配列
- 8.2 属性
- 8.3 コレクション
  - 8.3.1 コレクションであるパラメーター
  - 8.3.2 コレクション型のプロパティと戻り値
  - 8.3.3 配列かコレクションかの選択
  - 8.3.4 カスタムコレクションの実装
- 8.4 DateTime と DateTimeOffset
- 8.5 ICloneable
- 8.6 IComparable<T> と IEquatable<T>

- 8.7 IDisposable
- 8.8 Nullable<T>
- 8.9 Object
  - 8.9.1 Object.Equals
  - 8.9.2 Object.GetHashCode
  - 8.9.3 Object.ToString
- 8.10 シリアル化
- 8.11 Uri
  - 8.11.1 System.Uri の実装ガイドライン
- 8.12 System.Xml の使用法
- 8.13 等値演算子
  - 8.13.1 値型での等値演算子
  - 8.13.2 参照型での等値演算子
- まとめ

## 第9章 共通のデザインパターン

- 9.1 集約コンポーネント
  - 9.1.1 コンポーネント指向設計
  - 9.1.2 因子型
  - 9.1.3 集約コンポーネントのガイドライン
- 9.2 非同期パターン
  - 9.2.1 非同期パターンの選択
  - 9.2.2 タスクベース非同期パターン
  - 9.2.3 async メソッドの戻り値の型
  - 9.2.4 既存の同期メソッドに非同期版を追加する
  - 9.2.5 非同期パターンの一貫性に関する実装ガイドライン
  - 9.2.6 クラシック非同期パターン
  - 9.2.7 イベントベース非同期パターン
  - 9.2.8 IAsyncDisposable
  - 9.2.9 IAsyncEnumerable<T>
  - 9.2.10 await foreach の使用法のガイドライン
- 9.3 依存関係プロパティ
  - 9.3.1 依存関係プロパティの設計
  - 9.3.2 添付依存関係プロパティ
  - 9.3.3 依存関係プロパティのバリデーション
  - 9.3.4 依存関係プロパティの変更通知
  - 9.3.5 依存関係プロパティの値の強制
- 9.4 Dispose パターン
  - 9.4.1 基本Dispose パターン (Basic Dispose Pattern)
  - 9.4.2 ファイナライズ可能な型
  - 9.4.3 スコープ付き処理
  - 9.4.4 IAsyncDisposable
- 9.5 ファクトリ
- 9.6 LINQ サポート
  - 9.6.1 LINQ の概要
  - 9.6.2 LINQ サポートの実装方法

- 9.6.3 IEnumerable<T> による LINQ サポート
- 9.6.4 IQueryable<T> による LINQ サポート
- 9.6.5 クエリパターンによる LINQ サポート
- 9.7 オプション機能パターン
- 9.8 共変性と反変性
  - 9.8.1 反変性
  - 9.8.2 共変性
  - 9.8.3 偽共変性パターン
- 9.9 テンプレートメソッド
- 9.10 タイムアウト
- 9.11 XAML 読み取り可能型
- 9.12 バッファーを使用した処理
  - 9.12.1 データ変換処理
  - 9.12.2 サイズが固定または事前決定済みのバッファーに書き込む
  - 9.12.3 Try-Write パターンを使用して値を書き込む
  - 9.12.4 バッファーへの部分書き込みと OperationStatus
- 9.13 そして終わりに……

## 付録A C# コーディングスタイル規約

- A.1 一般的なスタイル規則
  - A.1.1 波かっこ ( { } ) の使用法
  - A.1.2 空白の使用法
  - A.1.3 インデントの使用法
  - A.1.4 縦方向の空白
  - A.1.5 メンバーの修飾子
  - A.1.6 その他
- A.2 命名規則
- A.3 コメント
- A.4 ファイルの編成

## 付録B 廃止されたガイダンス

- B.3 命名に関するガイドラインで廃止されたガイダンス
  - B.3.8 リソースの命名
- B.4 型の設計のガイドラインで廃止されたガイダンス
  - B.4.1 型と名前空間
- B.5 メンバーの設計で廃止されたガイダンス
  - B.5.4 イベントの設計
- B.7 例外の設計で廃止されたガイダンス
  - B.7.4 カスタムの例外の設計
- B.8 使用法のガイドラインで廃止されたガイダンス
  - B.8.10 シリアル化
- B.9 共通のデザインパターンで廃止されたガイダンス
  - B.9.2 非同期パターン
  - B.9.4 Dispose パターン

