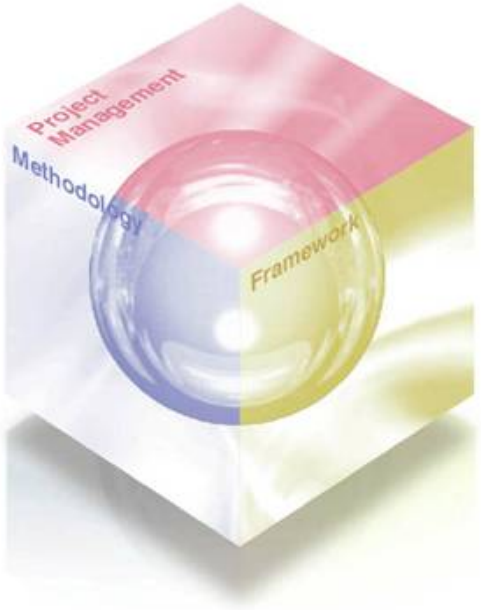
TERASOLUNA® Server Framework for .NET 2.1.0.1
アーキテクチャ説明書（Web版）



株式会社NTTデータ

TERASOLUNA®
BEST SOLUTION FOR SYSTEM DEVELOPMENT

NTT DATA

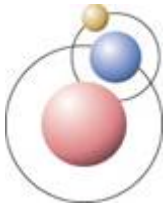
本ドキュメントを使用するにあたり、以下の規約に同意していただく必要があります。同意いただけない場合は、本ドキュメント及びその複製物の全てを直ちに消去又は破棄してください。

1. 本ドキュメントの著作権及びその他一切の権利は、NTTデータあるいはNTTデータに権利を許諾する第三者に帰属します。
2. 本ドキュメントの一部または全部を、自らが使用する目的において、複製、翻訳、翻案することができます。ただし本ページの規約全文、およびNTTデータの著作権表示を削除することはできません。
3. 本ドキュメントの一部または全部を、自らが使用する目的において改変したり、本ドキュメントを用いた二次的著作物を作成することができます。ただし、「参考文献：TERASOLUNA Server Framework for .NET 2.1 アーキテクチャ説明書 （Web版）」あるいは同等の表現を、作成したドキュメント及びその複製物に記載するものとします。
4. 前2項によって作成したドキュメント及びその複製物を、無償の場合に限り、第三者へ提供することができます。
5. NTTデータの書面による承諾を得ることなく、本規約に定められる条件を超えて、本ドキュメント及びその複製物を使用したり、本規約上の権利の全部又は一部を第三者に譲渡したりすることはできません。
6. NTTデータは、本ドキュメントの内容の正確性、使用目的への適合性の保証、使用結果についての的確性や信頼性の保証、及び瑕疵担保義務も含め、直接、間接に被ったいかなる損害に対しても一切の責任を負いません。
7. NTTデータは、本ドキュメントが第三者の著作権、その他如何なる権利も侵害しないことを保証しません。また、著作権、その他の権利侵害を直接又は間接の原因としてなされる如何なる請求（第三者との間の紛争を理由になされる請求を含む。）に関しても、NTTデータは一切の責任を負いません。

本ドキュメントで使用されている各社の会社名及びサービス名、商品名に関する登録商標および商標は、以下の通りです。

- Microsoft、Visual Stuido、Windows、.NET Frameworkは、米国Microsoft Corp.の米国及びその他の国における登録商標または商標です。
- TERASOLUNAは、株式会社NTTデータの登録商標です。
- その他の会社名、製品名は、各社の登録商標または商標です。

目次

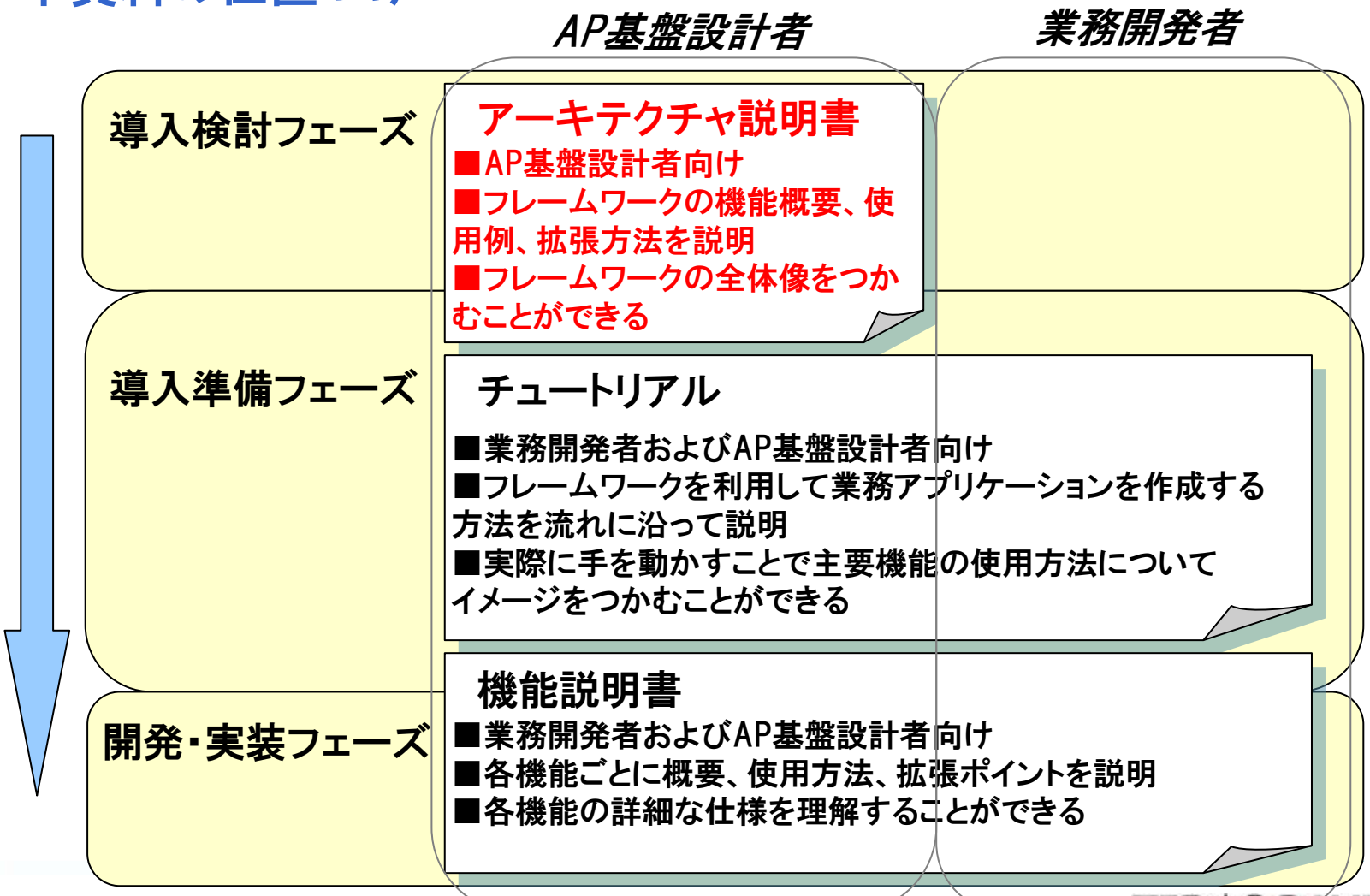


はじめに(1/2)

本資料は、「TERASOLUNA Server Framework for .NET 2.1 (Web版)」の機能について解説した資料である。

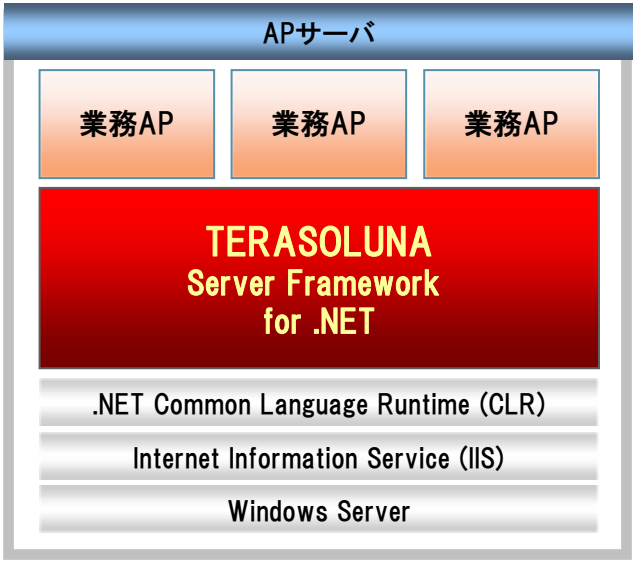
はじめに(2/2)

本資料の位置づけ



動作環境

- 動作確認環境
  - Webサーバ
    - Windows Server 2003 R2 SP2 上の IIS6.0
  - .NET Framework
    - .NET Framework 2.0
    - .NET Framework 2.0 SP1

