CM-01 アプリケーション起動・終了機能

■ 概要

本機能は、TERASOLUNA フレームワークを使ったアプリケーションの起動・終了を実施する。

- フレームワークの起動・終了
  TERASOLUNA フレームワークの起動・終了時は、エントリポイントとなるクラスから TerasolunaFramework クラスを呼び出す。フレームワーク起動時は、フレームワークの動作に必要な初期化処理を実施する。また、フレーム終了時は、フレームワーク終了のための後処理を実施する。
  実際の初期化・終了処理は、InitializationManager クラスが実施する。InitializationManager は、「CM-02 インスタンス管理機能」を利用して、IInitializer インタフェース実装クラスを取得し、初期化・終了処理を実施する。実行する IInitializer 実装クラスは、FW 起動構成ファイル(TerasolunaBootstrap.config)に設定する。
  本機能のデフォルトの設定では、IInitializer インタフェース実装クラスである SequenceInitializer クラスにより複数の Initializer 実装クラスを実行するようになっている。

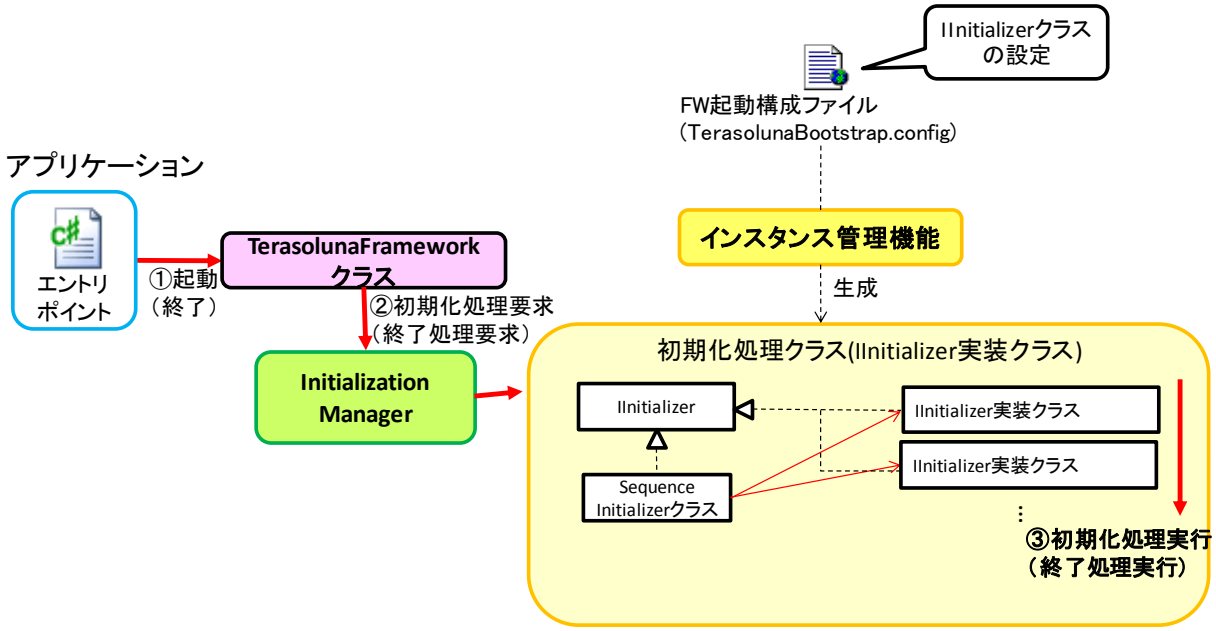

図 1フレームワークの起動・終了の動作概念図

- Windows Forms アプリケーションの FW 起動
  Windows Forms アプリケーションにおける TERASOLUNA フレームワークの起動処理は、TerasolunaStartupForm 継承クラスを呼び出すことにより実施する。
  TerasolunaStartupForm クラスは、アプリケーション起動時にフレームワークを自動的に起動するとともに、TerasolunaBootstrap.config に記述した初期画面クラスの型情報をもとにインスタンスを生成し、初期画面を起動する。
  なお、初期画面のクローズ時などアプリケーション終了時には、TERASOLUNA フレームワークの終了処理が自動的に実施される。

