



Web アプリケーションフレームワーク openGion 入門

Webエンジン Ver. 5.0対応
リリース 2.0

初版 : 2003 年 3 月
改訂 : 2003 年 8 月
第 2 版 : 2012 年 6 月 Ver5 対応



*Muratec Information
Systems.LTD.*

Web アプリケーションフレームワーク openGion 入門、リリース 2.0

原本部品番号:W1A0001-20

原本名: openGion Web Application Framework Introductory Book, Release 2.0

原著者: 長谷川 和彦

編集: 久田 雅子

Copyright © 2012、MURATEC INFORMATION SYSTEMS, LTD. All rights reserved.

Printed in Japan

制限付権利の説明

プログラム(ソフトウェアおよびドキュメントを含む)の使用、複製または開示は、ムラテック情報システムとの契約に記載された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。ムラテック情報システムは本ドキュメントの無謬性を保証しません。

* ムラテック情報システムとは、ムラテック情報システム株式会社を指します。

危険な用途への使用について

ムラテック情報システム製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。ムラテック情報システム社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、ムラテック情報システムおよびその関連会社は一切責任を負いかねます。

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

第 I 部	Web アプリケーションの概要	9
第1章	Web アプリケーションの基礎知識	10
	1. Web アプリケーションの特長	10
	2. エンジンの全体構想	13
	3. パッケージ	14
	4. 開発手順	18
	5. 使用ツール	20
	6. インストール手順	22
第 II 部	Web アプリケーション構造	24
第2章	ユーザーインターフェース(業務ロジックA)	25
	1. 業務ロジック A	25
	2. 業務ロジック B	25
	3. Web アプリケーションフレームワーク	25
第3章	ビジネスロジック(業務ロジックB)	27
第4章	エンジンコア(汎用オブジェクト)	28
	1. 表形式オブジェクト	29
	2. 汎用オブジェクト	30
	3. カラムオブジェクト	31
第 III 部	Web アプリケーション アーキテクチャ	32
第5章	Web アプリケーションの起動と停止	33
	1. startup.bat	33
	2. shutdown.bat	33
	3. workdelete.bat	33
第6章	アプリケーション・アーキテクチャ	35
	1. 画面制御	36
第7章	メモリー・アーキテクチャ	38
	1. データベース検索結果に関するメモリ管理	39
	2. リソース情報のメモリ管理	39
	3. Tomcat 起動時の Java オプション	40
第 IV 部	データ制御オブジェクト	41
第8章	リソース情報管理	42
	1. リソース管理	42
	2. リソースの階層管理	43
第9章	メッセージの使用方法	44
	3. メッセージの種類	44
	4. 国際化対応	44
	5. リソースの取得先	44
	6. メッセージの文法	45
	7. メッセージの表示	45

8. PL/SQL のエラーメッセージ.....	46
第10章 カラムオブジェクト.....	48
第11章 コードリソースとカラムオブジェクト.....	51
第12章 レンダラー、エディター、DBタイプ.....	52
9. 表示用レンダラー.....	52
10. 編集用エディター.....	54
11. データのタイプ.....	55
第13章 その他リソースファイル.....	57
1. ラベルリソース.....	57
2. メッセージリソース.....	57
3. ユーザーリソース.....	57
4. GUI リソース.....	58
第 V 部 データ・アクセス.....	61
第14章 SQL、PL/SQL および Java.....	62
1. 共通オブジェクトの定義.....	62
2. 検索系のコール条件.....	63
3. 登録系のコール条件.....	64
第15章 Query と Update.....	66
1. query タグ.....	66
2. PL/SQLUpdate タグ.....	67
3. tableUpdate タグ.....	67
4. queryType.....	68
第16章 View、WriteTable、ReadTable.....	69
1. view タグ.....	69
2. viewFormType 属性.....	70
3. writeTable タグ.....	71
4. writerClass 属性を、以下に示します。.....	72
5. readTable タグ.....	73
6. readClass 属性を、以下に示します。.....	74
第17章 tableFilter、process、schedule.....	74
1. tableFilter タグ.....	74
2. mainProcess タグ、process タグ.....	74
3. schedule タグ.....	74
第18章 filter (Servlet のフィルター機能の実装).....	74
1. Filter の設定.....	74
第19章 高度なデータアクセス.....	74
第 VI 部 ダイレクト・パス・インポートとダイレクト・パス・エクスポート.....	74
第20章 ダイレクト・パス・インポート.....	74
第21章 ダイレクト・パス・エクスポート.....	74
第 VII 部 アプリケーションの保護.....	74
第22章 アプリケーション・アクセスの制御.....	74
1. ユーザー認証.....	74

2. セキュリティ制限	74
3. directory listings	74
4. Digest 認証	74
5. HTTPS クライアント認証	74
第23章 権限、ロールおよびセキュリティ・ポリシー	74
1. 画面メニューのアクセス制限	74
2. 画面の登録ボタンのアクセス制限	74
第24章 アプリケーション監査	74
1. 状況表示	74
2. プール削除	74
3. ログインユーザー	74
4. プラグイン情報	74
5. タグリブ情報	74
6. システムリソース	74
7. パラメーター	74
8. アクセスストップ	74

はじめに

このマニュアルでは、Web アプリケーションの機能全般について説明します。このマニュアルによって、読者は Web アプリケーションの機能を理解することができます。

《対象読者》

このマニュアルは、Web エンジンのシステム管理者、アプリケーションの開発者を対象として記述しています。このマニュアルの読者は、システム管理、アプリケーション開発の概念に精通しているものと想定しています。前提条件として、すべての読者は『Web アプリケーション概要』の第1章「Web アプリケーション概要」に記載されている内容を理解している必要があります。第1章では、このマニュアルのその他の章で使用される概念と用語について、幅広く説明されています。

《本文の表記規則》

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則を示しています。

規則	意味
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。この句を指定する場合は、索引構成表を作成します。
大文字	大文字は、システムにより指定される要素を示します。
小文字	小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびサンプルのユーザー指定要素を示します。 注意 ：一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。
イタリック	イタリックは、プレースホルダまたは変数を示します。

《コード例の表記規則》

次の表は、コード例の記載上の表記規則を示しています。

規則	意味
[]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体は入力しないでください。
	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。
⋮	省略記号は、例に直接関係のないコード部分が省略されていることを示します。

《アイコン》

本文中には、特別な情報を知らせるために、次のアイコンが用意されています。



ヒント

提案や秘訣を示し、これらによって、時間の節約や手順の容易化などを実現できる場合があります。



警告

システムに致命的な影響を及ぼす可能性のあるアクションについて、注意が必要であることを示します。

コラム



関連する基礎知識や細かい技などを解説しています。

第 I 部

Web アプリケーションの概要

『完成させるのが目的じゃないんですよね！
自然界に完成は無いんです・・・・・・・・
いつまでも作り続けることが大切だと思うのです』

外尾悦郎 サグラダファミリア主任彫刻家

ここでは、Web アプリケーションフレームワーク openGion の概要について説明します。
構成は、次のとおりです。

第 1 章 Web アプリケーションの基礎知識

Web アプリケーションを理解する上で必要になる概念と機能について概説します。このマニュアルで説明されている詳細な情報を読む前に、この章をお読みください。

openGion
openGion
フレームワーク

第1章 Web アプリケーションの基礎知識

この章では、Web アプリケーションを理解する上で必要になる概念と機能について概説します。このマニュアルで説明されている詳細な情報を読む前に、この章をお読みください。

1. Web アプリケーションの特長

近年、オリジナルの PDM パッケージを作成するにあたり、Web アプリケーションでの開発が不可欠です。その一方で、業務アプリケーションは、ますます複雑かつ短期集中型になりつつあります。そこで、あらかじめ標準的な機能をフレームワークとして提供しておき、個別に業務アプリケーションを構築していく事により、短期間に高品質な Web アプリケーションを開発することが可能になります。

① Web アプリケーションの共通機能

- 国際化対応
国際化対応(ラベル、選択リスト、メッセージ)ラベル、選択リスト(HTML のメニュー)、メッセージなど、日本語、英語などの言語に依存するリソースを、アプリケーションからすべて分離し、プロパティファイルで扱うようにしました。また、国際化機能は、フレームワークに組み込まれているため、アプリケーション開発者は、開発時にまったく意識する必要がなく、通常の SQL 文を投げるだけで、国際化対応の画面を作成する事ができます。
- 個人ポータルサイト作成
ログイン時にユーザーを識別するため、個人ごとに自由に環境を設定することが可能です。個人宛のメールや、仕事残、スケジュール管理、進捗報告など、個人ごとに管理内容の異なるアプリケーションを簡単に導入する事ができます。
- アクセス制限の実施
ユーザーID とパスワードにより、アプリケーション(システム)のアクセス制限が付けられます。さらに、各画面に設けられた、ロールと、ユーザー個々のロールの組み合わせにより、ユーザー単位に、画面個々のアクセス制限と、リード/ライトを指定できます。
- 電子メールによる簡易ワークフローの実現
ユーザー情報として、ロールを持っており、それらを利用してメールを送信することが可能です。
- DBユーザーと利用ユーザーとの分離
Web アプリケーションでは、不特定多数のユーザーがログインする可能性があり、また、HTTP のプロトコル上、ステートレスになっています(ステート管理は、クッキーや、JSP のセッションで管理)。本フレームワークでは、データベース接続に、

コネクションプーリング技術を採用するにより、パフォーマンスを犠牲にせず、不特定多数の同時接続に対応しています。

② 開発工数の削減

- データベースアクセス
データベースに対して SQL を投げると、フレームワーク側で自動的に ResultSet から DBTableModel に変換し、JSP に渡して HTML 表示を行います。開発者が SQL 文をフレームワークに渡すだけで、その結果を HTML 画面に表示するという単純な方法です。複雑な業務ロジックの場合は、スタアドプロシージャ化する事により、業務ロジックの隠ぺいを行うことが可能になります。
- カスタマイズやシステム変更が容易な構造
フレームワークを構築するにあたり、オブジェクト指向技術 (Java の全面採用) を活用し、カスタマイズやシステム変更が容易な構造になっています。(但し、画面、データベース、業務ロジックなどのオブジェクト指向的でない過去からの資産が関係する部分は、オブジェクト指向的なアプローチは取っていません。)
- リソースファイル
リソースファイル (初期設定) を利用して、開発工数を削減することができます。国際化対応時は、リソースファイルを翻訳するだけで、言語の切り替えが可能です。

③ 保守性の向上

- 画面 (JSP) とデータ (ORACLE) と業務ロジック (SQL) の切分け / 連携
MVC (Model-View-Control) に明確に分ける事により、開発効率を高めました。画面の表現は、CSS (Cascading Style Sheets) を採用し、HTML4.01 および、XHTML1.0 に準拠した Web ページを出力します。業務ロジックも View 系は JSP、バッチ系は PL/SQL と分けましたので、従来の開発手法、および、ノウハウは、そのまま利用可能です。
- データベースの作成
データベース定義を先に作成し、そこからデータベース作成ファイルや、ロード定義、その他のツールを自動で作成することが可能です。また、運用中のデータベースの状態管理 (エクステンツ、テーブルスペースなど) も、Web 上から簡単に管理する事ができます。
- ログレベルの設定
JSP、JavaBeans 共に、共通のログ出力機能を使用しています。また、ログ出力箇所ごとにログレベルを設定しておけば、外部からログレベルを指定することにより、簡易ログや、詳細ログを設定することが可能です。
- システム管理の遠隔 (Web ブラウザ) 操作

ユーザーのログイン制限や、接続プーリングのリフレッシュ、キャッシュデータの開放など、一般的なシステム管理やシステム状態の監視を、遠隔から Web ブラウザ経由で行うことが可能です。

④ 販売戦略を意識

- 同時接続ユーザー数の制限(ライセンス販売時)
本フレームワークのコネクションプーリング技術で、プール数をシステムプロパティで設定できるようになっています。そのため、このフレームワーク上で同時接続数に制限をかけることにより、評価版と実稼動版の区別をつけたり、ライセンス販売時のコスト差に対応したパフォーマンスを提供させる事が可能です。
- BASIC 認証とフォーム認証に対応
ログイン時のセキュリティは、インターネットでの Web アプリケーションのユーザー認証の基本である、BASIC 認証とフォーム認証のどちらにも対応しています。個々の業務に合った方法で認証を行う事が可能です。



コラム MVC (Model-View-Control) とフレームワーク

最近の Web アプリケーションでは、MVC を、Model=JavaBeans、View=JSP、Control=Servlet と分けて考える傾向があります。もちろん、これは悪いことではなく、非常に『考え方としては』良いことです。