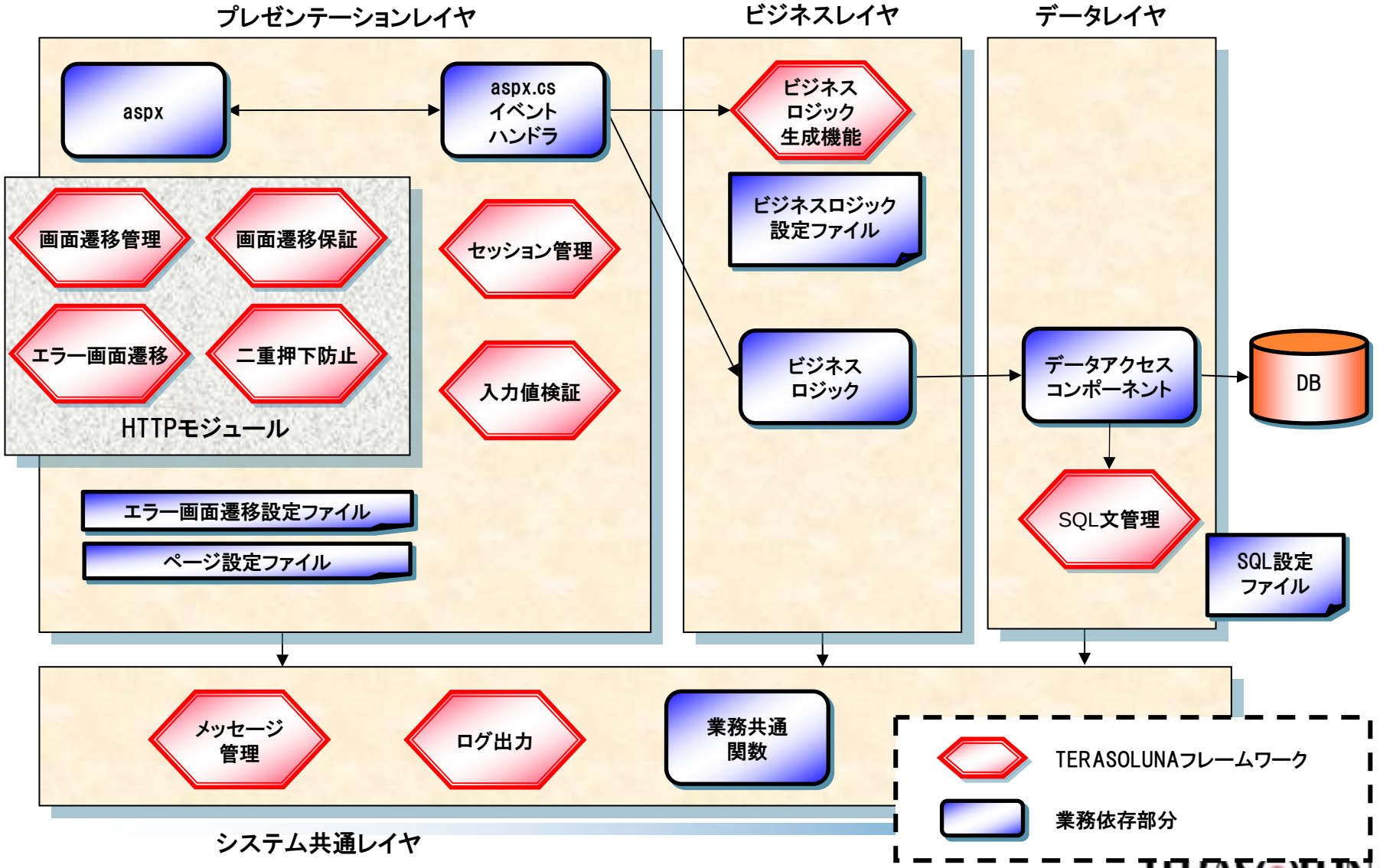


.NET Framework 3.0、3.5では動作確認を行っていない。

ただし、マイクロソフト社から .NET Framework 3.0 は .NET Framework 2.0 との、

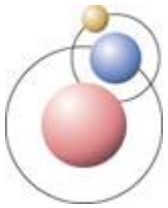
.NET Framework 3.5 は .NET Framework 2.0 SP1 との互換性が保証されている。

Web版アーキテクチャ概観（1/3）



Web版アーキテクチャ概観（2/3）

- 各レイヤの役割
  - プレゼンテーションレイヤ
    - クライアントとのやり取りを行うユーザインタフェース(UI)を提供
    - 入力値検証など直接業務ルールとは関係のない処理を行う
  - ビジネスレイヤ
    - 業務ロジックに関わる処理を行う
      - 業務ロジックとは、「特定の顧客に一定の信用限度を承認するかどうかを判断する」といったビジネスルール
  - データレイヤ
    - データベースに格納されているデータをビジネスレイヤに公開する処理を行う



Web版アーキテクチャ概観（3/3）

- レイヤ分割の意義
  - 保守性の向上、テスト容易性の実現
    - コンポーネントごとに役割を持たせて、機能を実現することで変更箇所の特定を容易にする
    - コンポーネント個別のテストができるように実装する
      - デバッグやバグ修正に必要な作業を軽減
  - 再利用性の向上
    - 異なるクライアントへの対応
      - ビジネスレイヤ、データレイヤは共通化、プレゼンテーションレイヤのみクライアント（ブラウザ、リッチクライアント）に合わせて開発
  - 開発の分業化
    - コンポーネントごとに個別に開発が可能
      - レイヤ間のインターフェイスを事前に定義することで開発の分業が可能

TERASOLUNA

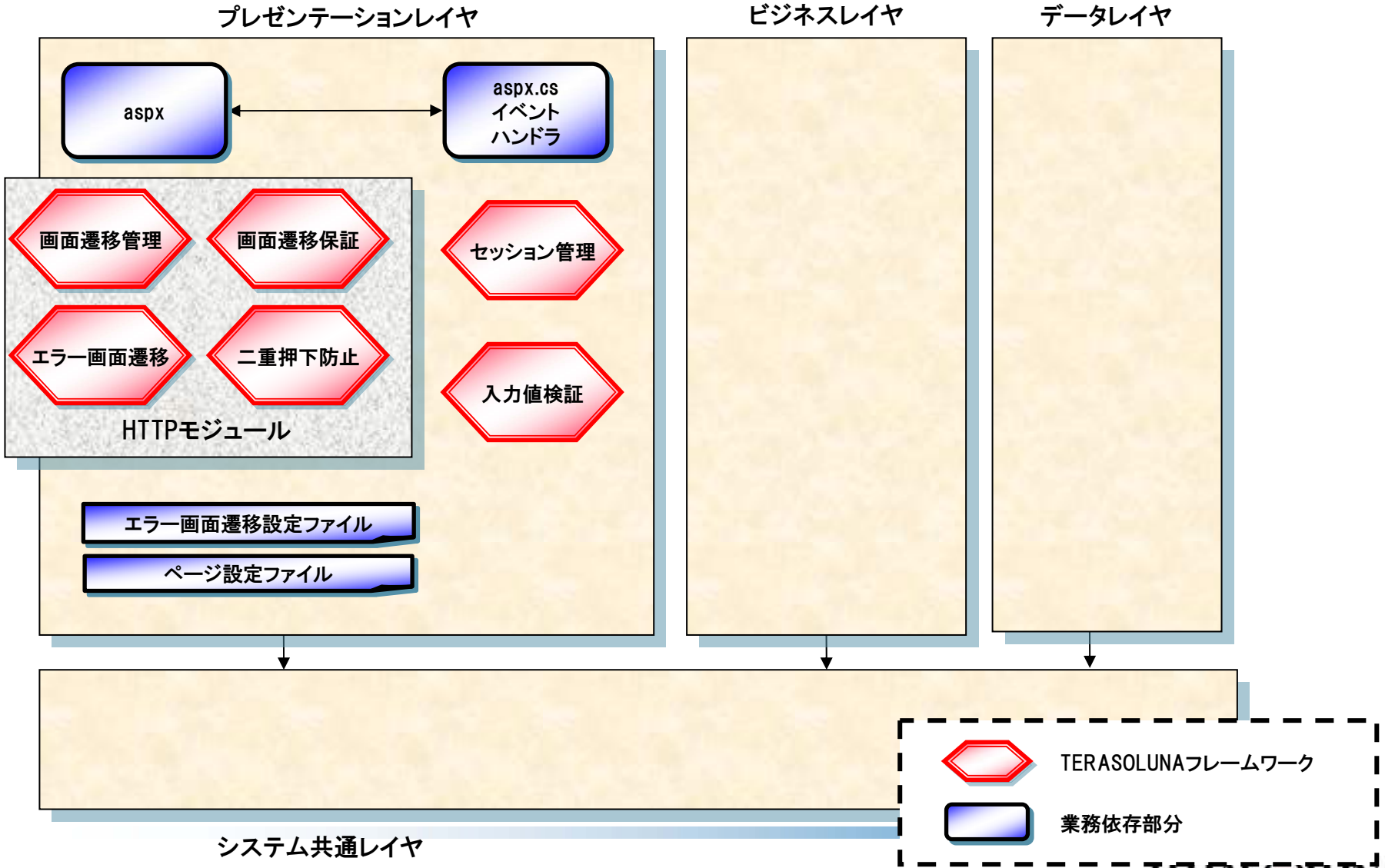
機能概要(1/2)