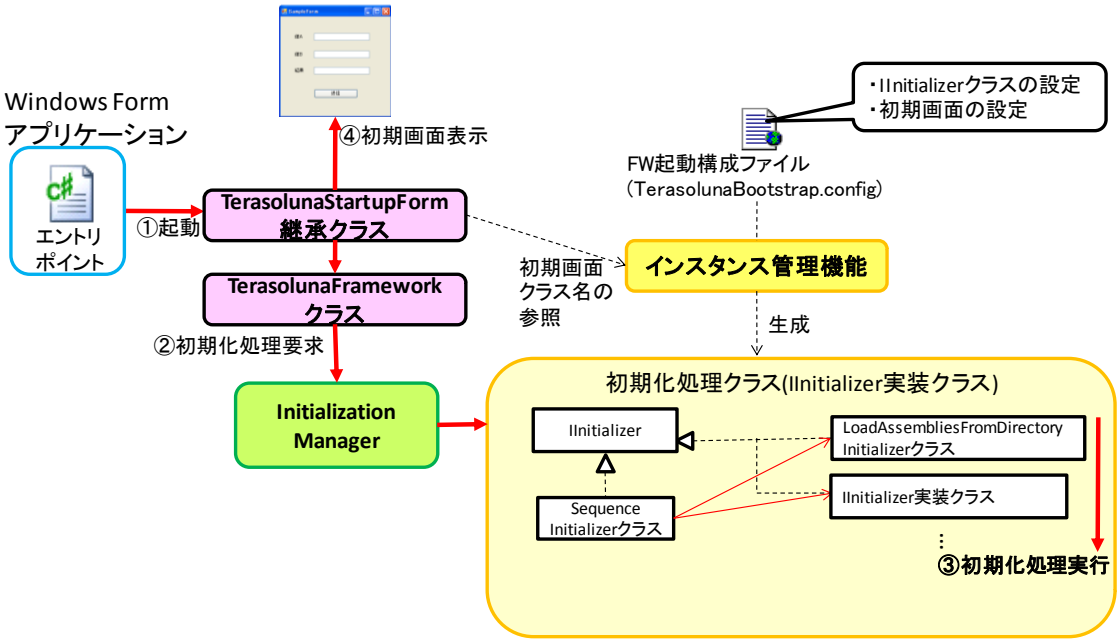

図 2 Windows Forms アプリケーションの FW 起動

- WCF サービスアプリケーションの FW 起動
  WCF サービスアプリケーションにおける TERASOLUNA フレームワークの起動処理は、「SV-01 WCF サービス管理機能」により、WCF サービスホストを作成する際に TerasolunaServicesFramework クラスを呼び出し、フレームワークを自動的に起動する。
  「SV-01 WCF サービス管理機能」の詳細については、「SV-01 WCF サービス管理機能」の機能説明書を参照のこと。

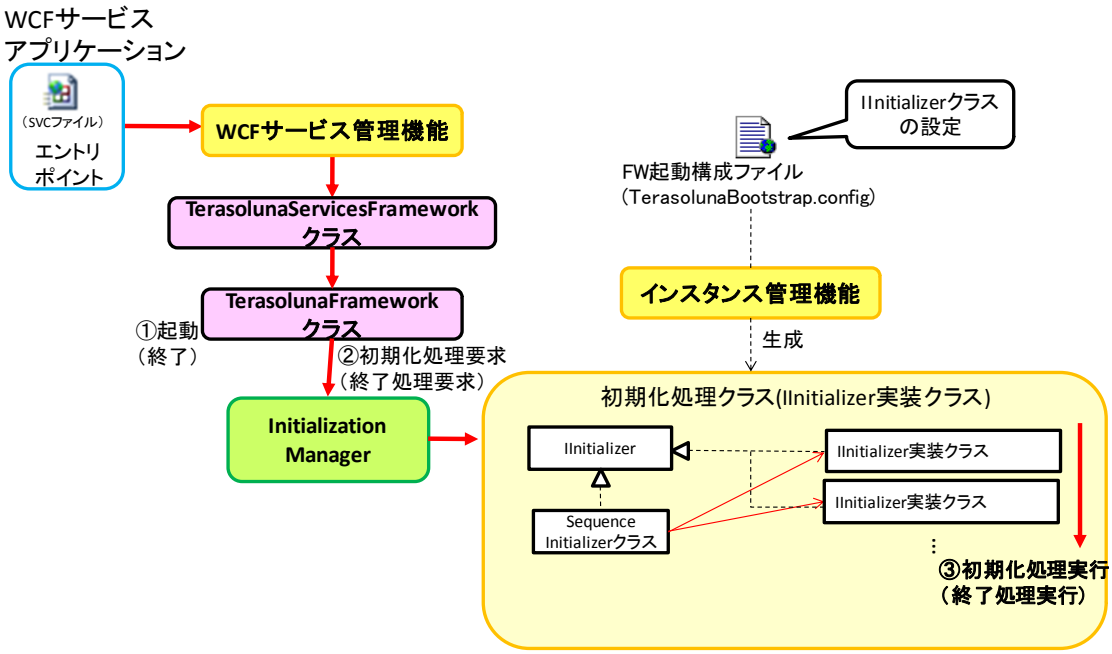

図 3　WCF サービスアプリケーションの起動

- 「デザインモード」と「実行時モード」の切り替え
  TERASOLUNA フレームワークは、通常のアプリケーション実行時に利用するだけなく、Visual Studio のデザイナの入力支援機能を提供するため、フレームワークの動作モードとして、以下の2つのモードを用意し、フレームワークがどちらのモードで起動しているかを状態管理する。

  - 実行時モード
    - TERASOLUNA フレームワークを利用したアプリケーションを実行するモード。開発したアプリケーションを実行する際に、TERASOLUNA フレームワークをロードし実行する。
  - デザインモード
    - TERASOLUNA フレームワークの各機能が実現するデザイナ拡張機能を動作させるためのモード。Visual Studio が TERASOLUNA フレームワークをロードし実行する。

■ 使用方法

◆ アプリケーションの起動・終了処理の実装

(1) コンソールアプリケーションにおけるフレームワーク起動・終了

コンソールアプリケーションなどでフレームワークの起動・終了する際は、直接 Terasoluna.TerasolunaFramework クラスの Run/Exit メソッドを実行する。

表 1 に TerasolunaFramework クラスのメソッドを示す。

Run メソッドで実行すると、フレームワークを利用可能にするための初期化処理を実施し、「実行時モード」[1]でフレームワークを起動する。Exit メソッドを実行することで、フレームワーク終了時の後処理を実施する。