では、この考え方が、本当にベストかといわれると、色々な問題点が出てきます。その中で2つの大きな問題点を挙げるとすると、1つは、アプリケーションが複雑になり、見通しが利きにくくなるというものです。もう1つは、Control とは、一体何か・・・ということですが。

1つ目のアプリケーションの複雑化については、Web アプリケーションのコントロール自身が本当に複雑かと問われた場合は、私は、No と考えています。なぜなら、元々、リンク、ボタン、テキスト/メニュー入力 というプアーなインターフェースしか持たない Web アプリケーションにおいて、純粋な MVC に分けて管理する必要があるのか、それだけの価値があるのかという疑問です。まず、必要ないと考えています。

2つ目の問題点は、もっと抽象的です。(いわば、哲学の世界ですね。)

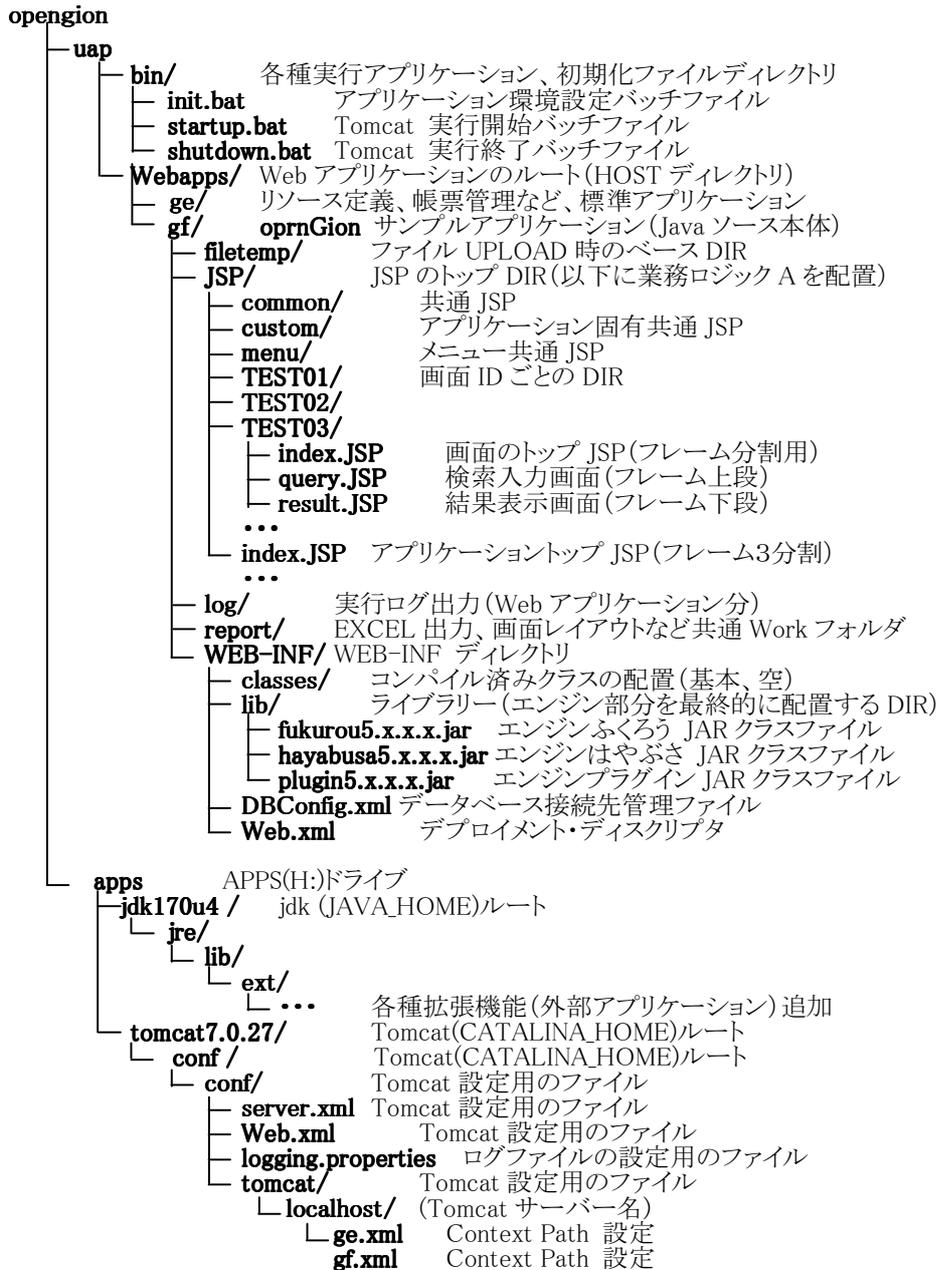
人が何か行おうとすると、一般には画面から操作を行います。また、画面から操作された命令を元に、アプリケーションがさらにコントロールします。しかし、どちらにしても、この操作のトリガ(きっかけ)は、画面であり、人による操作です。

このコントロールの結果、何らかの処理が行われて画面に結果が表示されます。この、コントロールは、View の1機能ではないか・・・という考えが、この Web アプリケーションフレームワークにあります。つまり、コントロールの方法を変えたり、その結果、現れる画面も、どちらも View を修正する必要があり、さらに、Control を修正するというのは、管理対象が分散していることを示しています。管理は1箇所で行うべきで、で、あれば、変更が発生する箇所(つまり、View)で Control を管理すべきという結論になります。

もちろん、JSP で記述するカスタムタグ内部で Control するのであって、JSP に Control ロジックをゴリゴリ書くという趣旨ではないことを、付け加えておきます。

2. エンジンの全体構想

Web アプリケーションフレームワークのエンジン部分の構成を次に示します。ディレクトリ構成は、標準的な Web アプリケーションに準拠します。JSP/Servlet エンジン、Tomcat を想定していますが、基本的にはどのエンジンでも動作するように標準的な作りになっています。



3. パッケージ

① パッケージ概要説明

Web アプリケーションフレームワークのエンジン部分のパッケージの概要説明を次に示します。

org.opengion.fukurou パッケージ

business	業務ロジックを処理するためのクラスを提供します。業務ロジックを、PL/SQLではなく、Java で記述するための関連クラスを収めたパッケージです。
db	データベースの Connection オブジェクトを管理するクラスを提供します。DBConfig.xml で定義された DBID を元に、接続先情報を作成し、Connection オブジェクトを取り出します。Connection オブジェクト関連のクラスを収めたパッケージです。
mail	はやぶさのメール(SMTP/POP3 クライアント)関係のクラスを提供します。このパッケージは、単独クラスとして、他の openGion パッケージと切り離されています。
model	[PN],[OYA] などの □ で指定されたカラムで表されたフォーマットデータに対して、DataModel オブジェクトを適用して 各カラムに実データを割り当てる処理を行うクラスを提供します。DataModel インターフェースは、1行をあらゆる情報を保持します。この配列型のクラスは、org.opengion.hayabusa.db.DBTableModel になります。
process	大量データの登録や抜き出しなど、バッチ的な処理を行うフレームワークを作成します。このフレームワークに、基本機能の実装と、各種システム固有機能の実装が簡単に行えるように、プラグイン方式を採用します。なお、このフレームワーク単独での採用は検討していません。エンジンとしては、一体として扱います。
security	セキュリティ強化の為に Hybs 独自の暗号化クラスを提供します。ここでの暗号化は、秘密キー方式でバイト文字列に変換されたものを、16進アスキー文字に直して、扱っています。よって、暗号化/復号化共に、文字列として扱うことが可能です。
taglet	このパッケージは、フレームワークドキュメントを自動生成・DB 登録するのに使用するドックレット、タグレットクラスを提供します。
transfer	伝送システム関係のクラスです。伝送データを読み取るデーモンクラスは、org.opengion.plugin.daemon 内にあります。このパッケージは、主に実行方法の処理を定義するクラス及び伝送定義を管理するためのデータクラスを管理します。
util	はやぶさのユーティリティ関係のクラスを提供します。このパッケージは、単独クラスとして、他の openGion パッケージと切り離されています。ユーティリティクラスは、他のパッケージで使用する便利クラス群です。
xml	はやぶさ共通のXML関連クラスを提供します。このパッケージは、単独クラスとして、他の openGion パッケージと切り離されています。ここで扱う XMLファイルは、拡張オラクル XDK形式のXMLファイルを処理するクラス群です。

org.opengion.hayabusa パッケージ

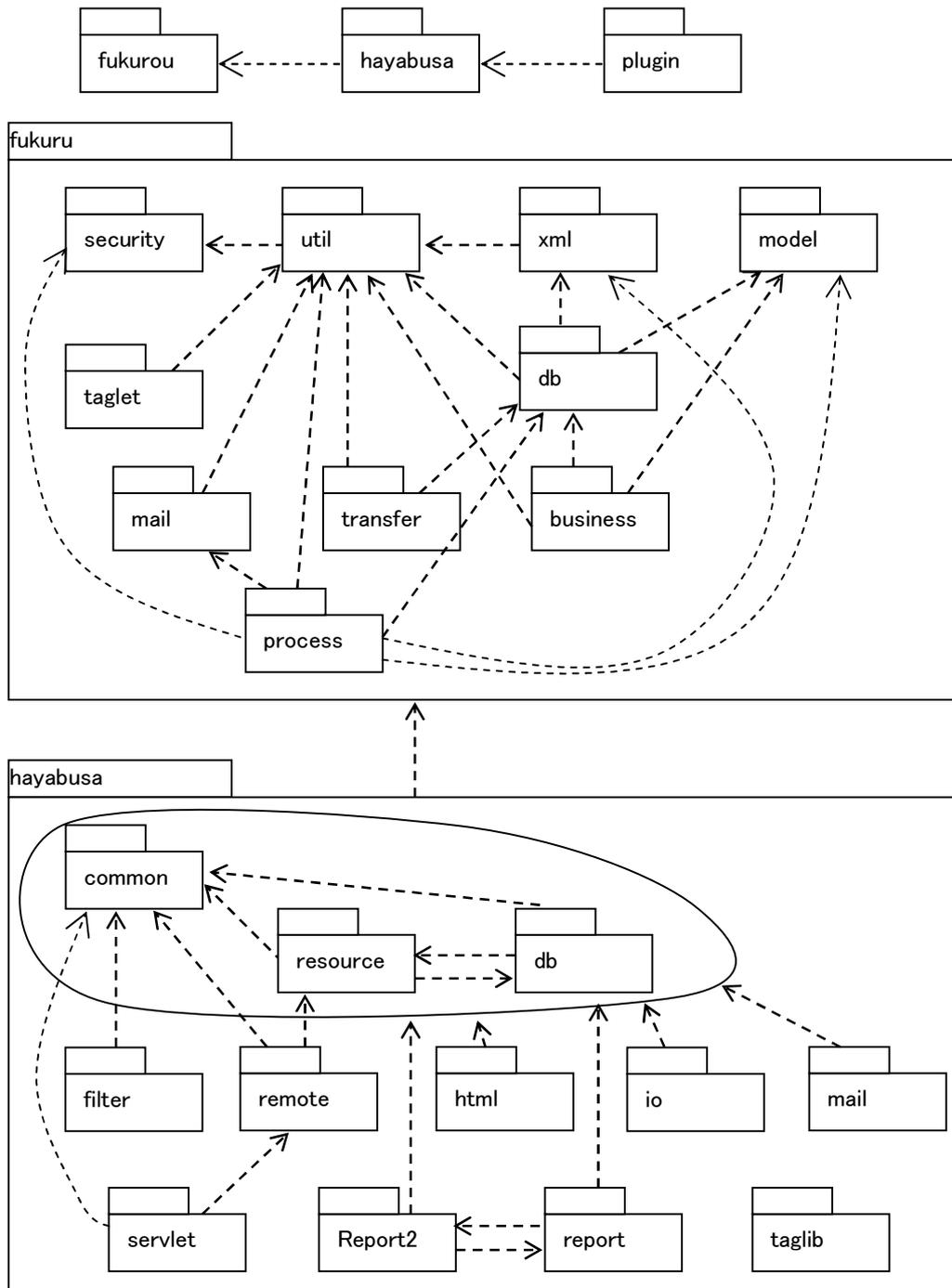
common	共通クラスフレームワークを利用する上でシステムリソースにアクセスしたり、例外 (Exception) を加工したり、ログを制御したりする共通的に利用するクラス群です。このパッケージのクラスは、他のパッケージから独立しています。
db	DB アクセス関連、DBTableModel、DBColumn 等データベースアクセス関連のクラス群です。特に、DBTableModel は、他のパッケージからも利用されるなど、フレームワークの中核をなすクラス (インターフェース) です。
develop	はやぶさの開発サポート関係のクラスを提供します。 ここでは、JSP ひな形から JSP 画面を作成する機能を提供します。
filter	JSP のフィルターチェーン用のフィルタークラス群です。各種フィルターを用意しています。
html	HTML 関連ユーティリティ ViewForm インターフェースと、その Abstract クラス、および、ViewLink、ViewMarker インターフェースとその実装、各種 Param 属性のキーファイルを提供します。
io	ファイル入出力関連 DBTableModel をファイルに書き出したり、ファイルから読み込んだりするクラス群です。チェックイン/チェックアウトや EXCEL 出力などのコア機能を提供します。
mail	メール伝送システムの関連クラスを提供します。メール伝送システムは、メール伝送に必要な情報を、データベースに登録することで簡単にメール関係の処理を行うことが出来るクラスです。
remote	RemoteControllable インタフェースを実装したサーブレット経由で遠隔リソース更新を行うためのクラスです。Tomcat 違いのコンテキストで、メモリ内のリソース関係のキャッシュをクリアさせます。将来的には、各種リモートコールのクラスを実装していく予定です。
report	Excel 帳票関連のクラス群です。印刷、PDF 化などを、EXCEL の雛型より作成できます。
report2	OpenOffice を利用した Calc 帳票システム関係のクラスです。 従来の Excel 帳票は、互換性のため残っていますが、こちらのパッケージが今後の主流の帳票システムになります。
resource	国際化 (リソース) 管理関連リソース関連のハンドリングを行うクラス群です。リソースそのものは、データベースで管理し、内部でオブジェクト化して、共有利用しています。
servlet	Servlet 関連のユーティリティクラス群です。FileUpdate 時の処理など、Servlet 機能を利用するクラス群を分けています。 現時点では、純粋なサーブレットクラスは、使用していません。
taglib	画面表示用タグライブラリー画面の JSP からアクセスされるのは、この Taglib のみです。JSP のスクリプトは使用を誤るとメンテナンス性が落ちるため、必ず Taglib からアクセスさせます。よって、画面作成時に必要な機能が出た場合は、Taglib の開発をエンジンチームが行います。