- プレゼンテーションレイヤ
  - WA-01 画面遷移管理機能
    - ページ設定ファイルに定義したページ情報に従って、画面遷移する機能を提供する
  - WA-02 画面遷移保証機能
    - トークンを使い、URLの直接入力などに伴う不正なアクセスを防ぐ機能を提供する
  - WA-03 二重押下防止機能
    - Submit ボタンが押下された際に画面内の全ての Submit ボタンを無効(disable) にして、リクエストの二重送信を防止する機能を提供する
  - WA-04 エラー画面遷移機能
    - アプリケーション内で未処理の例外発生時に、例外の型に応じた任意のエラー画面に遷移する機能を提供する
  - WC-01 セッション管理機能
    - セッション情報をライフサイクルレベルで一括管理する機能を提供する
  - CM-02 入力値検証機能
    - 様々な入力値検証ルールを提供する

機能概要(2/2)

- ビジネスレイヤ
  - CM-04 ビジネスロジック生成機能
    - ビジネスロジッククラスのインスタンスを生成する機能を提供する
- データレイヤ
  - WC-02 SQL文管理機能
    - 設定ファイルに定義したSQL文を取得する機能を提供する
- システム共通レイヤ
  - CM-01 メッセージ管理機能
    - アプリケーションで扱うメッセージに対し、統一的にアクセスする仕組みを提供する
  - CM-03 ログ出力機能
    - アプリケーションで統一的にログを出力する仕組みを提供する

プレゼンテーションレイヤ　機能説明



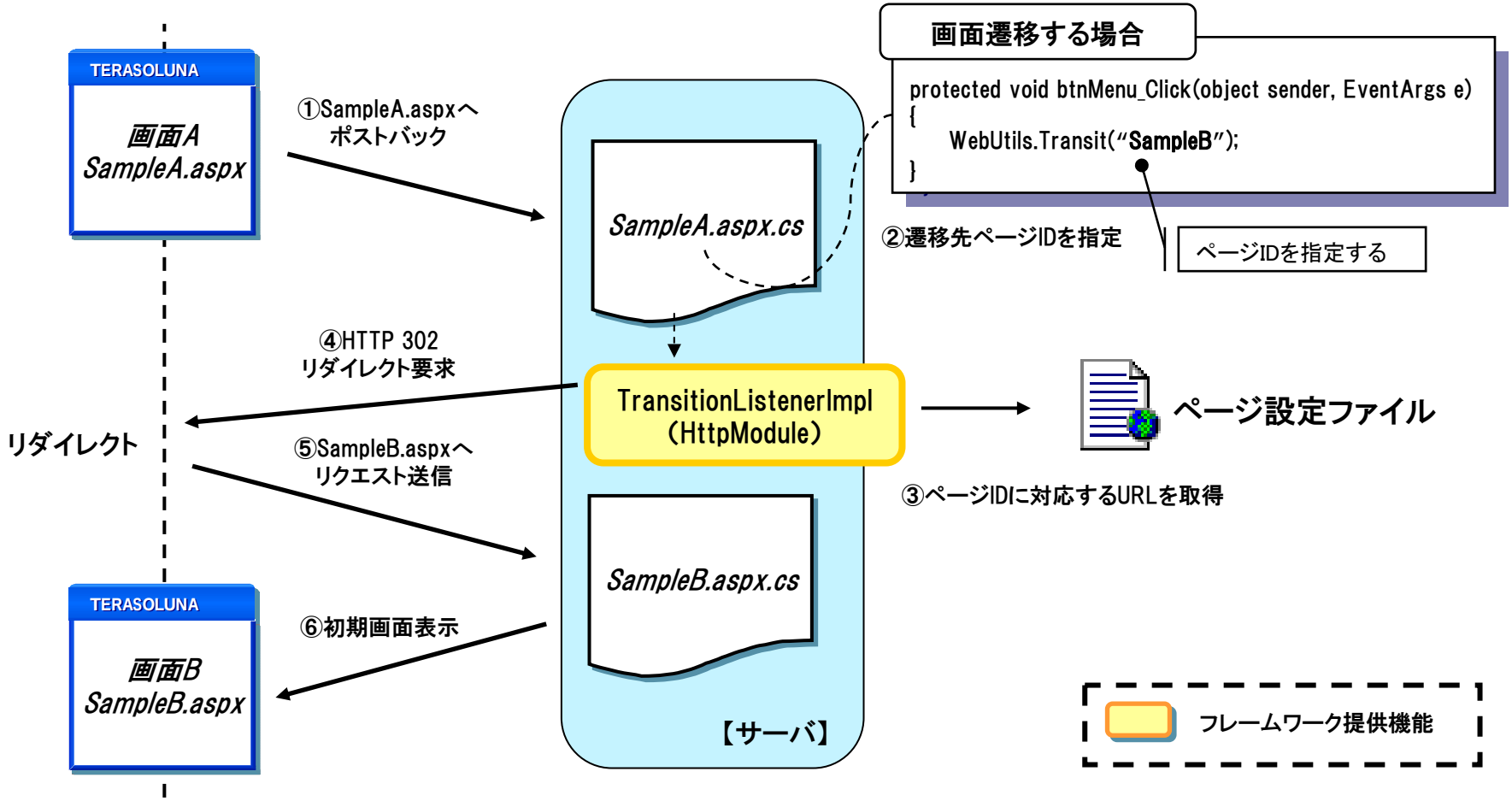
WA-01 画面遷移管理機能 – 概要

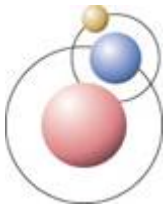
機能概要

- ページ設定ファイルに定義したページ情報に従って画面遷移する機能を提供する
  - ページ設定ファイルに定義したページIDに対応する画面へリダイレクトする
    - 通常、ASP.NETで画面遷移をする場合、リダイレクトを利用する。このときに遷移先画面のURLを直接指定することになるため、遷移元画面と遷移先画面の依存関係が強くなってしまう。そこで、実際の遷移先URLは設定ファイルに記述し、遷移先画面をページIDで指定することで、画面間の依存関係を疎にしている。

WA-01 画面遷移管理機能 – 動作イメージ

画面遷移管理機能の内部動作





WA-01 画面遷移管理機能 – 使用方法(1/3)

Web構成ファイルの設定(1)

- HTTPモジュールとしてTransitionListenerImplを設定する

```
<!--HTTPモジュールの設定-->
<system.web>
  <httpModules>
    <add name="TransitionListenerImpl"
         type="TERASOLUNA.Fw.Web.HttpModule.TransitionListenerImpl, TERASOLUNA.Fw.Web" />
  </httpModules>
</system.web>
```

WA-01 画面遷移管理機能 – 使用方法(2/3)

Web構成ファイルの設定(2)

利用するページ設定ファイルのパスを設定する

```
<!--ページ設定ファイルのパスの設定-->
<configSections>
  <section name="pageConfiguration"
           type="TERASOLUNA.Fw.Web.Configuration.Page.PageConfigurationSection, TERASOLUNA.Fw.Web"/>
</configSections>
<pageConfiguration>
  <files>
    <file path="Config¥PageConfiguration.config" />
  </files>
</pageConfiguration>
```

アプリケーションルートからの相対パスを指定する

WA-01 画面遷移管理機能 – 使用方法(3/3)

ページ設定ファイルの設定

- ページIDと遷移先画面のパスを設定する

```
<!--ページ設定ファイルの設定-->
<?xml version="1.0" encoding="utf-8" ?>
<pageConfiguration xmlns="http://www.terasoluna.jp/schema/PageSchema.xsd">
  <page name="SampleA" path="/Page/SampleA.aspx" />
  <page name="SampleB" path="/Page/SampleB.aspx" />
</pageConfiguration>
```

画面遷移の実行

- WebUtils.Transitメソッドを用いて画面遷移を実行する

画面遷移のコード例

```
protected void Button1_Click(object sender, EventArgs e)
{
   WebUtils.Transit("SampleB");
}
```