表 1 TerasolunaFramework クラスのメソッド一覧

項番	メソッド	説明
1	public static void Run()	フレームワークを起動する。 フレームワークの初期化処理を実施し、「実行時モード」で開始する。
2	public static void Exit()	フレームワークを終了する。

TerasolunaFramework クラスによる フレームワークの起動、終了処理の記述例を以下に示す。

```
class Program
{
    public static void Main(string[] args)
    {
        try
        {
            ///Terasolunaフレームワークの起動
            ///フレームワークの初期化、「実行モード」で起動
            TerasolunaFramework.Run();
            ///TODO: Startup実行後、プログラムの処理を記述
            ・・・
        }
        finally
        {
            ///Terasolunaフレームワークの終了処理
            TerasolunaFramework.Exit();
        }
    }
}
```

リスト 1 TerasolunaFramework クラスによる起動、終了処理の記述例

[1]「デザインモード」での起動については開発者が特に意識することはない。「デザインモード」で起動する場合、Visual Studio がフレームワークをロードする際 Run メソッドを実行しないため、TERASOLUNA フレームワーク自身が「デザインモード」で起動していることを知っている。

(2) Windows Forms アプリケーションにおけるフレームワーク起動・終了

Windows Forms アプリケーションの場合、Terasoluna.Windows.TerasolunaStartupForm の継承クラスを使ってフレームワークを起動する。

TerasolunaStartupForm は、System.Windows.Forms.Form クラスを継承しているので、通常の Windows Forms アプリケーションと同様の実装方法でアプリケーション起動することができ、そのタイミングで、同時にフレームワークも起動する。

アプリケーション起動時の TerasolunaStartupForm 継承クラスによるフレームワークの起動処理の記述例を以下に示す。

まず、エントリポイントがある(Main メソッドやメインフォームがある)プロジェクトにおいて、TerasolunaStartupForm を継承した FW 起動用画面クラスを作成する。継承してコンストラクタを作成するのみで特別な実装は必要ない。以下に、TerasolunaStartupForm 継承クラスの記述例を示す。

```
namespace TourSampleWinFormsApp
{
    public partial class TourSampleStartupForm : TerasolunaStartupForm
    {
        public TourSampleStartupForm()
        {
            InitializeComponent();
        }
    }
}
```

リスト 2 TerasolunaStartupForm 継承クラスの記述例

C#の場合、Main メソッドで、作成した TerasolunaStartupForm 継承クラスを引数として、Application.Run メソッドを実行する。また、実際に起動する初期画面クラスは、FW 起動構成ファイル(TerasolunaBootstrap.config)に設定する。記述方法については、後述の「FW 起動構成ファイル(TerasolunaBootstrap.config)の記述」を参照のこと。

以下に、アプリケーション起動処理の記述例を示す。

```
namespace TourSampleWinFormsApp
{
    static class Program
    {
        /// <summary>
        /// アプリケーションのメイン エントリ ポイントです。
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            ///TerasolunaStartupForm継承クラスによるフレームワークの起動処理
            ///TerasolunaBootstrap.configに設定した初期表示画面を起動する
            Application.Run(new TourSampleStartupForm());
        }
    }
}
```

リスト 3 TerasolunaStartupForm 継承クラスによる AP 起動処理の記述例(C#)

VB.NET の場合は、「プロジェクト」メニューの「プロパティ」の「アプリケーション」タブで、作成した TerasolunaStartupForm 継承クラスを「スタートアップフォーム」に設定する。