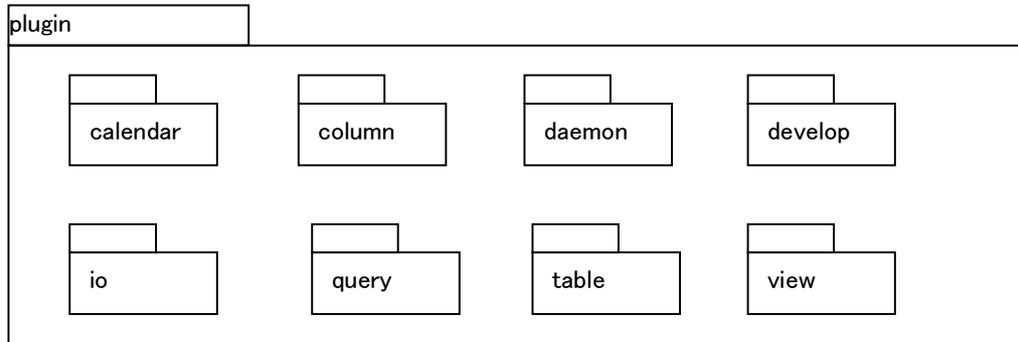
org.opengion.plugin パッケージ

calendar	CalendarQuery は、カレンダーデータベースを読み込む Query を定義しておく、エンジンでの内部表現は、統一して扱う為のプラグインです。
column	カラムリソース関連 PlugIn のクラス群です。カラムリソースは、内部に、Renderer、Editor、DBType というオブジェクトを持っています。 検索結果の表示には、Renderer を、結果を編集する場合は、Editor を、そのデータの種類(桁数、半角/全角 など)を規定し、チェック、正規化 を行うのが DBType です。
daemon	org.opengion.fukurou.util.HybsTimerTask の実装クラスで、デーモンクラスです。帳票デーモン、メールデーモンなど、各種デーモン系クラスの実装です。
develop	はやぶさの開発サポート関係のクラスの拡張クラス(プラグイン)を提供します。ここでは、JSP ひな形から JSP 画面を作成する機能の実装部を提供します。
io	ファイル入出力関連 PlugIn クラス群です。ファイル入出力には、ChartWriter、TableReader、TableWriter というインターフェースがあります。
query	Query は、データベースからの検索、登録を行うクラスを準備しています。基本的には、エンジン標準の DBTableModel オブジェクトを構築するか、そのオブジェクトの内容をデータベースに設定する作業を行います。
table	TableFilter は、TableFilterTag で呼び出される実装クラス群です。このクラスは、検索結果等の DBTableModel に対して、加工します。代表的なのは、CleateTable 文などの作成や、カラム ID をキーに、データベースやリソースとマッピングしたりする機能です。
view	画面表示を行う ViewForm の実装クラス群です。このクラスは DBTableModel を画面に表示するにあたってのコア実装クラス群にあたります。

② パッケージ関連図

前に説明した Web アプリケーションフレームワークのエンジン部分のパッケージの関連図を次に示します。





4. 開発手順

フレームワークを利用して開発する場合の作業手順を次に示します。

① 基本設計

基本設計は従来の開発と同様に行います。但し、次の資料は、従来、詳細設計のフェーズで作成していましたが、当フレームワークでは、基本設計の段階で作成する必要があります。

- DB 定義書(完成版・・・詳細設計で項目の変更可)
- 画面一覧(画面 ID、名称、アクセス制限等の基本属性は必要)

DB 定義で使用するカラム名は、全テーブルにおいてユニークにする必要があります。さらに、カラム属性として、文字種別(S9、XL、XU、R など)と GUI 種別を定義しておく必要があります。また、フラグやコードなど指定値以外の登録ができないものについては、あらかじめすべての項目の定義値を登録しておく必要があります(プルダウンメニュー)。これらの情報は、リソースファイルとしてフレームワークで利用されるので、開発に入る前に準備しておく必要があります。

② 画面サンプル(納入仕様書)

当フレームワークでは、検索系画面作成については、簡単に直接 JSP で記述することができます(EXCEL や HTML で作成しなくてよい)。そのため、作業効率を大幅に向上させることができます。もちろん、すべての画面やすべての DB 項目が埋まる必要はなく、ユーザーと打合せながら、画面サンプルを作成していきます。詳細設計に入る前に、実際に動作する画面をすべて作成してしまうため、従来のウォーターフォール方式からスパイラル方式へと発想を切り替える必要があります。このフェーズで画面 ID 一覧、画面遷移図、DB 項目定義を完成させます。

③ 詳細設計

検索系については、前フェーズでほとんど開発は終了している状態なので、あえて仕様書を作成する必要はありません。登録系についても、DB 定義で各カラムの論理属性を定義しておくので、ほとんどの設計については、仕様書を作成する必要はありません。ここで従来と同様の仕様書が必要となるのは、業務ロジック B といわれる PL/SQL でのリアルタイムバッチ処理に関する仕様書になります。これは、

JSP 画面から PL/SQL を CALL し、実行結果を直接/間接的に返して画面に表示しますが、PL/SQL の処理は一般に仕様展開や逆展開、差分などの高度な処理のため仕様書が必要となります。なお、ここでの仕様書は、開発のための仕様書であり、維持(保守)のための仕様書ではないため、このフェーズできっちり作成する必要はありません。また、フレームワークの機能不足により画面が作成できないケースが発生すると思われませんが、この段階で、フレームワークチームと共同で、フレームワークの機能強化を行います。各プロジェクトで個別に対応する事を避けるとともに、専門チームによる品質の高い部品を標準機能として拡張していくことにより、次回以降の開発期間の短縮も図ります。

④ 開発／単体テスト

環境作成は、DB 定義や画面一覧を、フレームワーク上のテーブルに登録しておきますが、この作業は、すでに基本設計、または画面サンプル作成時点で終了しています。このデータを元に、予想データ件数を登録することにより DB 作成コマンドや容量計算を自動で行い、テーブル作成も自動化します。本番サーバーに対してテーブルを作成し、実データを開発前に導入すれば、すでに検索系は本番と同等の機能として実現できているため、データの不具合や画面項目不足、検索速度などの確認がすぐに行えるようになります。そして、ユーザーの理解のもと、早い段階で動作を見てもらい、不具合点や機能不足の指摘を受けられるようにしておきます(その指摘に対して対応するかどうかは、機能の大きさと優先順位により決定)。実開発は、各画面のチューニング(使い勝手の向上やデザイン変更など)と、PL/SQL の実装になります。もちろん、JSP と PL/SQL との連結動作も確認しておく必要があります。

⑤ 総合テスト

総合テストは、従来の方法と同じく、全体機能を見ながら各モジュール間の連結不具合をチェックしていきます。これより前に、特定のユーザーにはオペレーションして頂いていますが、このフェーズでさらに、オペレーションのトレーニングを行う必要があります(操作マニュアルも必要)。ただし、これらは、もっと前に行うこともできます。

⑥ 本番導入/導入後サポート/維持仕様書作成

本番導入後安定するまで、導入後サポートを行います。また、この期間に、完成されたアプリケーションに関する正確な設計仕様書を作成します。ここでは、維持(保守)に必要な情報を仕様書にまとめていきます。最終ドキュメントは、ソースコードであり、そこにたどりつくための資料という位置付けなので、主に青紙の表紙(概要と DB アクセスと処理内容)と、画面遷移を作成します。

5. 使用ツール

フレームワークを構築するにあたり、各種ツール類を利用しています。以下に、各使用ツールと、概要説明を示します。

① Java エクステンション 設定 API

C:\opengion¥apps¥jdk170u4¥jre¥lib¥ext フォルダに各種 API をコピーしておくことで、パス設定などを行うことなく利用できます。

No	分類	ファイル名	機能
1	JDK	jdk170u4	Java Development Kit
2	Tomcat	tomcat7.0.27	Servlet / JSP コンテナ
3	JDBC	ojdbc6.jar orai18n.jar	Oracle JDBC Driver
4	JDBC	sqljdbc.jar	SQLServer JDBC Driver
5	JDBC	postgresql-8.4-701.jdbc4.jar	Postgresql JDBC Driver
6	JDBC	mysql-connector-java-5.1.10-bin.jar	MySQL JDBC Driver
7	JDBC	hsqldb.jar	Java ベースの組み込み用 DB
8	JDBC	h2-1.1.118.jar	組み込みデータベース
9	JDBC	jaybird-full-2.1.6.jar	Firebird プロジェクトが開発しているタイプ 4 jdbc ドライバです
10	JDBC	ogHsqldbExtension.jar	openGion で利用される HSQLDB 用の拡張用ファンクション
11	POI	poi-3.8-20120326.jar poi-excelant-3.8-20120326.jar poi-ooxml-3.8-20120326.jar poi-ooxml-schemas-3.8-20120326.jar poi-scratchpad-3.8-20120326.jar	Java ベースの Excel ファイルを操作するためのライブラリ
12	POI	xbean-2.5.0.jar	Java types(データ型定義) にバインドする事で XML にアクセスする技術であり、StAX 仕様に基づくオープンソースの XML-Java バインディングツール
13	POI	log4j-1.2.16.jar	Java の標準的なロギングライブラリ
14	POI	dom4j-1.6.1.jar	Java プラットフォーム上で XML, XPath, XSLT を扱える、単純で柔軟性に富んだオープンソース・ライブラリ
15	POI	geronimo-stax-api_1.0_spec-1.0.jar	オープンソースの Java アプリケーション・サーバー
16	POI	stax-api-1.0.1.jar	Java で XML 文書を読み書きするための API である
17	MAIL	mail-145.jar	Java でメールの送信の行うための API
18	Chart	jcommon-1.0.17.jar jfreechart-1.0.14.jar	Java ベースの画像生成ライブラリ