## 付録C API 仕様書のサンプル

## 付録D 破壊的変更 (breaking changes)

### D.1 アセンブリの変更

#### D.1.1 アセンブリの名前を変更

### D.2 名前空間の追加

#### D.2.1 既存の型と名前が衝突する名前空間を追加

### D.3 名前空間の変更

#### D.3.1 名前空間の大文字と小文字を変更

### D.4 型の移動

#### D.4.1 [TypeForwardedTo] 属性を介して型を移動

#### D.4.2 [TypeForwardedTo] 属性を使用せずに型を移動

### D.5 型の削除

#### D.5.1 型を削除

### D.6 型の変更

#### D.6.1 シールされていない型をシール

#### D.6.2 シールされた型のシールを解除

#### D.6.3 型名の大文字と小文字を変更

#### D.6.4 型の名前を変更

#### D.6.5 型の名前空間を変更

#### D.6.6 struct に readonly 修飾子を追加

#### D.6.7 struct から readonly 修飾子を削除

#### D.6.8 既存のインターフェイスに基底インターフェイスを追加

#### D.6.9 2 番目のジェネリックインターフェイス実装宣言を追加

#### D.6.10 class を struct に変更

#### D.6.11 struct を class に変更

#### D.6.12 struct を ref struct に変更

#### D.6.13 ref struct を (ref でない) struct に変更

### D.7 メンバーの追加

#### D.7.1 基底クラスのメンバーを new で隠す

#### D.7.2 abstract メンバーを追加

#### D.7.3 シールされていない型にメンバーを追加

#### D.7.4 シールされていない型に override メンバーを追加

#### D.7.5 struct に最初の参照型フィールドを追加

#### D.7.6 インターフェイスにメンバーを追加

### D.8 メンバーの移動

#### D.8.1 基底クラスにメンバーを移動

#### D.8.2 基底インターフェイスにメンバーを移動

#### D.8.3 派生型にメンバーを移動

### D.9 メンバーの削除

#### D.9.1 シールされていない型からファイナライザーを削除

#### D.9.2 シールされた型からファイナライザーを削除

#### D.9.3 オーバーライドできないメンバーを削除

#### D.9.4 仮想メンバーのオーバーライドを削除

#### D.9.5 抽象メンバーのオーバーライドを削除

#### D.9.6 シリアル化可能な型の非公開フィールドを削除

#### D.10 メンバーのオーバーロード

- D. 10. 1 メンバーの最初のオーバーロードを追加
- D. 10. 2 参照型のパラメーターに対する代替的パラメーターオーバーロードを追加
- D. 11 メンバーのシグネチャの変更
  - D. 11. 1 メソッドのパラメーターの名前を変更
  - D. 11. 2 メソッドのパラメーターを追加または削除
  - D. 11. 3 メソッドのパラメーターの型を変更
  - D. 11. 4 型が異なるメソッドパラメーターどうしを並び替え
  - D. 11. 5 型が同じメソッドパラメーターどうしを並び替え
  - D. 11. 6 メソッドの戻り値の型を変更
  - D. 11. 7 プロパティの型を変更
  - D. 11. 8 メンバーの可視性を public からそれ以外に変更
  - D. 11. 9 メンバーの可視性を protected から public に変更
  - D. 11. 10 仮想（または抽象）メンバーを protected から public に変更
  - D. 11. 11 static 修飾子を追加または削除
  - D. 11. 12 パラメーターを参照渡しへ、または参照渡しから変更
  - D. 11. 13 参照渡しパラメーターのスタイルを変更
  - D. 11. 14 struct のメソッドに readonly 修飾子を追加
  - D. 11. 15 struct のメソッドから readonly 修飾子を削除
  - D. 11. 16 必須パラメーターを省略可能パラメーターに変更
  - D. 11. 17 省略可能なパラメーターを必須パラメーターに変更
  - D. 11. 18 省略可能パラメーターの既定値を変更
  - D. 11. 19 const フィールドの値を変更
  - D. 11. 20 abstract メンバーを virtual に変更
  - D. 11. 21 virtual メンバーを abstract に変更
  - D. 11. 22 非 virtual メンバーを virtual に変更
- D. 12 振る舞いの変更
  - D. 12. 1 実行エラー例外を使用法エラー例外に変更
  - D. 12. 2 使用法エラー例外を機能するように動作変更
  - D. 12. 3 メソッドから返される値の型を変更
  - D. 12. 4 新しい種類のエラー例外をスロー
  - D. 12. 5 既にスローされている例外型から派生する新しい型の例外をスロー
- D. 13 最後に

## 付録 E 日本語版特別付録 訳者・監訳者による雑感