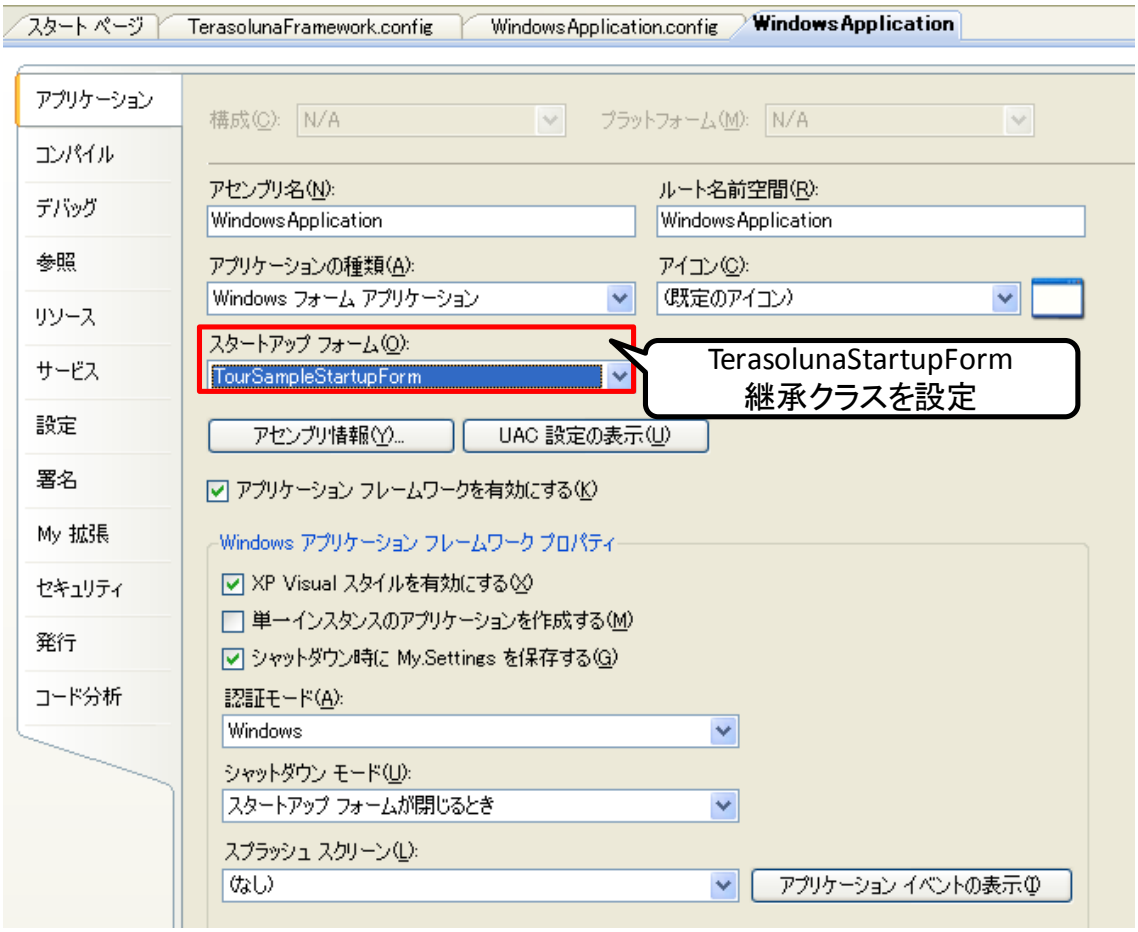

図 4 TerasolunaStartupForm 継承クラスによる AP 起動処理の設定例(VB.NET)

通常の Windows Forms アプリケーション同様に初期表示画面クラスをクローズするか Application.Exit メソッドを実行しアプリケーション終了することで、自動的にフレームワークも終了する。

(3) WCF サービスアプリケーションにおけるフレームワーク起動

WCF サービスアプリケーションの場合、TERASOLUNA が提供するカスタムアイテムテンプレートを使用して「WCF サービスアプリケーションプロジェクト」に、以下のような.svc ファイルを作成する。これにより、「SV-01 WCF サービス管理機能」が WCF サービスホストを作成する際にフレームワークを自動的に起動する。

利用方法の詳細については、「SV-01 WCF サービス管理機能」の機能説明書を参照のこと。

```
<%@ ServiceHost Language="C#" Debug="true"
        Factory="Terasoluna.Server.ServiceModel.DefaultServiceHostFactory" %>
```

リスト 4 .svc ファイルの記述例

◆ FW 起動構成ファイル(TerasolunaBootstrap.config)の記述

TERASOLUNA フレームワークを起動するには、FW 起動構成ファイル(TerasolunaBootstrap.config)を作成する必要がある。

通常、TERASOLUNA フレームワークが提供するカスタムプロジェクトテンプレートを利用することで、プロジェクト作成時に TerasolunaBootstrap.config を生成することができる。