第 I 部 Web アプリケーションの概要

19	EXCEL	jxl-2.6.12.jar	Java ベースの Excel ファイルを操作するためのライブラリ
20	OOF	openoffice3.2	オープンソースで配布されているオフィスソフト
21	EXCEL	odfdom.jar xercesImpl.jar	ODF(OpenDocument Format)の文書を扱うための Java API
22	JavaScript	jquery.form.js jqDnR.js	JavaScript フレームワーク
23	QR	Qrcode.jar	Java ベースの QR コード生成ライブラリ
24		jai_codec1.1.3.jar jai_core1.1.3.jar	
25	XML	jaxen-1.1.4.jar	W3C DOM、JDOM、dom4j、XOM などのドキュメントツリーに対して XPath 評価を行うことができる
26	セキュア	jcifs-1.3.17.jar	Windows のファイル共有プロトコル、CIFS/SMB networking protocol
27	セキュア	jsch-0.1.48.jar	pure Java な SSH-2 実装
28		junit-4.10.jar	Java で開発されたプログラムにおいてユニットテスト(単体テスト)の自動化を行うためのフレームワーク
29	FTP BASE64	commons-codec-1.6.jar commons-logging-1.1.1.jar commons-net-3.1.jar	ApacheCommons プロジェクトで配布されているエンコーダ・デコーダ

② Tomcat endorsed 設定 API

opengion/apps/tomcat7.0.27/endorsed フォルダに各種 API をコピーしておくことで、パス設定などを行うことなく利用できます。

No	分類	ファイル名	機能
30	Tomcat	jaxp-api.jar jaxp-ri.jar	Java で XML を扱うための API のひとつ。XML 文書の妥当性検証や構文解析のためのインタフェースを提供する

③ WEB-INF lib 設定 API

opengion/uap/Webapps/gf/WEB-INF/lib フォルダに各種 API をコピーしておくことで、パス設定などを行うことなく利用できます。

No	分類	ファイル名	機能
31	jstl	jstl.jar standard.jar	jakarta プロジェクトで配布されている JSP のスタンダードタグライブラリ

6. インストール手順

Web アプリケーションをインストールする手順を以下に示します。

① インストール前に決定しておく必要のある情報

インストール前に下記の事前情報を調査、決定しておいてください。

・インストール先のOS	例) Windows 2008
・ドライブ名	例) C: ※ OS 領域と分けても可
・最大使用メモリ	例) -Xmx128m
・初期使用メモリ	例) -Xms64m
・接続先DB情報	例) jdbc:oracle:thin:@localhost:1521:ORCL

② アプリケーションのインストール

opengion 以下、apps フォルダ、uap フォルダなどをコピーします。

apps フォルダは、アプリケーション (Java, Tomcat) で、uap は、サンプルアプリケーションになります。業務アプリケーションは、opengion/uap/Webapps/ge を参考に作成します。

③ opengion/uap/bin/init.bat の書き換え (DB 設定、メモリ設定)

データベースの接続先、ユーザー、パスワードや、Tomcat 起動ポート、Java 起動時のメモリサイズなどを、書き換えます。

```
set REALM_URL=jdbc:oracle:thin:@localhost:1521:XE
set REALM_NAME=GF ← DB 接続ユーザー
set REALM_PASSWORD=GF ← DB 接続パスワード

set CONNECTOR_PORT=8824 ← Tomcat アクセスポート

set JAVA_OPTS=-Xms64m -Xmx512m -Xss256k -XX:PermSize=32m
-XX:MaxPermSize=128m
```

④ opengion/apps/tomcat7.0.27/conf/server.xml の書き換え

標準で使う場合は、そのまま使える設定になっています。

書き換える必要があるとすると、ログイン認証 (Realm 設定) のテーブル名ですが、標準では、GEA10V01 ビューを定義していますので、各社のユーザーテーブルに対応したビューを作成すれば、変更の必要はなくなります。

```
<Realm className="org.apache.catalina.realm.JDBCRealm"
.....
userTable="GEA10V01" userNameCol="USERID"
userCredCol="PASSWD"
userRoleTable="GEA10V01" roleNameCol="SYSTEM_ID"
/>
```

⑤ opengion/apps/tomcat7.0.27/conf/tomcat/localhost/gf.xml の書き換え

gf は、opengion/uap/Webapps/gf/ 以下のアプリケーションの設定です。これは、各アプリケーションを配備する都度、作成します。
現時点では、自動配備ではなく、手動で配備することになっています。

⑥ Tomcat 実行

opengion/uap/bin/startup.bat をダブルクリックすれば、TOMCAT が起動します。
opengion/uap/bin/shutdown.bat で、停止することを確認しておいてください。

なお、startup.bat を実行すると、内部で shutdown.bat を call しています。
(startup.bat は、再起動:リスタート も兼ねています。)
そのため、初めて Tomcat を起動する場合は、下記のエラーが発生しますが問題ありません。

```
Catalina.stop: java.net.ConnectException: Connection refused: connect
java.net.ConnectException: Connection refused: connect
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    .....
```

⑦ ブラウザでの動作確認

http://サーバー名:8824/プロジェクトID/JSP/index.JSP でアクセスして、起動すれば、正常です。

初期設定時のログインユーザー/パスワードは、admin/admin です。

第II部 Webアプリケーション構造

『何かを学ぶためには、自分で体験する以上にいい方法はない』
Albert Einstein (アルベルト・アインシュタイン)

ここでは、Web アプリケーションの構造について説明します。
構成は次のとおりです。

第2章 ユーザーインターフェース(業務ロジックA)
JSP を用いてユーザーインターフェースを構築する場合の考え方について説明します。

第3章 ビジネスロジック(業務ロジックB)
業務に特化した機能をPL/SQLを用いてビジネスロジックを構築する場合の考え方について説明します。

第4章 エンジンコア(汎用オブジェクト)
エンジンコア、特にカスタムタグと汎用オブジェクトについて、説明します。

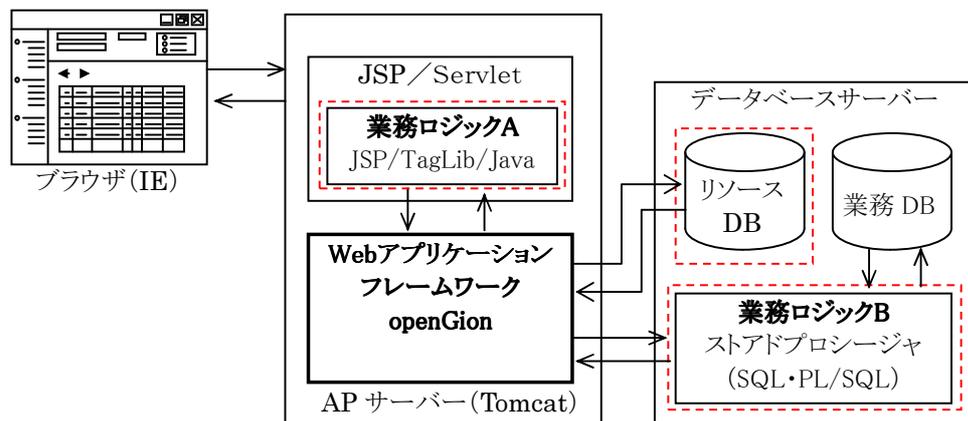
openGion
openGion
フレームワーク

第2章 ユーザーインターフェース(業務ロジックA)

この章では、JSP を用いてユーザーインターフェースを構築する場合の考え方について説明します。

Web アプリケーションは、3つの階層で構築されています。

- 業務ロジック A (JSP,SQL 等の画面系)
- 業務ロジック B (PL/SQL でのバッチ系)
- フレームワーク (Java によるエンジン機能)



1. 業務ロジック A

JSP のタグライブラリを利用して、画面を作成します。画面遷移や、SQL、ストアドプロシージャ呼び出し等の機能も、タグライブラリを通して実行できます。この業務ロジックは、ユーザーインターフェース部に相当しますので、簡単に開発/修正できることを特長にしています。

2. 業務ロジック B

オラクルのストアドプロシージャ/ストアドファンクションによる業務ロジックを記述します。主として複雑なバッチ処理的な業務ロジックを記述し、業務ロジック A の JSP より、CALL で呼び出して利用します。この業務ロジックは、システムの基幹処理を中心に通常ユーザーインターフェースに影響されないような業務を記述します。共通的な処理を PL/SQL で記述することで、画面系からのリアルタイムな使用だけでなく、バッチ的な処理にも利用可能です。

3. Web アプリケーションフレームワーク

先の2つと異なり、システム開発者が作成するのではなく、エンジンチームと呼ばれているメンバーで、業務ロジックとは切り離されて開発/運用されています。この部分は、拡張性やメンテナンス性を重視したオブジェクト指向言語 Java で開発されています。

第2章 ユーザーインターフェース(業務ロジックA)

エンジンと業務ロジックAとを切り分けるインターフェースの役割を行うのが、リソースファイルです。当フレームワークを利用して開発する場合、各種リソースが必要になります。

リソースとは、画面、項目、メッセージ、項目選択肢、ユーザーなどの基本情報と、それに付随するアクセス制限、言語(国際化対応)、表示形式、データ論理属性など、プログラミング時にアプリケーション全般に共通に使用される情報を、ここですべて管理し、ノン・コーディングで利用できるようにします。

これらのリソースを作成するにあたり、非常に重要なのがデータベース設計です。データベース設計をするためのデータベースをあらかじめ準備しておき、その情報よりリソースデータの自動作成を行うことで、作業効率の向上を図っています。

これらのリソースファイルを基本設計時に用意しておく必要があります。この情報を用いて、画面サンプル(納入仕様書)の作成が可能になります。(物理データベースは不要。リソースのみで動作させることができます。)

リソースファイルについては、『第8章 Web アプリケーション・リソースの管理』にて、再度取り上げたいと思います。

第3章 ビジネスロジック(業務ロジックB)

この章では、業務に特化した機能を PL/SQL を用いてビジネスロジックを構築する場合の考え方について説明します。

Web アプリケーションの考え方として、高品質、短工数、短納期 つまり、QCD を高めるという目標以外に、短変更、多流用、省管理 を挙げています。

先の業務ロジック A といわれる JSP 部分は、短変更という目標を担っています。つまり、お客様からの要望事項に合わせてすぐに変更できるためには、JSP でお手軽に開発/修正できる必要があります。

反面、この業務ロジック B では、多流用 の目標を担っているといえます。

ビジネスロジックといえども、画面や新機能は年々変化したり、担当者ごとに操作方法やその目的が異なることがあります。しかし、その企業で脈々と行われてきた処理(基幹処理)は、そう簡単には変わりません。

PL/SQL により、旧来のビジネスロジックを構築したり、コアの部分を構築しておき、付加項目については、JSP 等で検索できる様にする事が可能なため、業務ロジックの流用を促進する事が可能になります。

第4章 エンジンコア(汎用オブジェクト)

この章では、エンジンコア、特にカスタムタグと汎用オブジェクトについて、説明します。

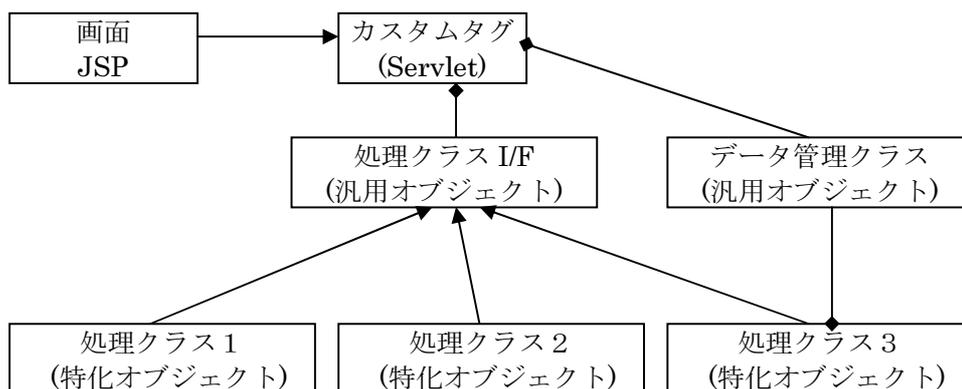
エンジンコア部は、JSP のカスタムタグと、汎用オブジェクトから成り立っています。JSP 画面には、カスタムタグで記述します。そのカスタムタグは、いわば、画面とエンジンコア部のインターフェースになっており、エンジンコア部を修正したとしても、JSP 画面には何ら影響を及ぼしません。

また、カスタムタグは、XML 形式のため、属性の追加時でも、既存のカスタムタグを修正することなく、(つまり、上位互換を保ったまま) エンジンコア部分を進化させることが可能になります。