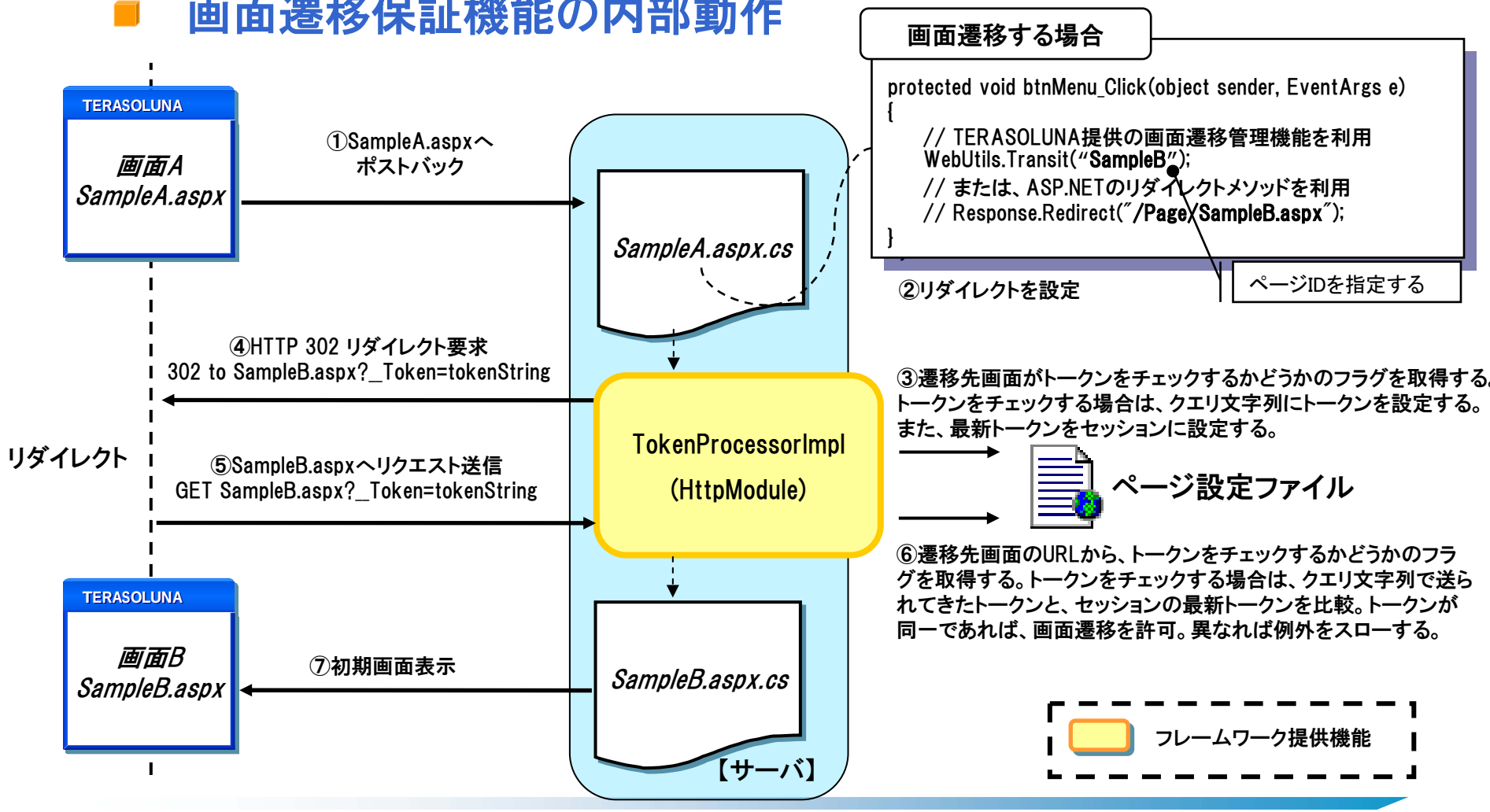
WA-02 画面遷移保証機能 – 概要

機能概要

- トークンを使い、URLの直接入力などに伴う不正なアクセスを防ぐ機能を提供する
  - トークンの確認が有効に設定された画面へ遷移するときに、遷移元画面からのリダイレクト時にトークンの設定を行う
  - リクエスト受信時、トークンの確認が有効に設定された画面でトークンが確認できない場合は、例外をスローする
    - URLを直接指定してアクセスされた時など
  - 画面遷移保証機能に関する設定は、ページ設定ファイルに記述する
    - checkToken属性
      - 画面単位で画面遷移保証機能を有効、無効にできる
    - updateToken属性
      - ブラウザの「戻る」ボタン押下後に表示されたページから、再度トークン確認が必要なページへの遷移を有効、無効にできる

WA-02 画面遷移保証機能 – 動作イメージ

画面遷移保証機能の内部動作

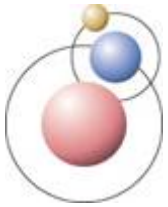


WA-02 画面遷移保証機能 – 使用方法(1/7)

Web構成ファイルの設定(1)

- HTTPモジュールとしてTokenProcessorImplを設定する

```
<!--HTTPモジュールの設定-->
<system.web>
  <httpModules>
    <add name="TokenProcessorImpl"
         type="TERASOLUNA.Fw.Web.HttpModule.TokenProcessorImpl, TERASOLUNA.Fw.Web" />
  </httpModules>
</system.web>
```



WA-02 画面遷移保証機能 – 使用方法(2/7)

Web構成ファイルの設定(2)

利用するページ設定ファイルのパスを設定する

```
<!--ページ設定ファイルのパスの設定-->
<configSections>
  <section name="pageConfiguration"
           type="TERASOLUNA.Fw.Web.Configuration.Page.PageConfigurationSection, TERASOLUNA.Fw.Web"/>
</configSections>
<pageConfiguration>
  <files>
    <file path="Config¥PageConfiguration.config" />
  </files>
</pageConfiguration>
```

WA-02 画面遷移保証機能 – 使用方法(3/7)

ページ設定ファイルの設定

- ページID、遷移先画面のパス、トークンチェックフラグ、トークン更新フラグを設定する

```
<!--ページ設定ファイルの設定-->
<?xml version="1.0" encoding="utf-8" ?>
<pageConfiguration xmlns="http://www.terasoluna.jp/schema/PageSchema.xsd">
  <page name="SampleA" path="/Page/SampleA.aspx" checkToken="on" updateToken="off" />
  <page name="SampleB" path="/Page/SampleB.aspx" checkToken="on" updateToken="on" />
  <page name="SampleC" path="/Page/SampleC.aspx" checkToken="off" />
</pageConfiguration>
```

“on” または ”off” を設定する。
指定しない場合は、既定値
“on” となる。

画面遷移の実行

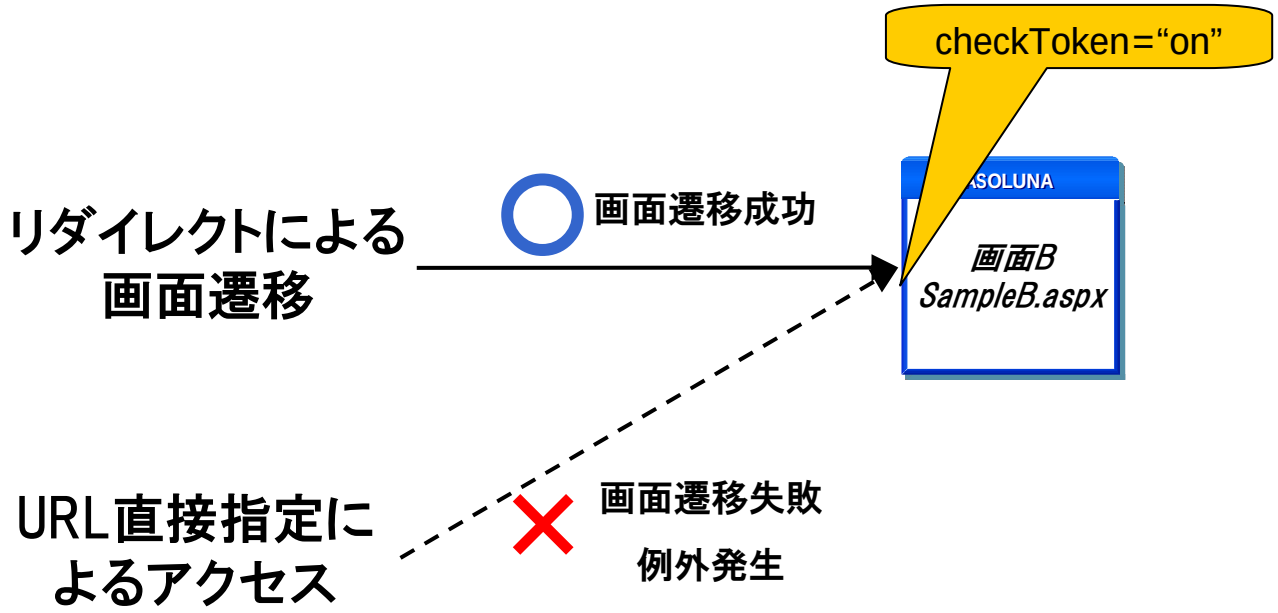
- リダイレクトを実行する

画面遷移のコード例

```
protected void Button1_Click(object sender, EventArgs e)
{
    WebUtils.Transit("SampleB");
}
```