Windows Forms アプリケーションの場合は、AP 起動用のプロジェクトテンプレート(「起動アプリケーション」プロジェクトテンプレート)を利用する。

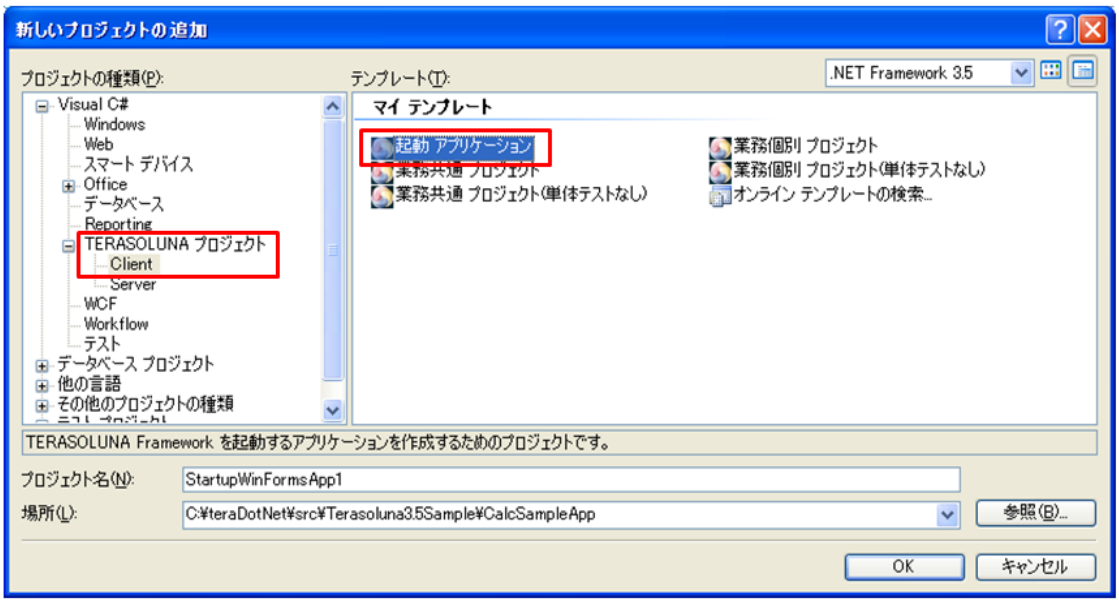

図 5「起動アプリケーション」プロジェクトテンプレート

WCF サービスアプリケーションの場合は、「WCF サービス アプリケーション」プロジェクトテンプ

レートを利用する。

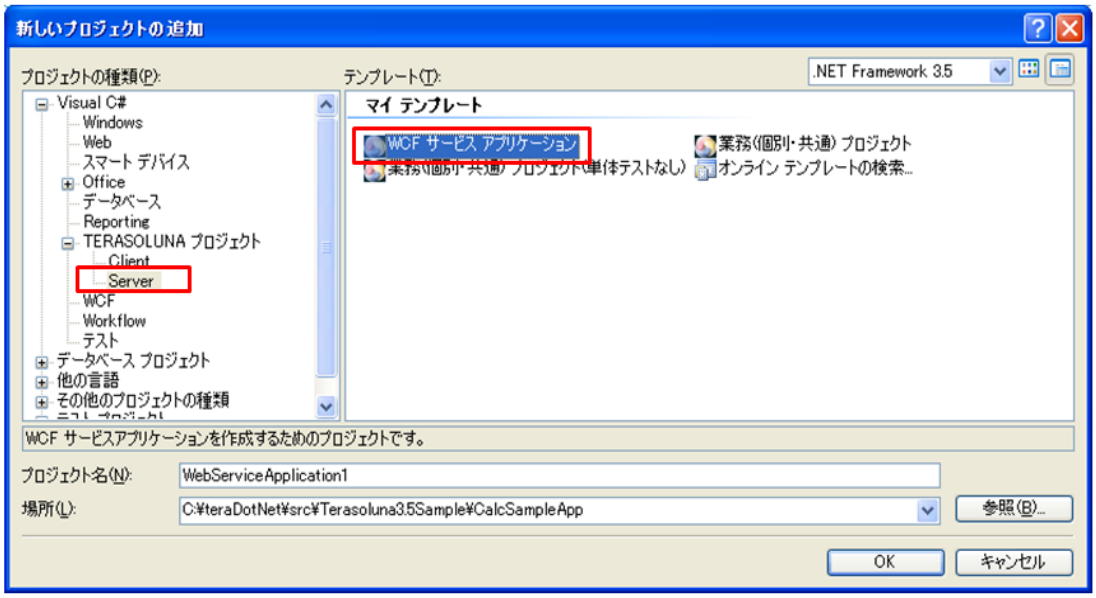

図 6「WCF サービス アプリケーション」プロジェクトテンプレート

また、作成済みのプロジェクトに FW 起動構成ファイル(TerasolunaBootstrap.config)を追加したい場合には、Terasoluna が提供するカスタムアイテムテンプレートである「FW 起動構成ファイル」テンプレートにより作成することもできる。

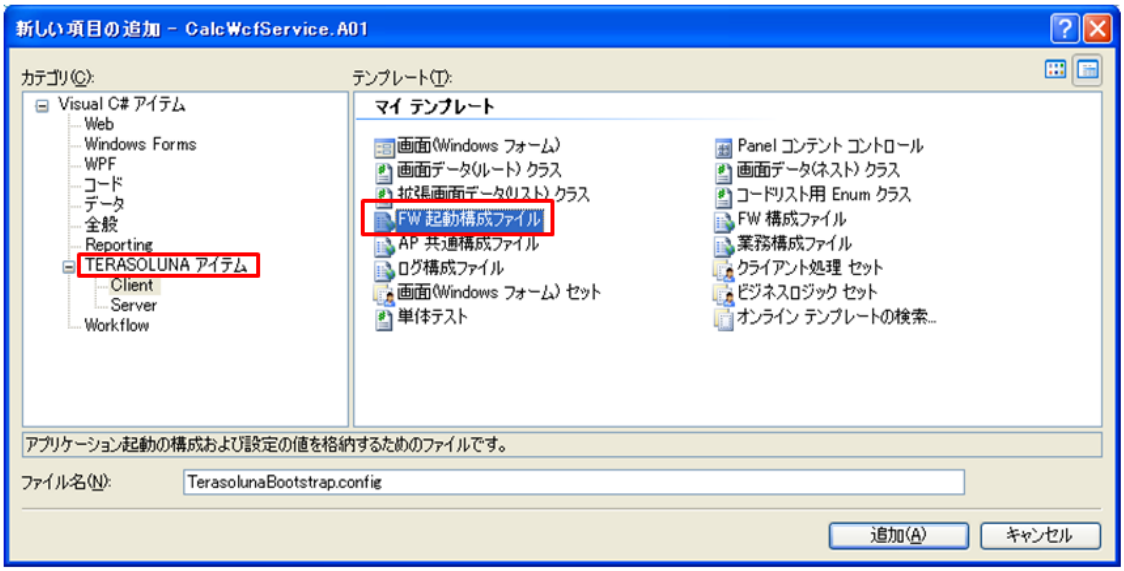

図 7 「FW 起動構成ファイル」テンプレート

TerasolunaBootstrap.config は、実行時に、実行ファイルと同じフォルダに配置しておく必要がある。Visual Studio での開発時、アーキテクトは、TerasolunaBootstrap.config を作成し、あらかじめエントリポイントがあるプロジェクトに配置しておく必要がある。また、プロパティエディタで「出力ディレクトリにコピー」を「新しい場合はコピーする」または「常にコピーする」に設定し、ビルド時に当該ファイルが実行フォルダに出力されるようにしておく必要がある。前述した「起動アプリケーション」プロジェクトテンプレートからプロジェクトを生成した場合、これらは最初から設定されている。