カスタムタグ自身の機能を、以下に示します。

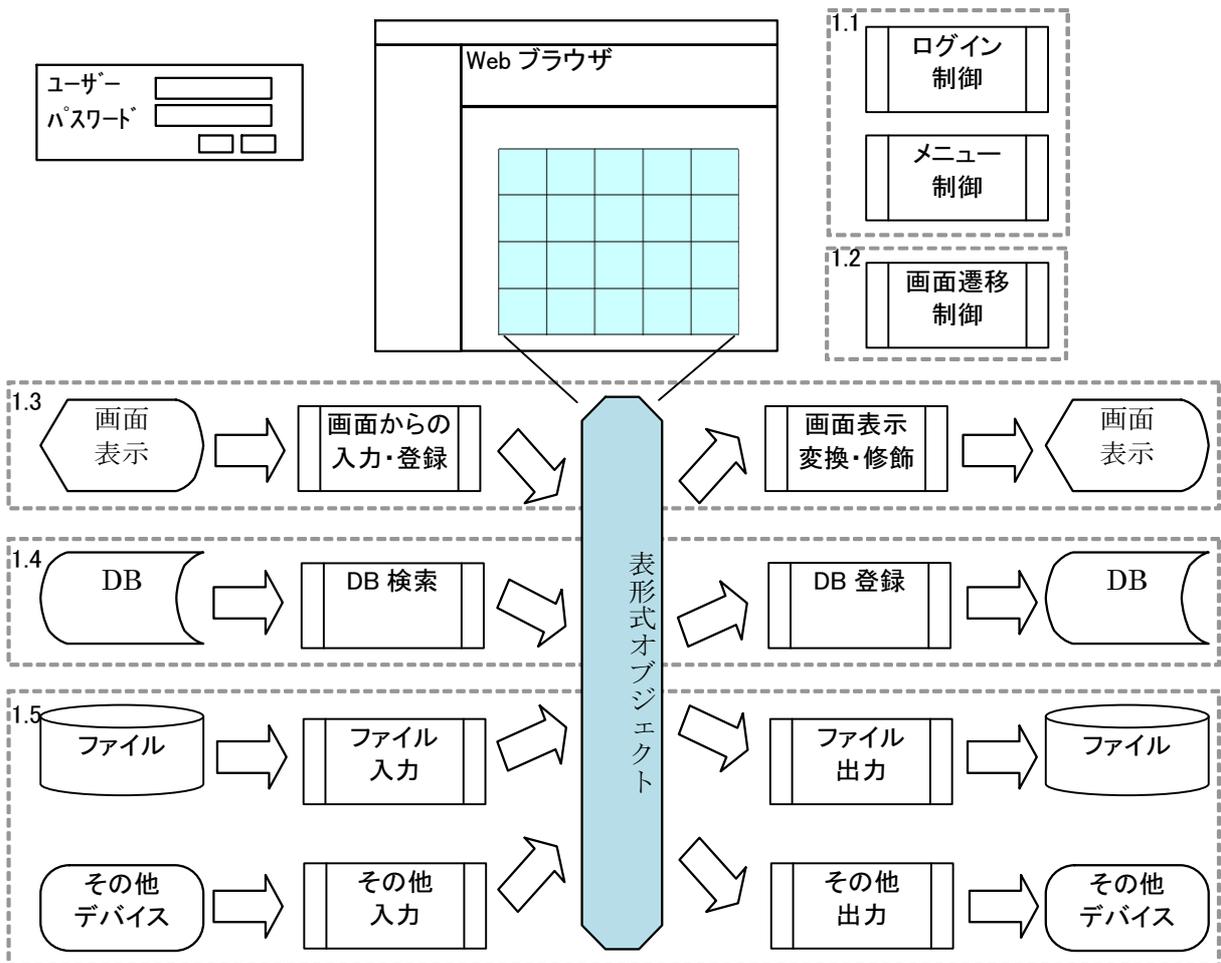
- ・ JSP 画面からの設定情報の取得
- ・ 設定情報に応じた汎用オブジェクトの作成/取得
- ・ 汎用オブジェクトへ設定情報を転送
- ・ 汎用オブジェクトより、処理結果の受取
- ・ 処理結果の表示/または、設定

多くの場合、カスタムタグは、自分では処理を行いません。JSPやServlet 関係のクラスを一旦汎用オブジェクト(データ管理クラス)に設定し、そこで、汎用オブジェクト(データ処理クラス)で処理を行います。結果も、汎用オブジェクト(データ管理クラス)で受け取りますので、最終表示は、これを処理するオブジェクトへの転送で実現しています。



1. 表形式オブジェクト

通常の検索結果やファイルからの入力、その他デバイス (FTP や LDAP など) からの入力も、すべて共通の「表形式オブジェクト」として管理されます。このオブジェクトは、画面表示や DB 登録、ファイル出力、その他デバイスへと書き出すことが可能です。つまり、各種入力を共通の表形式オブジェクトを介して各種出力することで、相互にデバイス変換を行うことが可能です。

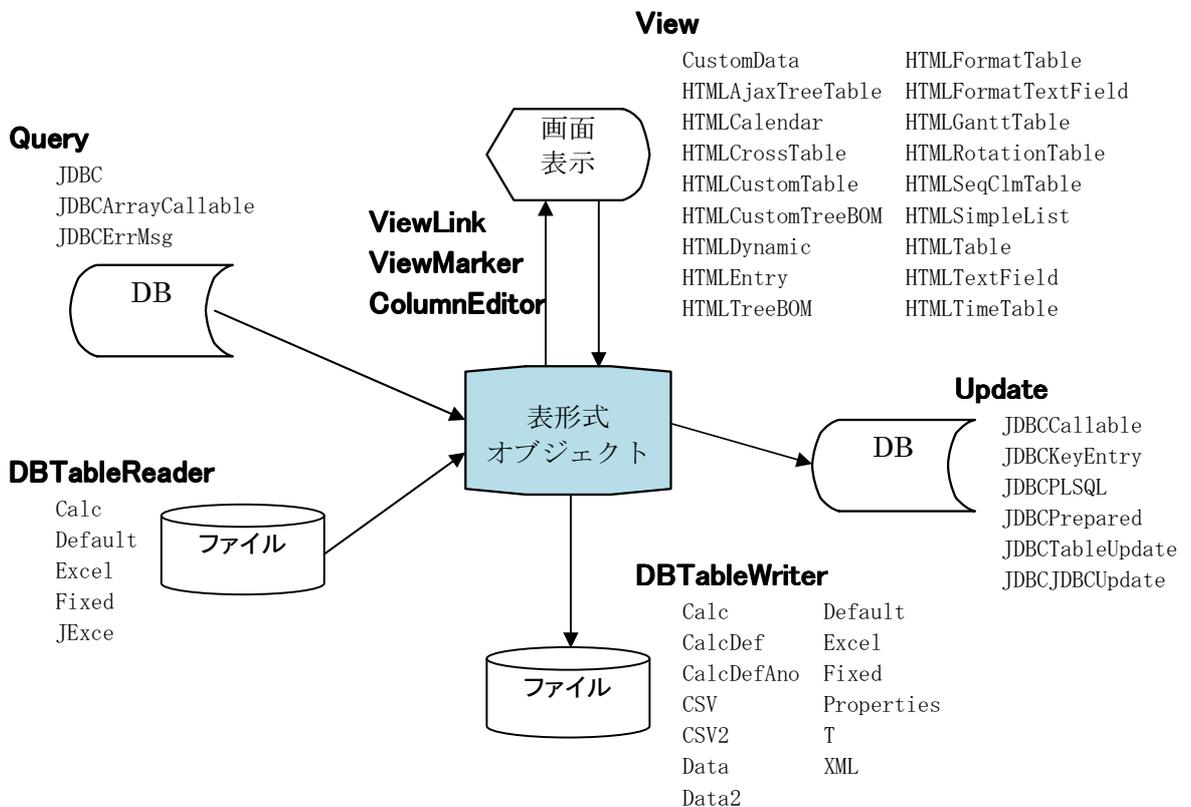


2. 汎用オブジェクト

各種デバイスから、表形式オブジェクトを作成したり、表形式オブジェクトを、各種デバイスに出力するために、デバイス毎にインターフェースが定義されています。このインターフェース（または、その実装クラス）から作成されたオブジェクトを汎用オブジェクトと呼んでいます。

汎用オブジェクトの実態は、各サブクラスで記述されるため、新しいデバイスが必要になった場合でも、サブクラスを記述するだけで、既存のデバイスとの入出力をすぐに利用できます。

標準で、下記のインターフェースとサブクラスが用意されています。

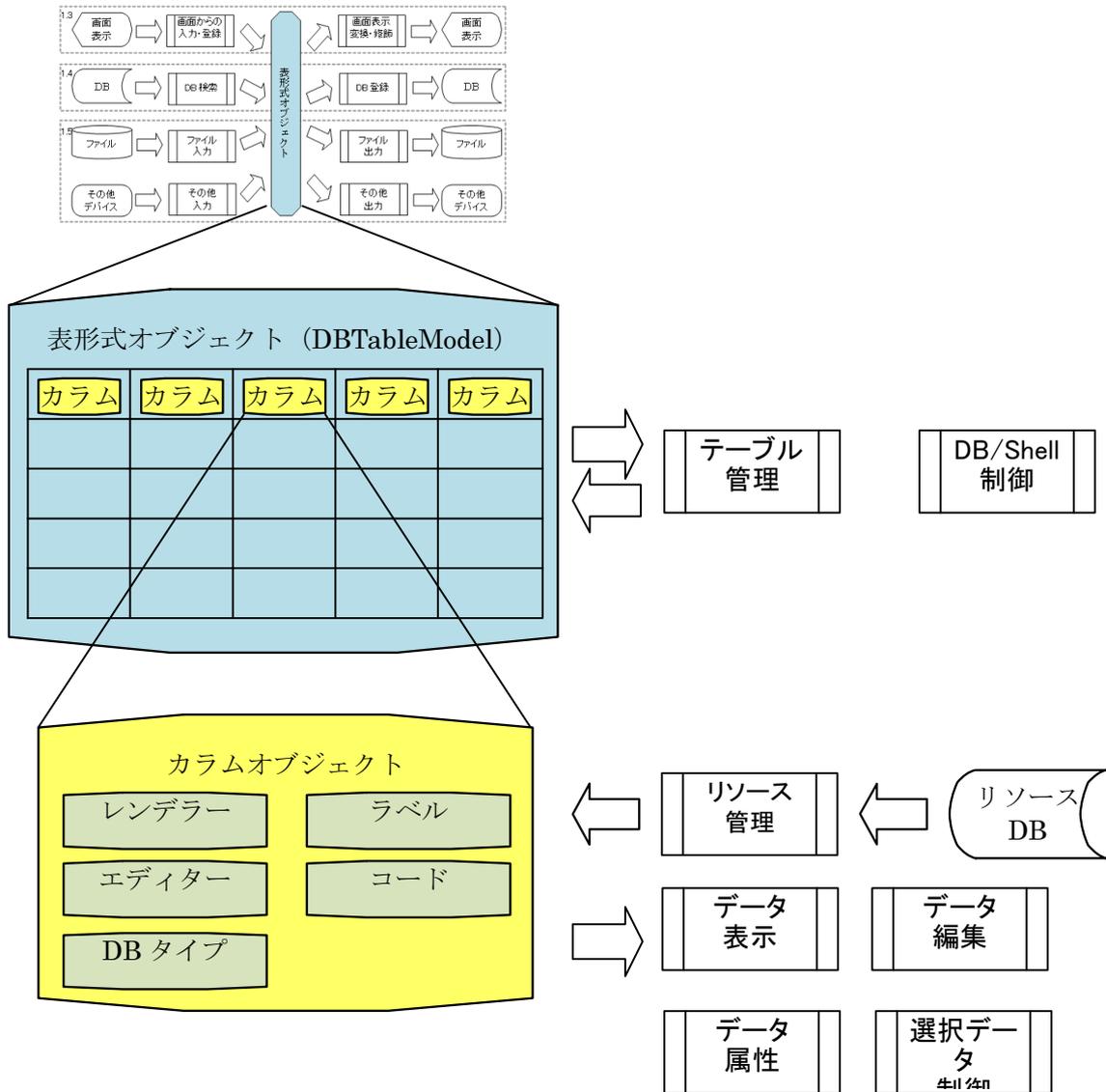


3. カラムオブジェクト

フレームワークの中心が、表形式オブジェクトだとすると、それを構成するオブジェクト群もまた、非常に重要な働きをします。

表形式オブジェクトは、表形式（行と列）を持ち、各行に対して、共通のカラムオブジェクトを持っています。この、カラムオブジェクトが、表形式オブジェクトに格納されたデータをどのように表示（Renderer）し、どのように登録させ（Editor）どのような値を格納するか（DBType）を規定します。

これらの設定値は、カラム ID で一意に決まる必要があり、このカラム ID と、データベースのカラム名を一致させることで、DB 定義からリソースを作成し、リソースからカラムオブジェクトの属性を定義するという流れが作られます。