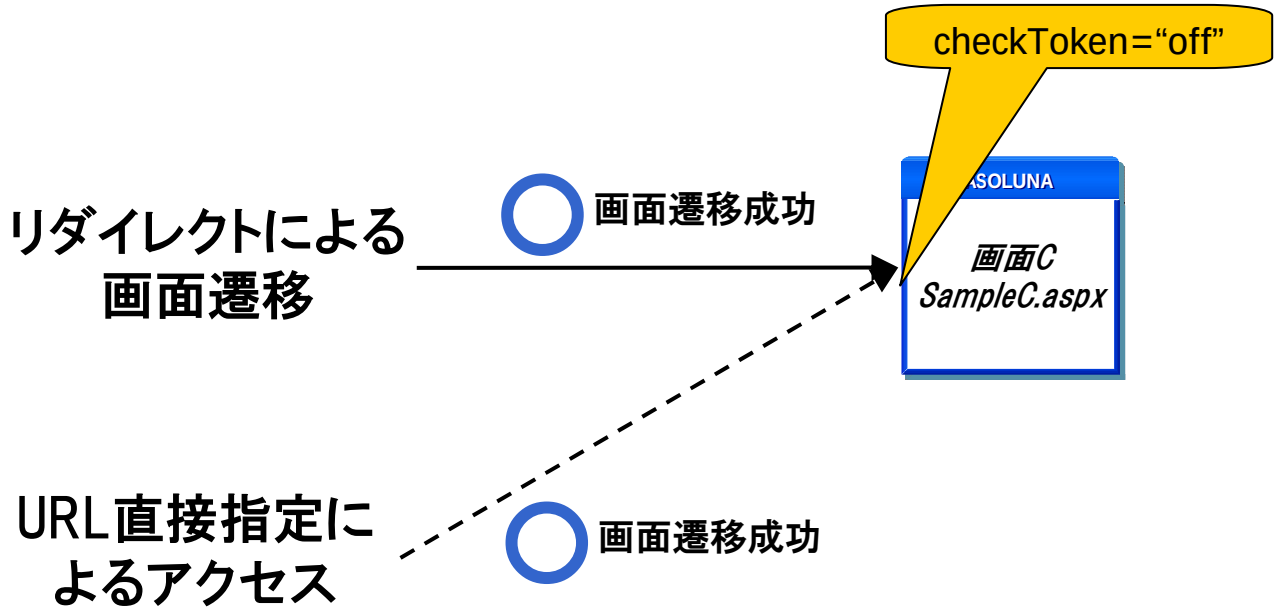
WA-02 画面遷移保証機能 – 使用方法(4/7)

- 動作例(checkToken="on" の画面へ遷移する場合)
  - checkToken="on" の画面へ遷移する場合は、Webアプリケーション内からのリダイレクトの場合は遷移でき、URL直接指定などの不正アクセスの場合は遷移できない



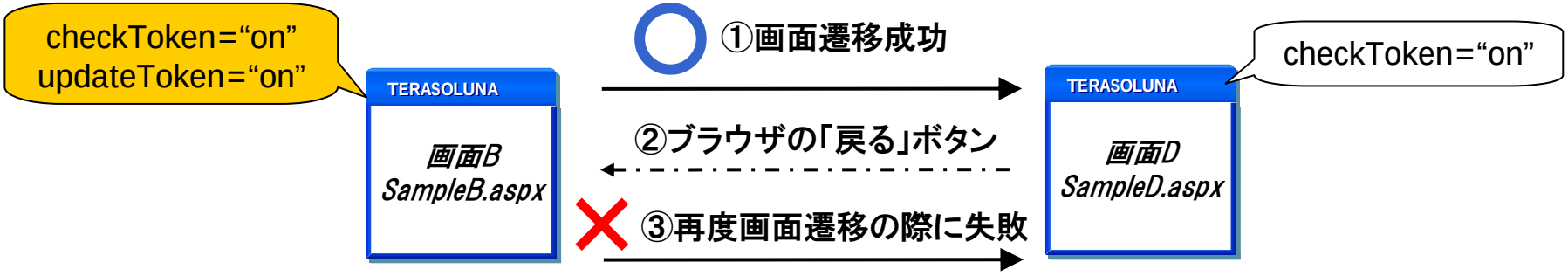
WA-02 画面遷移保証機能 – 使用方法(5/7)

- 動作例(checkToken="off" の画面へ遷移する場合)
  - checkToken="off" の画面へ遷移する場合は、Webアプリケーション内からのリダイレクトの場合も、URL直接指定などのアクセスの場合も遷移できる



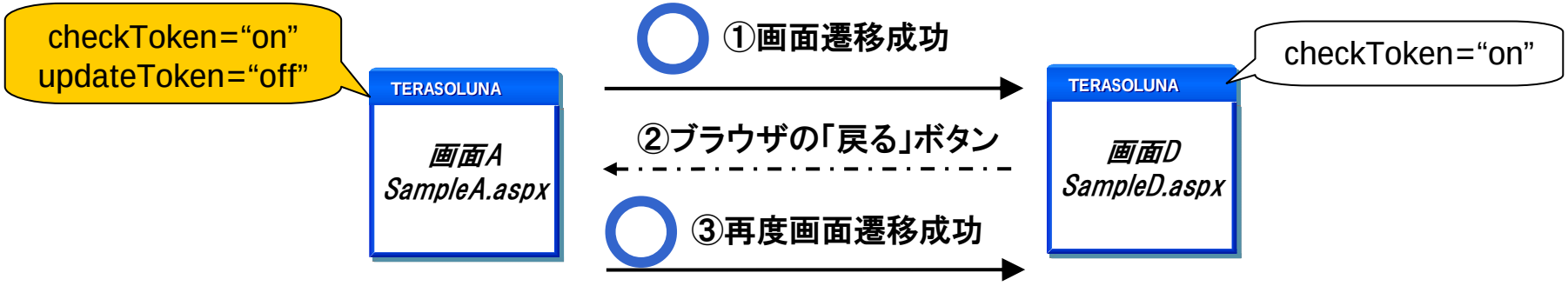
WA-02 画面遷移保証機能 – 使用方法(6/7)

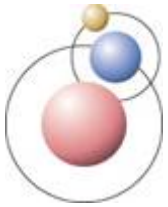
- 動作例(checkToken="on" かつ updateToken="on" の画面遷移)
  - ブラウザの戻るボタンにて checkToken="on" かつ updateToken="on" の画面に戻ってきた際に、再度 checkToken="on" の画面へ遷移するときに、不正画面遷移となる
    - ブラウザの戻るボタンで前画面へ戻った後の画面遷移を許可しない場合は、updateToken="on" にする



WA-02 画面遷移保証機能 – 使用方法(7/7)

- 動作例(checkToken="on" かつ updateToken="off" の画面遷移)
  - ブラウザの戻るボタンにて checkToken="on" かつ updateToken="off" の画面に戻ってきた際に、再度 checkToken="on" の画面へ遷移できる
    - ブラウザの戻るボタンで前画面へ戻った後の画面遷移を許可する場合は、updateToken="off" にする