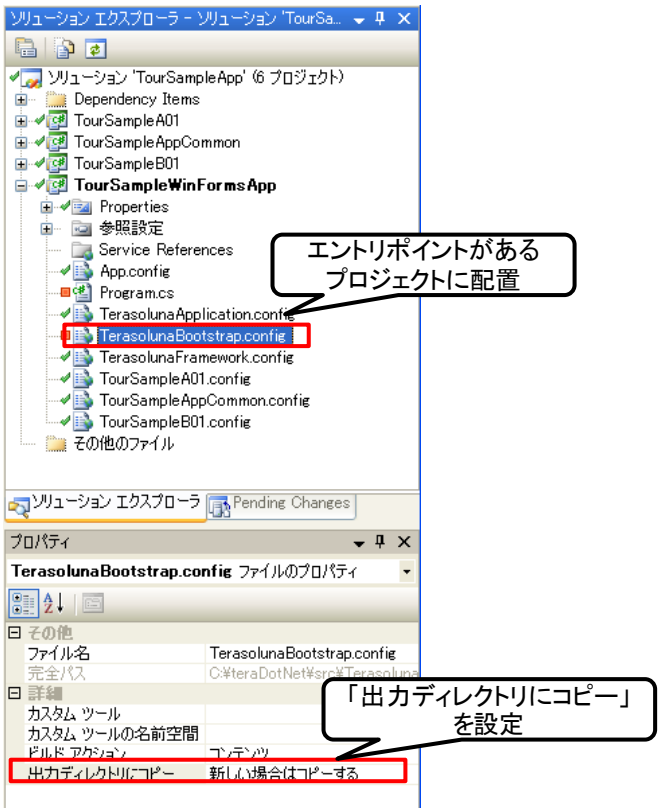

図 8 TerasolunaStartup.config の配置

(1) 初期化・終了処理実施クラス(IInitializer 実装クラス)の DI 設定

本機能は、「CM-02 インスタンス管理機能」を利用しており、TerasolunaBootstrap.config に書かれた Unity AB の設定をロードして、IInitializer インタフェース実装クラスのインスタンスを生成する。

IInitilizer クラスのインスタンスは、configuration/unity/containers/container/types/type タグで設定する。

IInitializer 実装クラスとして、標準で表 2 に示すクラスを提供している。

表 2 IInitializer インタフェース実装クラス

項番	クラス名	説明
1	SequenceInitializer	Initializers プロパティ(IInitializer 配列)に格納されている IInitializer 実装クラスを順番に実行し初期化(または終了)処理を実施する。

項番	クラス名	説明
2	LoadAssembliesFromDirectoryInitializer	アプリケーション開始時に、実行ファイルが存在するディレクトリにあるアセンブリをロードするInitializer。 このInitializerにより、「CL-02 画面遷移機能」において別アセンブリに存在する画面クラスのカスタム属性の読み取りを実現するといった、複数アセンブリに分割されたアプリケーションに対応したフレームワーク機能を動作可能にする。
3	EmptyInitializer	何もしない、空のInitializer。 特に初期化処理を必要としない場合に設定する。

本機能は、IInitializer インタフェースのデフォルト(name 属性を指定しない定義)の DI 定義を参照して初期化(終了)処理を実施する。IInitailizer インタフェースのデフォルトの DI 設定として標準では、SequenceInitializer クラスを設定しておく。
SequenceInitializer クラスは自身も IInitializer インタフェースを実装しておりアーキテクトは、実行したい IInitializer 実装クラスを Initializers プロパティに DI 設定する。

【Windows Forms アプリケーションの場合の例】TERASOLUNA フレームワークのデフォルトの画面遷移機能では、複数のアセンブリから ScreenId 属性に合致する画面クラスを検索する必要があるため、SequenceInitializer クラスの Initializers プロパティに LoadAssembliesFromDirectoryInitializer を DI 設定しておく必要がある。
以下に、Windows Forms アプリケーションの場合の IInitializer 実装クラスの DI 設定の例を示す。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <unity>
    <typeAliases>
      <!-- エイリアスの定義 -->
      ・・・
      <typeAlias alias="IInitializer[]" type="Terasoluna.Initialization.IInitializer[],
                        Terasoluna" />
      <typeAlias alias="IInitializer" type="Terasoluna.Initialization.IInitializer,
                        Terasoluna" />
      <typeAlias alias="SequenceInitializer" type="Terasoluna.Initialization.SequenceInitializer,
                        Terasoluna" />
    <containers>
      <container>
        <types>
          <!-- フレームワーク起動時に実行される初期化処理実施(Intializer)クラスの設定-->
          <!-- IInitilizerインタフェースのデフォルトDI定義
                SequenceInitializerクラスが、Intializersに登録したIIntializerクラスに呼び出す -->
          <type type="IInitializer" mapTo="SequenceInitializer">
            <lifetime type="singleton" />
            <typeConfig>
              <property name="Initializers" propertyType ="IInitializer[]">
                <array>
                  <dependency name=" LoadAssembliesFromDirectoryInitializer "/>
                </array>
              </property>
            </typeConfig>
          </type>
          <!-- Windows Formsアプリケーションの場合には
                LoadAssembliesFromDirectoryInitializer の実行が必要-->
          <type type="IInitializer" name="LoadAssembliesFromDirectoryInitializer"
                mapTo="Terasoluna.Windows.Initializers.LoadAssembliesFromDirectoryInitializer,
                        Terasoluna.Windows" >
            <lifetime type="singleton" />
          </type>
          ・・・
        </container>
      </containers>
  </unity>
</configuration>
```

リスト 5 Window Form アプリケーションの IInitializer の設定例(TerasolunaBootstrap.config)

【WCF サービスアプリケーションの場合の例】
標準では特に実行しなければならない初期化処理は存在しない。このような場合には、
EmptyInitializer クラスを DI 設定しておく。
以下に、WCF サービスアプリケーションの場合の IInitializer 実装クラスの DI 設定の例を示す。

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <unity>
    <typeAliases>
      <!-- エイリアスの定義 -->
      ・・・
      <typeAlias alias="IInitializer[]" type="Terasoluna.Initialization.IInitializer[],
                        Terasoluna" />
      <typeAlias alias="IInitializer" type="Terasoluna.Initialization.IInitializer,
                        Terasoluna" />
      <typeAlias alias="SequenceInitializer" type="Terasoluna.Initialization.SequenceInitializer,
                        Terasoluna" />
    <containers>
      <container>
        <types>
          <!-- フレームワーク起動時に実行される初期化処理実施(Intializer)クラスの設定-->
          <!-- IInitilizerインタフェースのデフォルトDI定義
                SequenceInitializerクラスが、Intializersに登録したIIntializerクラスに呼び出す  -->
          <type type="IInitializer" mapTo="SequenceInitializer">
            <lifetime type="singleton" />
            <typeConfig>
              <property name="Initializers" propertyType ="IInitializer[]">
                <array>
                  <dependency name=" EmptyInitializer "/>
                </array>
              </property>
            </typeConfig>
          </type>
          <!-- 特に初期化処理がない場合はEmptyInitializer を設定-->
          <type type="IInitializer" name="EmptyInitializer"
                mapTo="Terasoluna.Initialization.EmptyInitializer, Terasoluna" >
            <lifetime type="singleton" />
          </type>
          ・・・
      </container>
    </containers>
  </unity>
</configuration>
```