第 III 部

Webアプリケーション アーキテクチャ

『「偶然」は、準備のできていない人を助けない』
パストゥール

ここでは、Web アプリケーションのアーキテクチャについて説明します。
構成は、次のとおりです。

第5章 Web アプリケーションの起動と停止

Web アプリケーションについて説明し、Web アプリケーション管理者が Web アプリケーションへのアクセスを制御する方法について説明します。

第6章 アプリケーション・アーキテクチャ

アプリケーションの構造とその実行アーキテクチャについて説明します。

第7章 メモリー・アーキテクチャ

Web アプリケーションで使用するメモリー構造について説明します。検索データや、リソース等のキャッシュ方法について説明します。

openGion
openGion
フレームワーク

第5章 Web アプリケーションの起動と停止

この章では、Web アプリケーションについて説明し、Web アプリケーション管理者が Web アプリケーションへのアクセスを制御する方法について説明します。

Tomcat サーバーを使用する場合の方法を以下に示します。



ヒント

【事前準備】

第1章 Web アプリケーションの基礎知識 / 6. インストール手順 にて、すでにアプリケーションのインストールは済ませておいてください。

opengion/uap/bin/ 以下に、init.bat , startup.bat , shutdown.bat , workdelete.bat のファイルが存在していることを確認しておいてください。さらに、init.bat の除く3ファイルのショートカットをデスクトップ等に作成しておく、便利です。

1. startup.bat

Web アプリケーションを起動します。

すでに、起動済みの場合は、それを一旦終了させてから、再起動を行います。

Tomcat の状況によっては、起動済みのアプリケーションが終了しない場合があるため、shutdown.bat を行って、完全に終了した後に再起動する方がベターです。

なお、Web アプリケーションが立上がっていない状態で、startup.bat を実行すると、シシャットダウンできないためのエラーが発生しますが、問題ありません。

2. shutdown.bat

Web アプリケーションを安全にシャットダウンします。

シャットダウン時処理で、データベースとのセッション切断や、統計情報の設定、ユーザー情報の記憶等を行っている場合がありますので、通常は、shutdown.bat の実行により終了させてください。

なお、startup.bat 時の終了後再起動時も、同様に安全です。

3. workdelete.bat

Web アプリケーションを再起動します。

startup.bat との違いは、JSP のコンパイルされたクラスファイルを破棄してから再起動を行います。

通常の Tomcat では、%CATALINA_HOME%\work 以下に、JSP から、JSP クラスを生成し、これをコンパイルして使用します。このクラスを一旦削除することにより、JSP 画面の変更が確実に、Tomcat に反映されます。



コラム JSP のコンパイル

JSP ファイルを修正した場合、Tomcat はその修正を自動認識して、JSP クラスのソースを自動生成し、コンパイルします。

そのため、JSP 画面変更後の初めてのアクセスは、コンパイル時間がかかるため、通常より遅くなります。

Tomcat は、この JSP クラスと JSP 画面のタイムスタンプとを比較して、JSP クラスが JSP 画面よりも新しい場合は、再度ソース作成⇒コンパイルを行います。

ただし、このタイムスタンプの比較では、

- ・ JSP 画面にインクルードされている画面の修正分の自動判断。
- ・ JSP 画面の編集端末(クライアント)と JSP クラス作成端末(サーバー)間でのシステム時刻の違いによるタイミングのずれ。

が、問題になります。

JSP のインクルード画面を修正した場合は、まったく反映されないため、比較的容易に気付きます。通常はインクルードされるファイルは、共有されている場合が多いので、workdelete.bat にて、ワークを削除した方が無難です。

サーバーとクライアントのシステム時刻がずれている場合は、非常にわかりにくい現象になります。

サーバーが、仮に2分進んでいる場合、JSP 画面を修正したあと、1回目は、すぐに変更が反映されますが、その直後(例えば2分以内)に変更しても、サーバー側が2分進んでいるため、JSP ソースとの差が(クライアント側の方が古いため)コンパイルされません。セーブできていないと思い込み、何度かセーブを繰り返しているうちに、2分が経過して、反映されます。

このケースで、毎回、workdelete.bat で反映させていたのでは時間の無駄になるため、サーバーとクライアントの時刻は、合わせておく必要があります。

そのような時刻設定サーバーを使用できない環境では、クライアント側を進めておくくらいの方が無難かもしれません。

第6章 アプリケーション・アーキテクチャ

この章では、アプリケーションの構造とその実行アーキテクチャについて説明します。

JSP 画面は、大きく、メニュー画面、QUERY 画面、RESULT 画面に分かれています。

メニュー画面:

メニューを表示します。

基本的には、このメニューをクリックすると、右側のコンテキスト画面が呼ばれます。

コンテキスト画面:

各機能ごとのフォルダは、このコンテキスト内で独自にフレーム分割を行います。

通常は QUERY 画面、RESULT 画面をフレーム分割して使用します。

QUERY 画面:

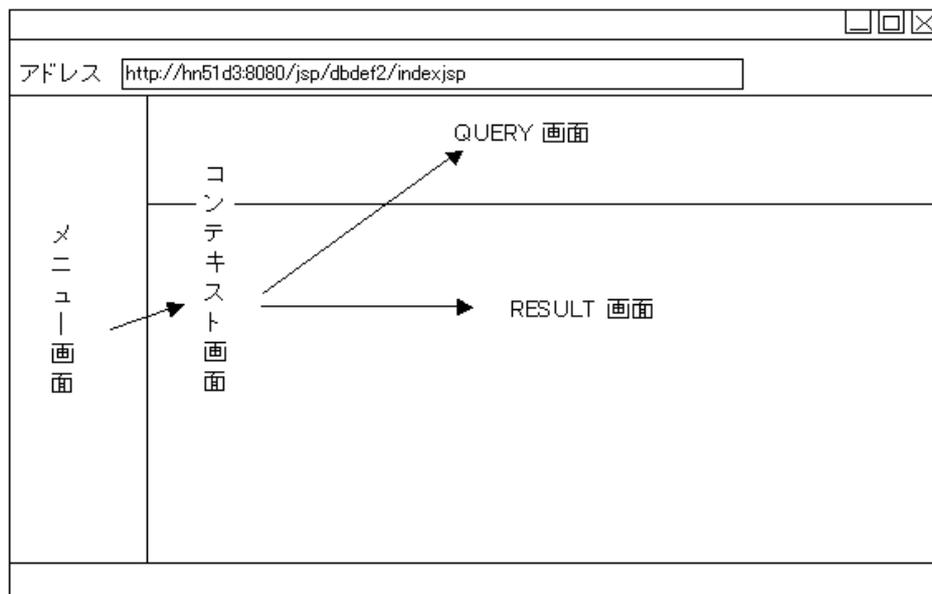
検索条件を登録したり、表示順を指定しています。

この画面の『検索』ボタンで、RESULT 画面が呼ばれる。

RESULT 画面:

検索条件を受け取り、実際に検索／表示します。

この画面から、登録処理メニューを選ぶと、同じターゲットに登録画面が現れます。



1. 画面制御

① コピー画面

コピー処理は検索画面で条件を入力(または、何らかの業務ロジックを実行)して、その結果を RESULT 画面に結果を出力します。検索結果の画面より、新規追加/変更/削除/コピー を選ぶ事により、それぞれの処理を行います。

ユーザーがその画面の登録権限が無い場合は、編集ボタンは現れません。

論理キーの重複チェックは行っていないので、各業務アプリケーション側で対応する必要があります。

□ □ ×

アドレス

ログイン
[ログイン](#)

設計変更
[品番採番](#)

構成変更
[図番変更](#)

特殊指示
[作番処理](#)

図面管理
[CAD抽出](#)

[PDF作成](#)

機種

親品番

ソート

親.子
 図番

検索

	親品番	子品番	数量	単位	図番	備考
<input type="checkbox"/>	008-10000-10	008-10010-60	1	P	0082000-001	フレーム
<input checked="" type="checkbox"/>	008-10000-10	008-10020-60	2	P	0083000-001	
<input checked="" type="checkbox"/>	008-10000-10	008-10030-60	1	P	0083000-002	
<input checked="" type="checkbox"/>	008-10000-10	008-10040-60	1	P	0083000-003	
<input type="checkbox"/>	008-10000-10	008-20000-60	1	P	0084000-001	

□ □ ×

アドレス

ログイン
[ログイン](#)

設計変更
[品番採番](#)

構成変更
[図番変更](#)

特殊指示
[作番処理](#)

図面管理
[CAD抽出](#)

[PDF作成](#)

機種

親品番

ソート

親.子
 図番

検索

	親品番	子品番	数量	単位	図番	備考
<input type="checkbox"/>	008-10000-10	008-10010-60	1	P	0082000-001	フレーム
<input checked="" type="checkbox"/>	008-10000-10	008-10010-60	1	P		
<input type="checkbox"/>	008-10000-10	008-10020-60	2	P	0083000-001	
<input type="checkbox"/>	008-10000-10	008-10030-60	1	P	0083000-002	
<input type="checkbox"/>	008-10000-10	008-10040-60	1	P	0083000-003	
<input type="checkbox"/>	008-10000-10	008-20000-60	1	P	0084000-001	

36

② 変更画面

変更ボタンは、その項目の変更を行います。

なお、ここでの変更は、物理的なフレームワーク上のユニークキーに対しての変更であり、論理的なキー（親子など）に対しては、各業務ロジックで作り込む必要が有ります。例えば、自動的に開始／終了でAデータとDデータを作成するとか、子品番の項目を書き換えられない様に、チェックする（テキストフィールドにしない）などです。

③ 削除画面

削除処理は、選ばれた項目をそのまま変更しないで、削除します。

削除する場合でも、物理削除と論理削除がありますので、それぞれの処理は、書く業務ロジックで作りこむ必要があります。

④ 新規追加画面

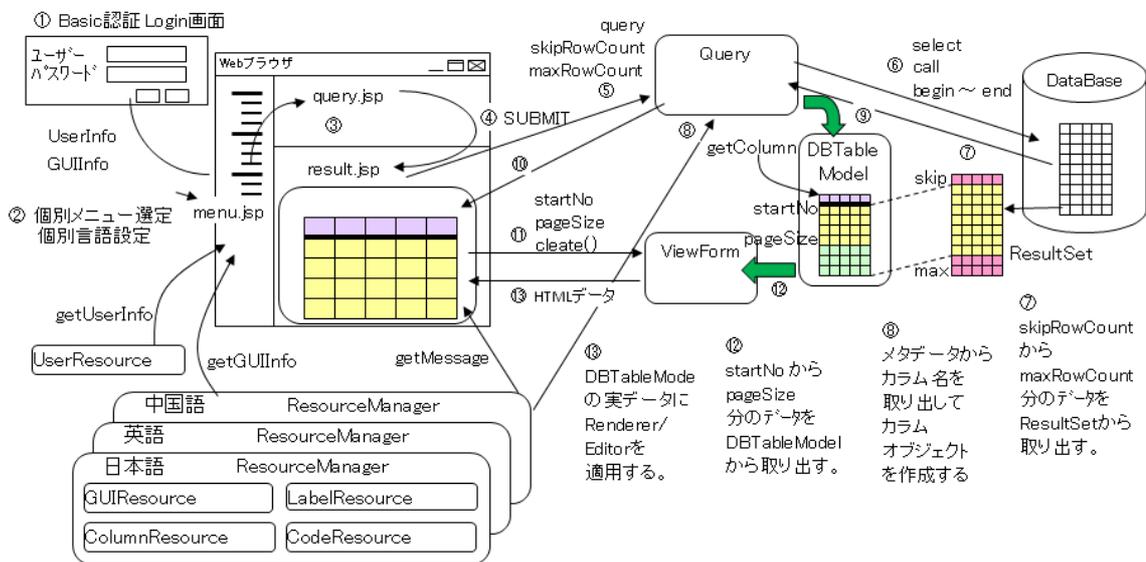
基本的には、コピー画面と同様の動きをします。

違いは、行選択時のカラムデータが、コピーは、選択された行データがセットされ、新規は、空白で行データが作成されます。

カラムセットタグのデフォルト値が異なると考えてかまいません。

第7章 メモリー・アーキテクチャ

この章では、Web アプリケーションで使用するメモリー構造について説明します。検索データや、リソース等のキャッシュ方法について説明します。



まず、通常の Web アプリケーションのデータの流れを以下に示します。

- ① BASIC 認証により Login 画面を表示(ユーザー情報取得)
- ② ユーザー毎にアクセスできる画面を、指定言語で表示する。
- ③ query.JSP より検索条件等を入力する。
- ④ SUBMIT することにより、result.JSP を呼び出す。以下、result.JSP 内部の処理
- ⑤ Query クラスを作成する。検索条件と 取り込み開始ポイント、最大取り込み可能数をセットする。
- ⑥ jdbc接続により、DataBase を検索する。(DB 以外にフラットファイル、XML,CSV ファイル等の取り込みも可能)
- ⑦ 検索結果を元に、内部に DBTableModel を生成させる。
- ⑧ ResourceManager を用いて,DBTableModel のラベル、コードを lang により指定された言語に変換する。
- ⑨ Query より DBTableModel を取り出し、session に登録する。
- ⑩ result.JSP に制御が戻る。