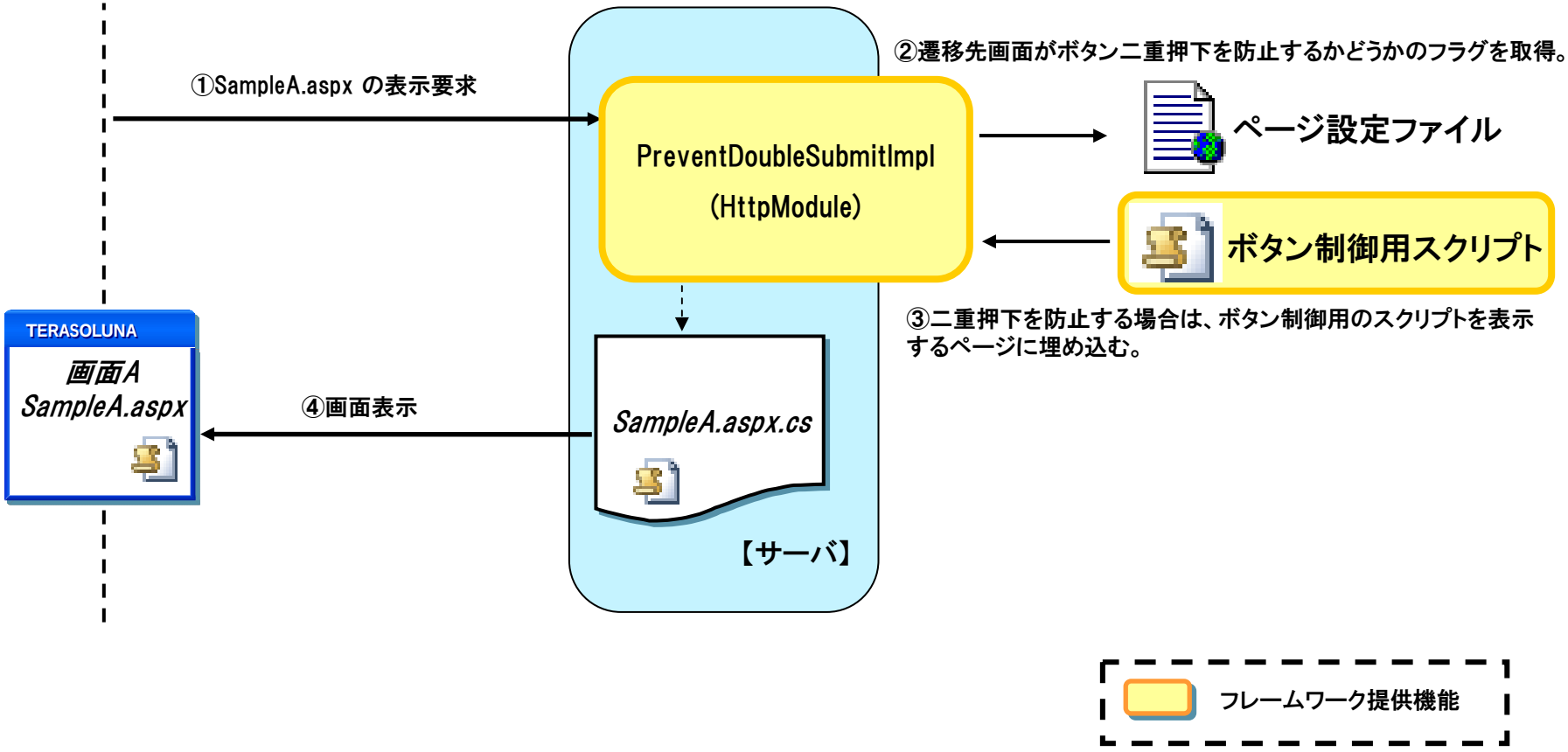
WA-03 二重押下防止機能 – 概要

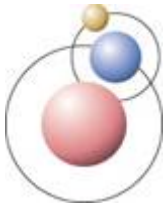
機能概要

- Submit ボタンが押下された際に画面内の全ての Submit ボタンを無効(disable)にして、リクエストの二重送信を防止する機能を提供する
  - 画面に JavaScript を埋め込むことで、Submit ボタン押下後の、他の Submit ボタンを無効にする
  - 画面内の Submit ボタンのみを無効にし、ハイパーリンクや AutoPostBack プロパティが有効に設定されたコントロールは無効にならない

WA-03 二重押下防止機能 – 動作イメージ

二重押下防止機能の内部動作





WA-03 二重押下防止機能 – 使用方法(1/3)

Web構成ファイルの設定(1)

- HTTPモジュールとしてPreventDoubleSubmitImplを設定する

```
<!--HTTPモジュールの設定-->
<system.web>
  <httpModules>
    <add name="PreventDoubleSubmitImpl"
         type="TERASOLUNA.Fw.Web.HttpModule.PreventDoubleSubmitImpl, TERASOLUNA.Fw.Web" />
  </httpModules>
</system.web>
```

WA-03 二重押下防止機能 – 使用方法(2/3)

Web構成ファイルの設定(2)

- 利用するページ設定ファイルのパスを設定する

```
<!--ページ設定ファイルのパスの設定-->
<configSections>
  <section name="pageConfiguration"
           type="TERASOLUNA.Fw.Web.Configuration.Page.PageConfigurationSection, TERASOLUNA.Fw.Web"/>
</configSections>
<pageConfiguration>
  <files>
    <file path="Config¥PageConfiguration.config" />
  </files>
</pageConfiguration>
```

WA-03 二重押下防止機能 – 使用方法(3/3)

- ページ設定ファイルの設定
  - ページID、遷移先画面のパス、二重押下防止フラグを設定する

```
<!--ページ設定ファイルの設定-->
<?xml version="1.0" encoding="utf-8" ?>
<pageConfiguration xmlns="http://www.terasoluna.jp/schema/PageSchema.xsd">
  <page name="SampleA" path="/Page/SampleA.aspx" preventDoubleSubmit="on“ />
</pageConfiguration>
```

WA-04 エラー画面遷移機能 – 概要

機能概要

- アプリケーション内で未処理の例外発生時に、例外の型に応じた任意のエラー画面に遷移する機能を提供する

（参考）ASP.NETにおけるエラー画面遷移

- ASP.NETでのエラー画面遷移は、HTTP ステータスコードに対応付けて設定できる
- 未処理の例外が発生した場合は、HTTP 500エラーとして処理できるが、例外の型に応じたエラー画面の振り分けはできない

web.config

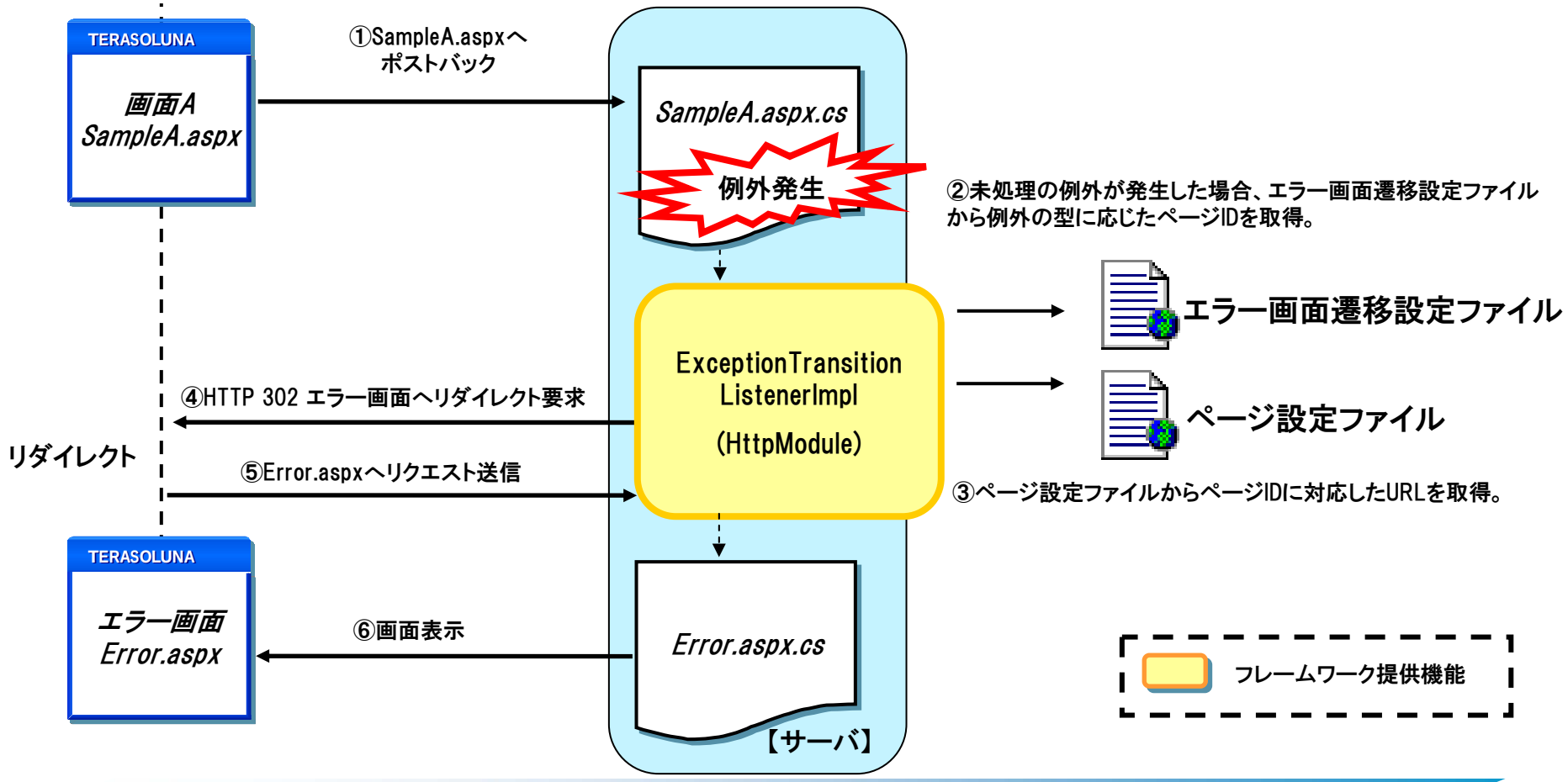
```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <system.web>
    <customErrors defaultRedirect="customError.html" mode="On">
      <error statusCode="403" redirect="NoAccess.htm" />
      <error statusCode="404" redirect="FileNotFound.htm" />
    </customErrors>

  </system.web>
</configuration>
```

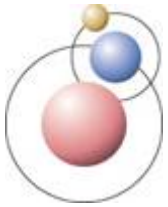
デフォルトの設定

HTTPステータスコード単位の設定

WA-04 エラー画面遷移機能 – 動作イメージ

エラー画面遷移機能の内部動作





WA-04 エラー画面遷移機能 – 使用方法(1/5)