リスト 6　WCF サービスアプリケーションの IInitializer の設定例(TerasolunaBootstrap.config)

(2) Windows Forms アプリケーションの初期表示画面クラスの設定

Windows Forms アプリケーションの場合に必要な設定である。
TerasolunaStartupForm 継承クラスが、「CM-02 インスタンス管理機能」を利用して TerasolunaBootstrap.config に記述した初期表示画面クラスの型オブジェクト(System.Type)を取得し、アプリケーションの初期画面を表示する。
初期表示画面の設定は、configuration/unity/containers/container/instances タグに、name 属性を「StartFormType」として、初期画面クラスの Type インスタンスを設定する。value 属性には、初期表示画面クラスの完全修飾型名を記述する。
また typeConverter 属性には、UnityAB の Microsoft.Practices.Unity.Configuration.AssemblyQualifiedTypeNameConverter クラスを設定すること。AssemblyQualifiedTypeNameConverter クラスは、value に設定した完全修飾型名の文字列をもとに Type インスタンスに変換してくれる。

以下に、初期画面の設定例を示す。

```
・・・
  <unity>
    <typeAliases>
      <!-- エイリアスの定義 -->
      <typeAlias alias="Type" type="System.Type" />
・・・
      <!-- AssemblyQualifiedTypeNameConverterの定義 -->
      <typeAlias alias="TypeNameConverter"
                 type="Microsoft.Practices.Unity.Configuration.AssemblyQualifiedTypeNameConverter,
                       Microsoft.Practices.Unity.Configuration,
                       Version=1.2.0.0,
                       Culture=neutral,
                       PublicKeyToken=31bf3856ad364e35" />
    </typeAliases>
    <containers>
      <container>
・・・
        <!-- 起動画面の設定 -->
        <instances>
          <add name="StartFormType" type="Type"
               value="TourSampleB01.B01_01.View.SC_B01_01_01View, TourSampleB01"
               typeConverter=" TypeNameConverter "/>
        </instances>
        <extensions>
        </extensions>
      </container>
    </containers>
  </unity>
</configuration>
```

リスト 7 初期画面の設定例(TerasolunaBootstrap.config の記述抜粋)

(3) Windows Forms アプリケーションプロジェクトの参照設定

Windows Forms アプリケーションの場合に必要な設定である。

本機能が提供する初期画面起動機能や「CL-02 画面遷移機能」の仕組みを利用することで、TERASOLUNA フレームワークを使った Windows Forms アプリケーションは、画面クラスの呼び出し関係が完全に疎結合になっている。このため、ビルド時に必要な最小限の参照設定では、エントリポイントのプロジェクトには初期画面クラスが存在する業務個別プロジェクト(アセンブリ)の参照設定は不要となる。また、各業務個別プロジェクトにおいても、業務共通プロジェクト以外の参照はが不要であるため、エントリポイントのプロジェクトに実行時に必要な業務プロジェクトのアセンブリ(DLL)の大半が生成されない。

そのため、エントリポイントのプロジェクトに、実行時に必要なアセンブリを出力させて、LoadAssembliesFromDirectoryInitializer を利用してロードさせるためには、エントリポイントのあるプロジェクトの「参照設定」で、全ての業務のアセンブリ(またはプロジェクト)を追加する必要がある。