- ⑪ result.JSP から、ViewForm 呼び出す。(startNo,PageSize,lang)
- ⑫ session から、DBTableModel を取り出し、ViewForm にセットする。
- ⑬ 変換結果の HTML 文字列を返す。

1. データベース検索結果に関するメモリ管理

データベースから検索したデータは、DBTableModel オブジェクトに格納されます。これは、セッション情報に識別子ごとにメモリに格納されます。通常、1ユーザーごとに1セッションが割り当てられ、かつ、識別子に初期値を使用すると、1ユーザーごとに、1つの DBTableModel を検索ごとに作成して置き換えていることになります。また、このメモリ情報は、どの画面で検索した結果でもいつでも取り出すことができます。

不要なメモリを残さないためには、識別子は、デフォルトのみとするか、または、別に使用した場合は、すぐに破棄するようしなければなりません。また、DBTableModel にセットするデータ量も制限することで、多くのログインユーザーがいてもメモリ不足にならないようにする必要があります。

以下に query タグのメモリ制御に関係する属性を説明します。

- tableId
メモリに格納する場合の識別子です。通常はデフォルト(指定しない)を使用します。
- scope
セッションやリクエストなど、どのスコープに保存するか指定します。通常はデフォルト(指定しない)ですが、検索のみなど、tableId を指定して登録系と別に結果を表示させたい場合に利用できます。ただし、リクエストスコープに登録した場合は、データ抜き出しにより取り出すことはできません。
- maxRowCount
検索時の最大行数を指定します。不適切な検索絞り込み条件によって、大量の検索結果によりメモリ不足を引き起こさないために、最大件数を制限できます。デフォルトは、SystemResource.properties の DB_MAX_ROW_COUNT で指定できます。

2. リソース情報のメモリ管理

リソース情報とは、ユーザー、画面、ラベル、カラム、コードなどの情報で、各リソースオブジェクトとしてキャッシュ、利用しています。

国際化対応(日本語、英語、中国語等)は、ラベルにのみ言語ごとにリソースを持っており、ログインユーザーの言語設定に応じて動的にリソースオブジェクトが構築されます。

これらのリソースオブジェクトは、セッション情報ではなく、アプリケーションごとに共通で使用しますので、ログインユーザー人数対して、メモリ使用量は、大きく変わりません。

ただし、画面リソースは、ログインユーザー単位の、あらかじめアクセス制限を加味した状態で、利用するオブジェクトの Map を用意することで、処理時間の短縮を図っていますので、画面リソースのポインタのメモリ容量は、人数に応じて加算されます。

リソース情報については、『第8章 Web アプリケーション・リソースの管理』で、詳細を説明します。

3. Tomcat 起動時の Java オプション

Tomcat 起動時に、Java オプションを指定します。

このオプションで、Java 実行時のメモリ割り当てを指定できます。

オプションは、`opengion/uap/bin/init.bat` に、記述しています、`JAVA_OPTS` で指定します。

例)

```
set JAVA_OPTS = -Xms64m -Xmx512m -Xss256k -XX:PermSize=32m -XX:MaxPermSize=128m
```

`-Xmsn`

メモリ割り当てプールの初期サイズをバイト数で指定します。

指定する値は、1M バイトより大きい 1024 の倍数にしなければなりません。

キロバイトを指定するには、文字 `k` または `K` を付けます。メガバイトを指定するには、文字 `m` または `M` を付けます。既定値は 2M バイトです。次に例を示します。

```
-Xms6291456
```

```
-Xms6144k
```

```
-Xms6m
```

`-Xmxn`

メモリ割り当てプールの最大サイズをバイト数で指定します。

指定する値は、2M バイトより大きい 1024 の倍数にしなければなりません。

キロバイトを指定するには、文字 `k` または `K` を付けます。メガバイトを指定するには、文字 `m` または `M` を付けます。既定値は 64M バイトです。次に例を示します。

```
-Xmx83886080
```

```
-Xmx81920k
```

```
-Xmx80m
```

`-XX:PermSize -XX:MaxPermSize`

JVM のヒープ領域を指定します。ここにはクラス定義やメソッド、フィールドなどのメタデータが格納されますので、大量のクラス等をロードする JSP/Servlet 等では、初期値のままでは不足するケースがあります。

```
-XX:PermSize=32m
```

```
-XX:MaxPermSize=128m
```

第 IV 部 データ制御オブジェクト

『この素晴らしい応用科学は労働を軽減し、生活をより豊かにしながら、
なぜ我々に幸福をもたらしてくれないのか。答えは簡単である。
我々がそれを有意義に利用するにいたっていないからである』
Albert Einstein (アルベルト・アインシュタイン)

ここでは、データの制御について説明します。
構成は、次のとおりです。

第 8 章 リソース情報管理
Web アプリケーションで必要なリソースの管理方法について説明します。

第 9 章 メッセージの使用方法
エラー・メッセージに関する一般情報および補足的ヒントを紹介します。

第 10 章 カラムオブジェクト
データベースのカラム属性に対するオブジェクトについて、説明します。

第 11 章 コードリソースとカラムオブジェクト
カラムオブジェクトを修飾する機能として、コードリソースがあります。コードリソースとカラムオブジェクトの関係を説明します。

第 12 章 レンダラー、エディター、DBタイプ
カラムオブジェクトに対して、データを適用するときに、表示(レンダラー)、編集(エディター)、物理特性(DBタイプ)の取扱を説明します。

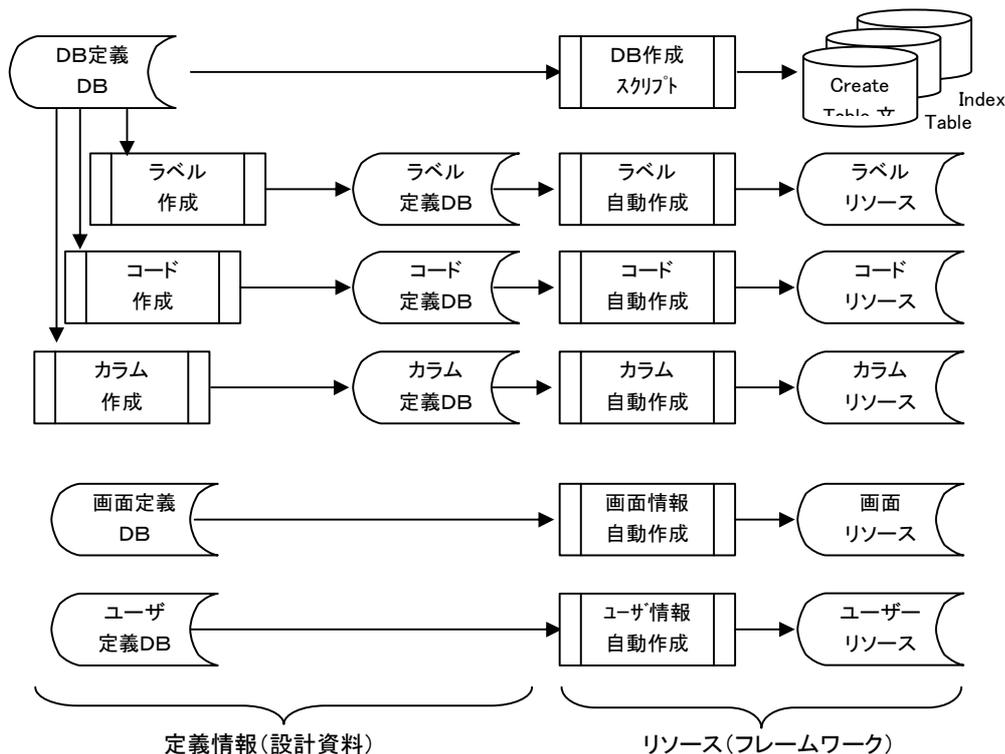
第 13 章 その他リソースファイル
ラベル、メッセージ、ユーザー、画面、の各リソースについて説明します。

第8章 リソース情報管理

この章では、Web アプリケーションに必要なリソースを制御する方法について説明します。エラーメッセージを説明するに当たり、リソースの管理方法を理解しておくことは重要です。

1. リソース管理

当フレームワークを利用して開発する場合、各種リソースが必要になります。リソースとは、画面、項目(カラム)、ラベル、項目選択肢(コード)、ユーザーなどの基本情報と、それに付随するアクセス制限(ロール)、言語(国際化対応)、表示種別(レンダラー)、編集種別(エディター)、データ論理属性(DBタイプ)など、プログラミング時にアプリケーション全般に共通に使用される情報です。当フレームワークでは、それらを一元管理し、ノン・コーディングで利用しています。これらのリソースを作成するにあたり、非常に重要なのがデータベース設計です。データベース設計をするためのデータベースをあらかじめ準備しておき、その情報よりリソースデータの自動作成を行うことで、作業効率の向上を図っています。これらのリソースファイルを基本設計時に用意しておく必要があります。



定義情報(設計資料)

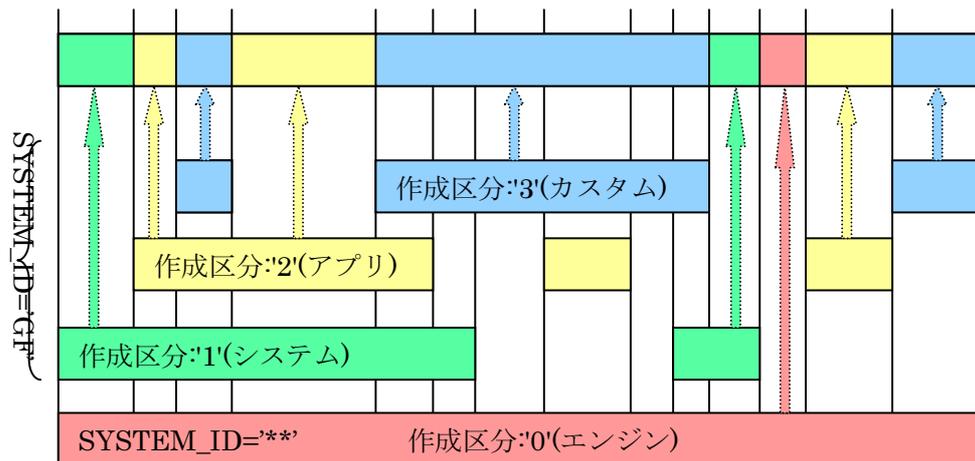
定義情報は、Web フレームワークを用いて開発する場合に、基本設計段階で完成させておく必要がある。

リソース(フレームワーク)

Web アプリケーションは、このリソースを元にすべての共通属性を定義し、アプリケーション自動的に組み込む。

2. リソースの階層管理

フレームワークのリソースは、階層構造を持っています。エラーメッセージを管理しているラベルリソースも同様です。



リソースは、SYSTEM_ID 単位に持ちます。ただし、SYSTEM_ID='**'として、標準リソースが定義されています。なにもしなければ、この標準リソースが使われます。

次の階層は、各 SYSTEM_ID 単位に持つリソースです。各 SYSTEM_ID の内部で、作成区分を持っています。この値が大きいほど、強いリソースという事になります。

先の SYSTEM_ID='**'のリソースの作成区分は、'0' です。

同一 ID のリソースが存在する場合は、作成区分が、'1' → '2' → '3' … と大きくなるほど有効です。

つまり、基本システムに対して、作成区分の大きなリソースを「追加」することで、リソースの変更が可能で、元のリソースに戻したければ、先に追加したリソースを「削除」すればよいことになります。なお、既存に定義されているリソースを削除することはできません。これは、システムそのものが、先に準備されているリソースを利用することを前提に作られている可能性があるためです。

存在していないリソースを追加することは、問題なく可能です。その場合は、アプリケーションの作成が前提です。(でない、追加したリソースを使う箇所が存在しないことになります。)