- Web構成ファイルの設定(1)
  - HTTPモジュールとしてExceptionTransitionListerImpl を設定する

```
<!--HTTPモジュールの設定-->
<system.web>
  <httpModules>
    <add name="ExceptionTransitionListerImpl"
         type="TERASOLUNA.Fw.Web.HttpModule.ExceptionTransitionListenerImpl, TERASOLUNA.Fw.Web" />
  </httpModules>
</system.web>
```

WA-04 エラー画面遷移機能 – 使用方法(2/5)

- Web構成ファイルの設定(2)
  - 利用するエラー画面遷移設定ファイルのパスを設定する

```
<!--エラー画面遷移設定ファイルのパスの設定-->
<configSections>
  <section name="exceptionTransitionConfiguration"
           type="TERASOLUNA.Fw.Web.Configuration.ExceptionTransition.ExceptionTransitionConfigurationSection,
                 TERASOLUNA.Fw.Web"/>
</configSections>
<exceptionTransitionConfiguration mode="on" logging="on">
  <files>
    <file path="Config¥ExceptionTransition.config" />
  </files>
</exceptionTransitionConfiguration>
```

WA-04 エラー画面遷移機能 – 使用方法(3/5)

Web構成ファイルの設定(3)

利用するページ設定ファイルのパスを設定する

```
<!--ページ設定ファイルのパスの設定-->
<configSections>
  <section name="pageConfiguration"
           type="TERASOLUNA.Fw.Web.Configuration.Page.PageConfigurationSection, TERASOLUNA.Fw.Web"/>
</configSections>
<pageConfiguration>
  <files>
    <file path="Config¥PageConfiguration.config" />
  </files>
</pageConfiguration>
```

WA-04 エラー画面遷移機能 – 使用方法(4/5)

エラー画面遷移設定ファイルの設定

- 例外の型と、それに対応するページIDを設定する
- ページIDは、ページ設定ファイルのページIDとマッピングさせる

```
<!--エラー画面遷移設定ファイルの設定-->
<?xml version="1.0" encoding="utf-8"?>
<exceptionTransitionConfiguration xmlns="http://www.terasoluna.jp/schema/ExceptionTransitionSchema.xsd">
  <exceptionTransition exceptionType="System.Web.HttpException" nextPage="Error" />
  <exceptionTransition exceptionType="TERASOLUNA.Fw.Web.HttpModule.SessionTimeoutException" nextPage="Error"/>
  <exceptionTransition exceptionType="TERASOLUNA.Fw.Web.HttpModule.InvalidTransitionException" nextPage="Error2"/>
</exceptionTransitionConfiguration>
```

ページ設定ファイルのページIDとマッピングさせる

WA-04 エラー画面遷移機能 – 使用方法(6/6)

ページ設定ファイルの設定

- ページIDと遷移先画面のパスを設定する

```
<!--ページ設定ファイルの設定-->
<?xml version="1.0" encoding="utf-8" ?>
<pageConfiguration xmlns="http://www.terasoluna.jp/schema/PageSchema.xsd">
  <page name="SampleA" path="/Page/SampleA.aspx" />
  <page name="Error" path="/Page/Error.aspx" />
  <page name="Error2" path="/Page/Error2.aspx" />
</pageConfiguration>
```

エラー画面遷移設定ファイルの
ページIDとマッピングさせる

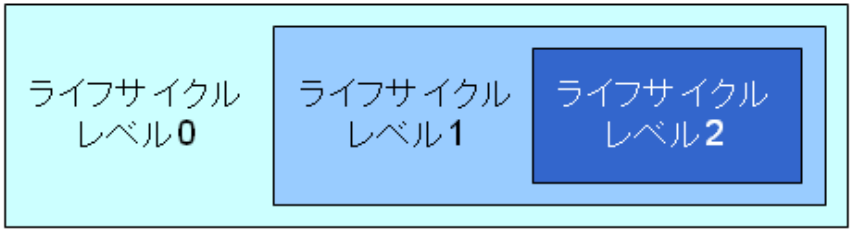
WC-01 セッション管理機能 – 概要

機能概要

- セッション情報をライフサイクルレベルで一括管理する機能を提供する
  - ライフサイクルレベルごとにセッション情報を管理することで、セッション情報のグループごとの一括削除を実現する

ライフサイクルレベルとは

- セッション情報をグループ化する単位のことで、ライフサイクルレベルごとにデータを管理する
- ライフサイクルレベルは包含関係を形成する



ライフサイクルレベル0はライフサイクルレベル1を包含し、ライフサイクルレベル1はライフサイクルレベル2を包含する。以降同様に、包含関係を形成する。

WC-01 セッション管理機能 – 使用方法、動作イメージ

セッション管理機能の使用方法と内部動作

- ライフサイクルレベルを指定してセッションに値を格納し、ライフサイクルレベル単位で一括削除する例を示す

```
SessionManager manager = new SessionManager(Session);
// Add(セッションキー名、セッションに格納するオブジェクト、ライフサイクルレベル)
manager.Add("key10", object10, 0);
manager.Add("key11", object11, 1);
manager.Add("key12", object12, 2);
manager.Add("key13", object13, 2);

// Remove(ライフサイクルレベル)
manager.Remove(1);
```