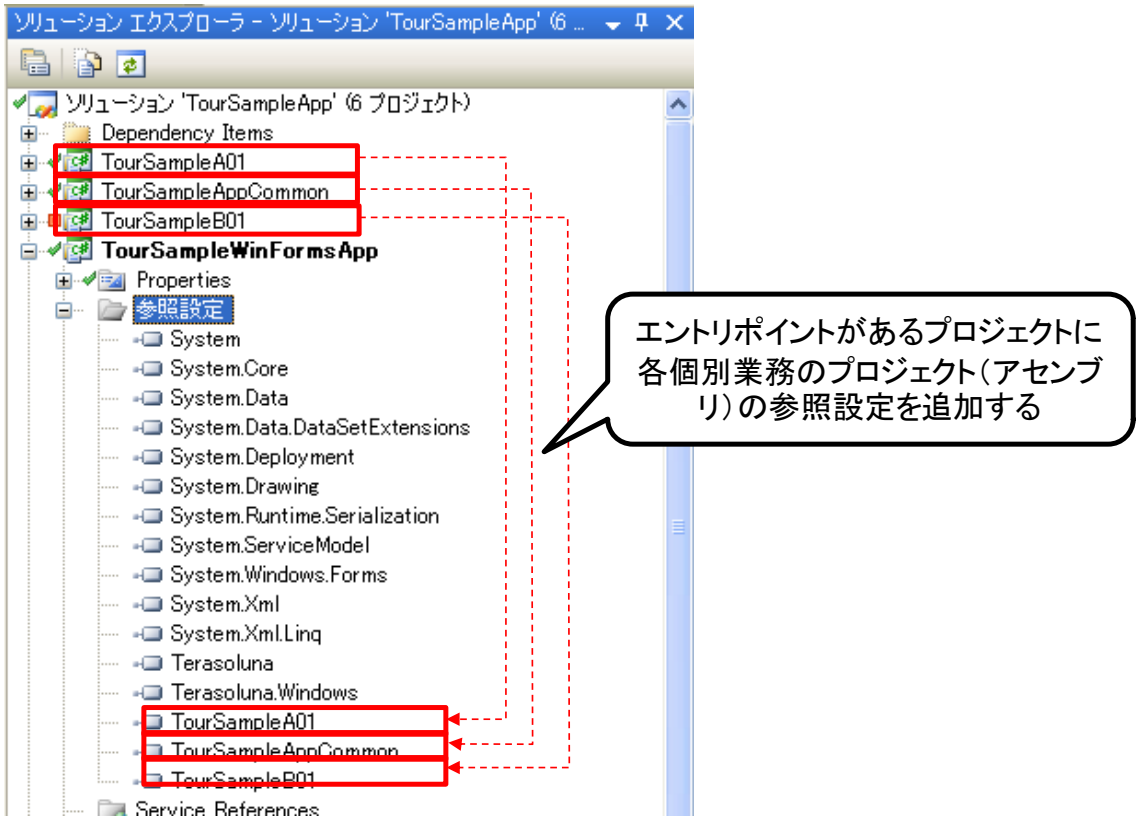

図 9 プロジェクト(アセンブリ)の参照設定

■ 内部構成

◆ 構成クラス

本機能を構成するクラスを以下に示す。

表 3 構成クラス一覧

項番	クラス名	説明
Terasoluna 名前空間		
1	TerasolunaFramework	フレームワークの起動・終了クラス。
Terasoluna.Initialization 名前空間		
2	InitializationManager	初期化・終了処理管理を管理するクラス。
3	IInitializer	初期化・終了処理が実装するインタフェース
4	InitializerBase	IInitializer 実装クラスの基底クラス。
5	EmptyInitializer	IInitializer 実装クラスの空実装。
6	SequenceInitializer	内部に保持した、IInitializer 実装クラスのリストを順番に実行する IInitializer 実装クラス。
7	InitializeException	初期化処理でエラーが発生したことを通知する例外クラス
Terasoluna.Windows 名前空間		
8	TerasolunaStartupForm	Windows Forms アプリケーションにおけるフレームワーク起動の基底クラス。
Terasoluna.Windows.Initializers 名前空間		
9	LoadAssembliesFromDirectoryInitializer	アプリケーション開始時に、実行アプリケーションの各アセンブリをロードする Initializer。
Terasoluna.Server 名前空間		
10	TerasolunaServicesFramework	WCF サービスにおけるフレームワーク起動・終了クラス。

■ 拡張ポイント

フレームワーク起動・終了時に、プロジェクト独自の初期化・終了処理を追加したい場合は、IInitializer インタフェースを実装し、TerasolunaBootstrap.config で、SequenceInitializer クラスの Initializers プロパティに追加する。

```
・・・
    <containers>
      <container>
        <types>
          <!-- フレームワーク起動時に実行される初期化処理実施(Intializer)クラスの設定-->
          <type type="IInitializer" mapTo="SequenceInitializer">
            <lifetime type="singleton" />
            <typeConfig>
              <property name="Initializers" propertyType ="IInitializer[]">
                <array>
                  <dependency name="LoadAssembliesFromDirectoryInitializer"/>
                  <!-- IIntializer実装クラスの追加設定-->
                  <dependency name="SampleInitializer"/>
                </array>
              </property>
            </typeConfig>
          </type>
・・・
          <!--作成したIIntializer実装クラスの定義->
          <type type="IInitializer" name="SampleInitializer"
                mapTo="Terasoluna.Sample.SampleInitializer,   Terasoluna.Sample" >
            <lifetime type="singleton" />
          </type>
        </types>
・・・
      </container>
    </containers>
  </unity>
</configuration>
```

リスト 8 TerasolunaBootstrap.config の記述例

■ 関連機能

- CM-02 インスタンス管理機能
- CL-02 画面遷移機能
- SV-01 WCF サービス管理機能