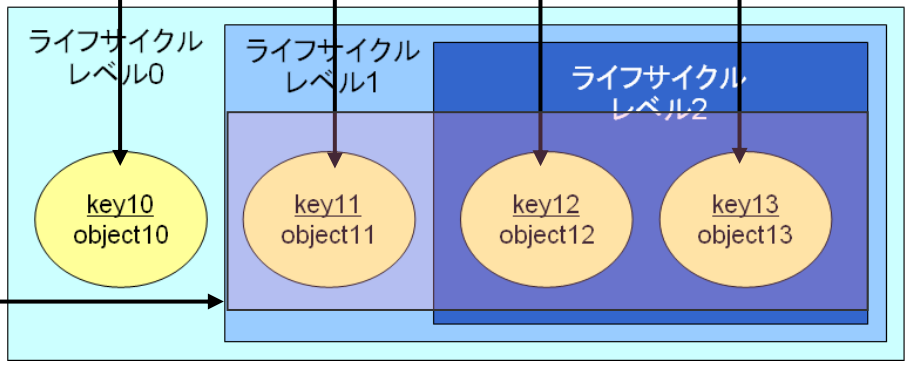
追加

追加

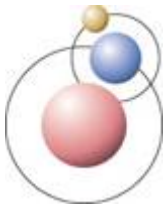
追加

追加

セッション内部



セッションからライフサイクルレベル1に
包含されるデータを全て削除



CM-02 入力値検証機能

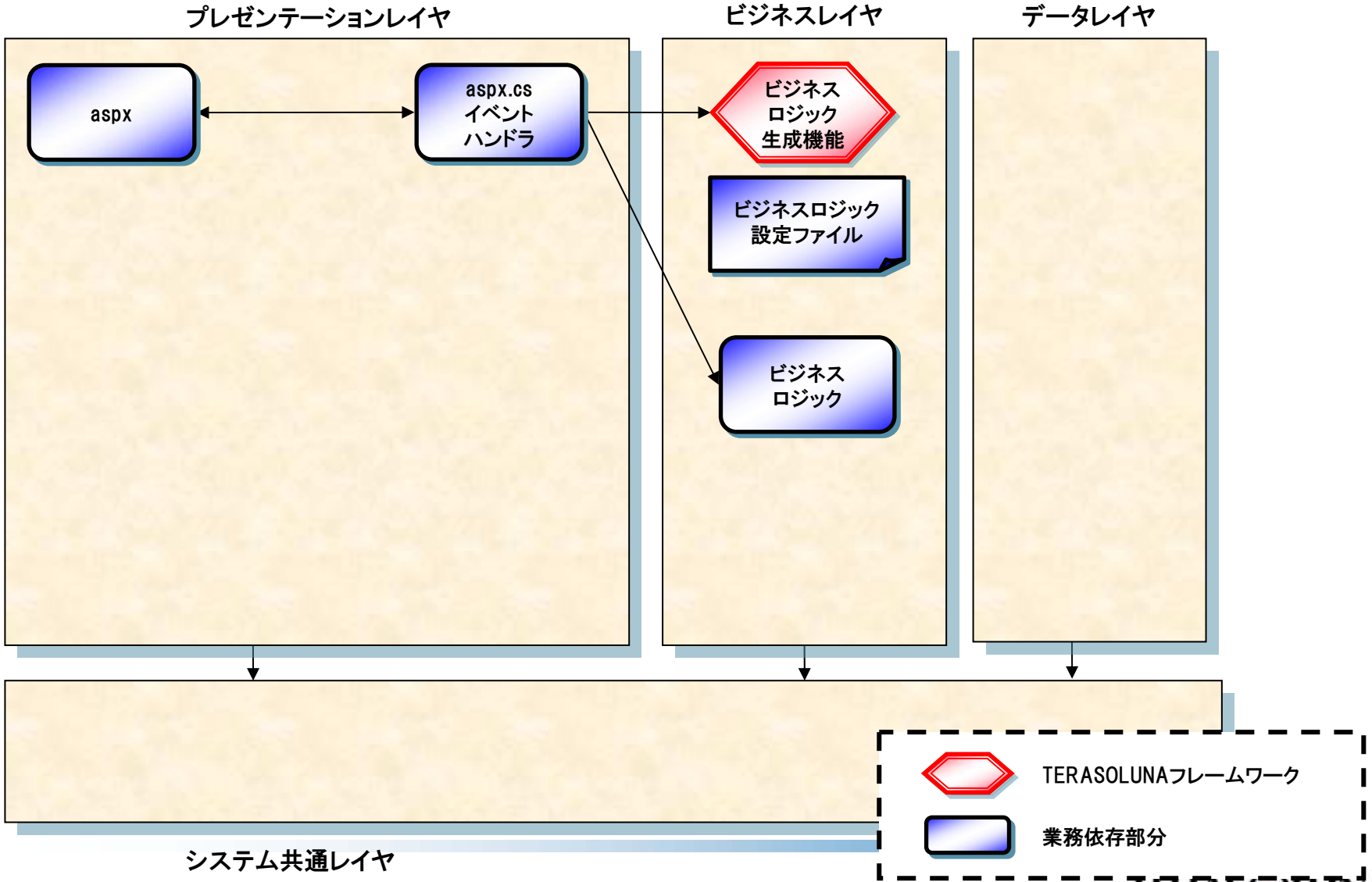
機能概要

- 様々な入力値検証ルールを提供する
- 詳細は「TERASOLUNA Server/Client Framework for .NET 2.1 アーキテクチャ説明書(共通編)」を参照のこと

CustomValidator の ServerValidate イベントの実装例

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    // 全角文字列チェックのValidatorを生成
    TERASOLUNA.Fw.Common.Validation.Validators.ZenkakuStringValidator validator
        = new TERASOLUNA.Fw.Common.Validation.Validators.ZenkakuStringValidator(null, "テキストボックス", false);
    // 検証
    Microsoft.Practices.EnterpriseLibrary.Validation.ValidationResults results = validator.Validate(TextBox1.Text);
    // 検証失敗の場合、args.IsValidプロパティの設定と検証エラーメッセージを設定する
    // 以下省略
}
```

ビジネスレイヤ　機能説明

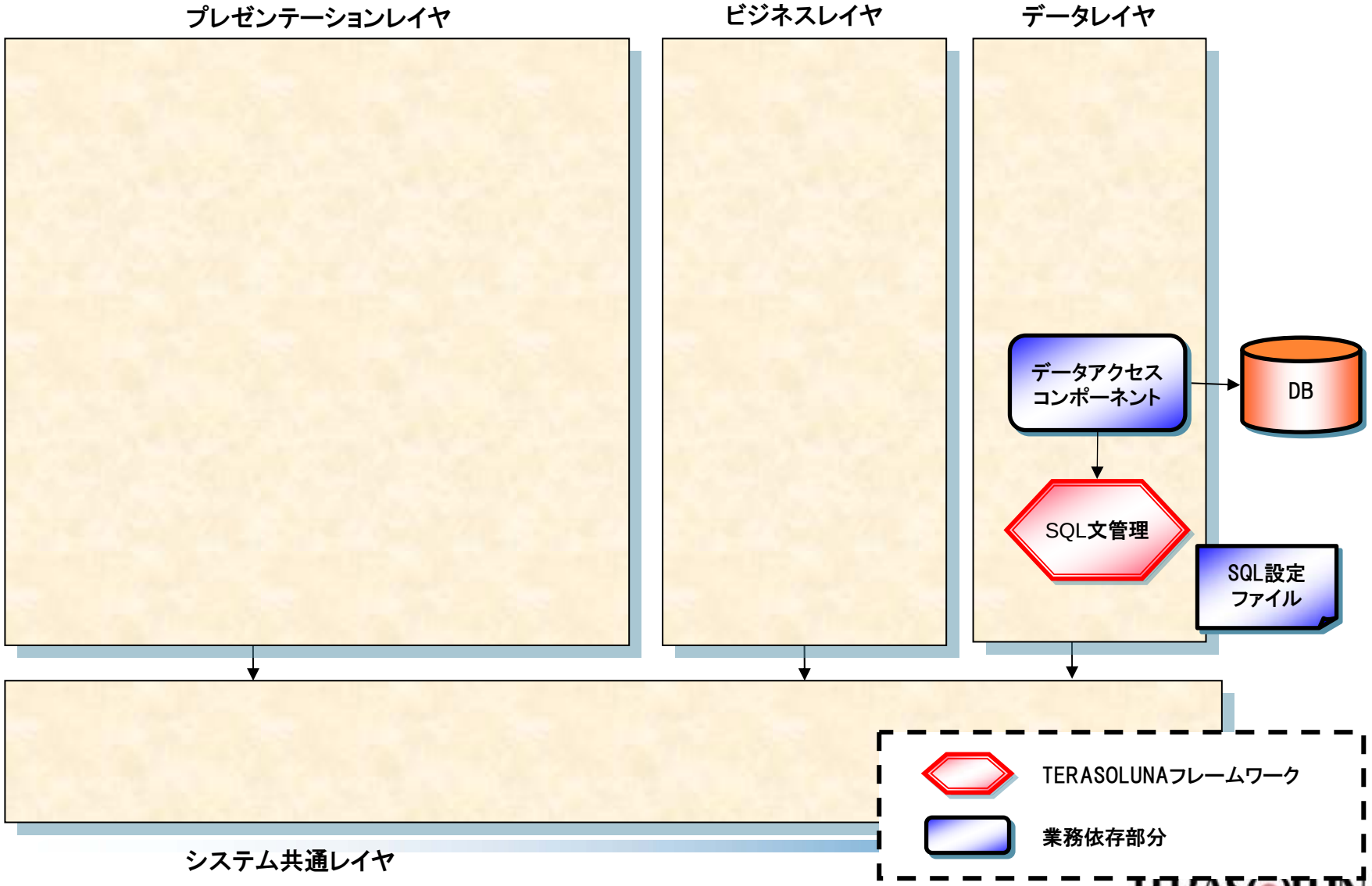


CM-04 ビジネスロジック生成機能

機能概要

- ビジネスロジッククラスのインスタンスを生成する機能を提供する
- 詳細は「TERASOLUNA Server/Client Framework for .NET 2.1 アーキテクチャ説明書(共通編)」を参照のこと

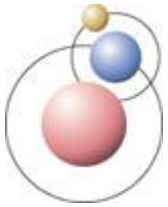
データレイヤ　機能説明



WC-02 SQL文管理機能

機能概要

- SQL文設定ファイルからSQL文を取得する機能を提供する
  - SQL文のIDに対応するSQL文を返却する
  - SQL文を設定ファイルから取得することにより、再ビルドを行わずにSQL文を変更できる

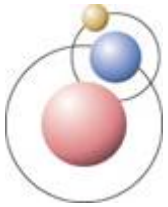


WC-02 SQL文管理機能 – 使用方法(1/2)

Web構成ファイルの設定

- 利用するSQL文設定ファイルのパスを設定する

```
<!--SQL文設定ファイルのパスの設定-->
<configSections>
  <section name="sqlConfiguration"
           type="TERASOLUNA.Fw.Web.Configuration.Sql.SqlConfigurationSection, TERASOLUNA.Fw.Web"/>
</configSections>
<sqlConfiguration>
  <files>
    <file path="Config¥SqlConfiguration.config" />
  </files>
</sqlConfiguration>
```



WC-02 SQL文管理機能 – 使用方法(2/2)

SQL文設定ファイルの設定

- SQL IDとSQL文を設定する

```
<!--SQL文設定ファイルの設定-->
<?xml version="1.0" encoding="utf-8" ?>
<sqlConfiguration xmlns="http://www.terasoluna.jp/schema/SqlSchema.xsd">
  <sql name="selectSample">
    <![CDATA[
    SELECT * FROM TABLE
    ]]>
  </sql>
</sqlConfiguration>
```

SQL文の取得

SQL文取得のコード例

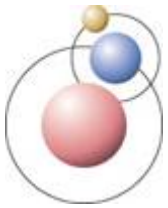
```
string query = SqlConfiguration.GetSql("selectSample");
```

システム共通レイヤ　機能説明

CM-01 メッセージ管理機能

機能概要

- アプリケーションで扱うメッセージに対し、統一的にアクセスする仕組みを提供する
- 詳細は「TERASOLUNA Server/Client Framework for .NET 2.1 アーキテクチャ説明書(共通編)」を参照のこと



CM-03 ログ出力機能

機能概要

- アプリケーションで統一的にログを出力する仕組みを提供する
- 詳細は「TERASOLUNA Server/Client Framework for .NET 2.1 アーキテクチャ説明書(共通編)」を参照のこと