第9章 メッセージの使用法

この章では、エラー・メッセージに関する一般情報および補足的ヒントを紹介します。

3. メッセージの種類

フレームワークには、あらかじめ、SYSTEM_ID='**' 作成区分='0' のメッセージが組み込まれています。システム起動時に、フレームワークのバージョン番号がアップしていれば、データベースに書き込まれます。

つまり、SYSTEM_ID='**' 作成区分='0' のメッセージを追加、更新したとしても、消えることを意味します。先に述べたように、SYSTEM_ID 毎に作成するか、作成区分を'0'以上にして作成しておく必要があります。

エラーメッセージは、標準メッセージと、エラーメッセージの2種類存在します。

標準メッセージは、MSGxxxx (xxxxは連番) で定義され、エラーメッセージは、ERRxxxx (xxxxは連番) で定義されます。

メッセージのリソースは、ラベルリソースとして管理されるため、カラム名称もメッセージも区別がありません。よって、ID の採番方法で区別することをお勧めします。

(ラベルメッセージの属性に、「ラベル区分」というのがあり、一応データとして区別することは可能ですが、フレームワークの内部処理としては区別していません。)

メッセージを独自に作成する場合は、上記の MSGxxxx や ERRxxxx を使うと、混乱をきたす場合がありますので、登録時の採番ルールを決めてください。

たとえば、SYSTEM_ID + Mxxxx とすると、GFMxxxx や GFExxxx などとできます。

4. 国際化対応

メッセージは、リソース DB で管理されています。

リソース関係で、ユニーク属性に含まれる言語属性(LANG)を持つのは、この、ラベルリソースだけです。ユーザーリソースの LANG 属性は、単なる属性であり、ログイン時の言語を決定するのに利用されるだけです。

国際化する場合は、この、ラベルリソースのデータを言語単位に作成することで、実現できます。

以前のバージョンでは、国際化対応として、リソースを properties ファイルで管理していたため、デフォルト以外と、差分として各国語に翻訳されたメッセージを独自に持つことが出来ましたが、最新のリソース DB では、言語に対する階層化は持っていません。必要であれば、一旦すべてのラベルリソースを移してから、翻訳してください。

5. リソースの取得先

リソースは、データベースで管理されていますが、その取得先は、通常は、システムのデータベースになっています。

ユーザーリソース(GEA10)を除く他のリソース(カラム(GEA03),ラベル(GEA08),コード(GEA04),画面(GEA11),システムパラメータ(GE12),アクセス統計(GE15),ユーザーパラメータ

(GE16),URL 転送(GE17))のデータベースを通常の、DEFAULT_DB_URL 設定値以外の場所にアクセスする場合に、RESOURCE_DBID で指定します。

この値は、システムリソースとして設定できます。

DBID は、あらかじめ、opengion/uap/Webapps/gf/WEB-INF/DBConfig.xml で定義しておいて下さい。

6. メッセージの文法

メッセージリソースは、ラベルリソースとして管理されますが、メッセージとして利用する場合は、特別に引数処理が可能です。これは、java.text.MessageFormat を用いて処理されます。

MessageFormat の、引数には、フォーマットタイプを指定できます。すべての引数は、String (文字列)とみなして処理されます。

引数は、{0} ~ {9} までの数字で指定できます。

通常のタグでは、引数は、指定できませんが、message タグで指定する場合は、10個分フルに指定することが可能です。

PL/SQL からのエラーメッセージは、引数を5個まで指定できます。

引数は、{0} のように、{} で囲み、中に引数の順番を入れます。

例) リソース文字

ERR0013=マスター未登録エラー。キー {0} は、{1} に存在していません。

ERR0014=マスター登録済みエラー。キー {0} は、{1} にすでに登録済みです。

ERR0015=データがありませんでした。キー {0} で {1} しましたが、0件でした。

ERR0016={0} エラー。キー {1} で {2} しましたが、{3} のため、{4} でした。

ERR0017=選択エラー。選択行数({0} 件)が、制限値({1} 件)以上選ばれました。

ERR0018=選択エラー。選択行数({0} 件)が、制限値({1} 件)以下選ばれました。

7. メッセージの表示

メッセージ(ラベル)を JSP 画面上で利用するには、2種類の方法が用意されています。

1つ目は、{@LBL.XXXX} と記述することです。これは、ラベルリソースの ID を、XXXX 部分に記述することで、タグリブの BODY 部で解析され、ラベルが表示されます。

この記述では、ラベルリソースのタイプ (Label,Short,Tips,Description,RawShortLabel)は指定できますが、引数は指定できません。

例:

```
{@LBL.XXXX}
```

XXXX にラベル ID を指定します。

```
{@LBL.XXXX %Y}
```

%Y の Y 部分に、ラベルリソースのタイプ Label, Short, Tips, Description, RawShortLabel を指定します。%Y が指定されない場合、つまり、先の例の標準ケースでは、Label が指定されたのと同じになります。

この、%Y は、先頭1文字しか見ていませんので、頭文字('L','S','T','D','R')の指定でも構いません。

第9章 メッセージの使用方法

{@LBL.XXXX @ZZ }

%Y の記述で、リクエスト引数を指定できます。@ZZ の箇所、ZZ というリクエスト変数に、'S' と渡すことで、Short タイプを指定できます。

{@LBL.@XXXX}

ラベル ID の指定方法として、リクエスト引数を渡すことができます。@XXXX の箇所がそれで、XXXX がリクエストのキーになります。これらは組み合わせて使用することも可能です。

2つ目は、message タグを使用する方法です。こちらは、ラベルリソースのタイプも、引数の指定も可能です。

例:

```
<og:message
  lbl      = "ラベル ID"      ラベルリソースのラベル ID
  language = "[ja|en|zh]"    タグ内部で使用する言語コード
  command  = "{@command}"    コマンド (INSERT, COPY, MODIFY, DELETE)
  comment  = "コメント"      コメント
  type     = "Short"        タイプを (Label, Short, Tips, Description) 指定
  val0     = "AAA"          メッセージの引数 {0}
  ~
  val9     = "BBB"          メッセージの引数 {9}
  caseKey  = "{@ckey}"      このタグ自体を利用するかどうかの条件キー
  caseVal  = "{@cval}"      このタグ自体を利用するかどうかの条件値
/>
```

8. PL/SQL のエラーメッセージ

PL/SQL で検索/登録する場合に、実行時エラーやエラーチェックでエラーメッセージを返す場合、エラーコードと引数を使用します。

例)

```
CREATE OR REPLACE PACKAGE INSERT_DB01_PKG AS
  PROCEDURE INSERT_DB01 (
    P_KEKKA          OUT    NUMBER,
    P_ERRMSGs        OUT    ERR_MSG_ARRAY,
    P_NAMES          IN     VARCHAR2,
    P_SYSARGS        IN     SYSARG_ARRAY,
    P_DB01ARG        IN     DB01ARG_ARRAY );
END;
```

P_KEKKA は、その PL/SQL 全体を通しての結果で、以下のコードを返します。

記号	コード	説明
OK	0	正常な場合
WARNING	1	警告。処理は正常終了します。

第 IV 部 データ制御オブジェクト

NG	2	エラー。ロールバックします。
EXCEPTION	8	エラー(拡張用予約)
ORCL_ERR	9	エラー(拡張用予約)

※ 記号は、エラーコードに対応するメニューです。(内部記号)

P_ERRMSGs は、別途定義済みの ERR_MSG_ARRAY です。
このARRAY に値を設定する為に、SET_ERRMSGs PROCEDURE を用意しています。

```
PROCEDURE SET_ERRMSGs
(P_ERRMSGs IN OUT ERR_MSG_ARRAY,
 P_NO      IN      NUMBER := NULL,
 P_KEKKA  IN      NUMBER := NULL,
 P_ID     IN      VARCHAR2 := NULL,
 P_MSG1   IN      VARCHAR2 := NULL,
 P_MSG2   IN      VARCHAR2 := NULL,
 P_MSG3   IN      VARCHAR2 := NULL,
 P_MSG4   IN      VARCHAR2 := NULL,
 P_MSG5   IN      VARCHAR2 := NULL );
```

引数	入出力	説明
P_ERRMSGs	IN OUT	エラーメッセージ配列
P_NO	IN	行番号
P_KEKKA	IN	このメッセージに対する結果(エラーコード)です。このARRAYは、複数のメッセージを登録できる為、エラーやワーニングや正常な場合のメッセージも個別に登録できます。
P_ID	IN	エラーメッセージID
P_MSG1	IN	メッセージパラメータ1 {#ラベル ID} を使用するとそのラベルIDに対応する言語に応じたリソースが使用されます。
P_MSG2	IN	メッセージパラメータ2(同上)
P_MSG3	IN	メッセージパラメータ3(同上)
P_MSG4	IN	メッセージパラメータ4(同上)
P_MSG5	IN	メッセージパラメータ5(同上)

第10章 カラムオブジェクト

この章では、データベースのカラム属性に対するオブジェクトについて、説明します。

本、フレームワークで最も重要な考えが、このカラムオブジェクトです。

通常、データベースから検索した結果は、文字列(VARCHAR2)や数字(NUMBER)などのプリミティブ型情報しか持っていません。もちろん、桁数などの情報は持っていますが、それ以上の情報(例えば、全角/半角や、大文字/小文字、その他ユーザーカスタマイズ情報)をコントロールすることはできません。

カラムオブジェクトとは、カラム名に対応するオブジェクトを割り当て、検索結果に適用させることでオブジェクト的な振る舞いを検索結果に持たせることを可能にするオブジェクトのことです。

SQL文(SELECT PN,CDK,NM FROM RK08) や、ColumnTag の name 属性の『CDK』というカラム名より、Renderer、Editor、DBType という 汎用オブジェクトインターフェース の組み合わせで構成されます。

カラムオブジェクトのラベルを使用。
これは、ラベルリソースより、構築される。

入力フィールドは、カラムオブジェクトの入力属性により指定され、その属性がコードリソースを使用する場合は、コードリソースを利用してプルダウンメニューを作成する。

品番	<input type="text"/>	工場コード	<input type="text" value="▼"/>	検索
品番	工場 C	品名		
D40A	O:大分	パネル		
D40B	O:大分	プリンタ		
D40C	A:海外	電源		
Z06X	K:加賀	フレーム		
Z06Y	S:犬山	ブラケット		

ColumnTag の name 属性に『CDK』を指定する。全体は、カラムオブジェクトにより作成される

QueryTag の SELECT 文に、『CDK』を指定する。表示は、ViewFormTag にて『CDK』をキーに表示する。

カラムオブジェクトで指定できる属性は、以下のとおりです。

- カラム名
- データの属性
- 使用桁数
- 表示桁数
- 表示種別
- 編集種別
- 文字種別
- データのデフォルト値
- ラベルカラム
- コードカラム
- カラムパラメータ
- 表示パラメータ
- 編集パラメータ
- 文字パラメータ
- カラムロール

以下に、主要な属性について、説明します。

•カラム名

カラムオブジェクトのキーになります。

カラムオブジェクトは、検索結果のカラム名をキーにオブジェクトを構築します。

また、カラム名は、データベースのテーブルにまたがって、同一名称が使われることがあります。逆にいえば、カラムオブジェクトを決定するカラム定義は、1システム(Web アプリケーション)を通して、ユニークである必要があります。テーブル間にまたがる場合でも、同一カラム名は、同一カラムオブジェクトに割り当てられます。

•データの属性

VARCHAR2 や NUMBER といった、カラムのデータベース属性を指定します。

•使用桁数

カラムの桁数を指定します。(文字数ではなく、バイト数です。)

使用桁数は、そのカラムとして使用できる最大桁数(バイト数)です。桁数は、システムリソースの、DB_ENCODE キーで指定されるデータベースのエンコードによって、バイト数変換された結果で判定します。

ただし、PostgreSQL では、varchar の桁数は、「文字数」となっており、「バイト数」チェックではなく、「文字数」チェックしたい場合は、システムリソースの DB_USE_TEXT_LENGTH を true にしてください。(標準は、バイト数チェック=false です)

•表示桁数

表示桁数は、画面上に表示する桁数です。

第10章 カラムオブジェクト

標準では、システムリソースの、HTML_COLUMNS_MAXSIZE で、検索条件などのテキストフィールドの最大桁数を、HTML_VIEW_COLUMNS_MAXSIZE で、登録時のテキストフィールドの最大桁数を定義しています。

カラムリソースで表示桁数を指定した場合は、こちらが優先されます。

- 表示種別、表示パラメーター(レンダラー)
カラムの値をどのように表示するか、指定します。
例えば、数字なら、カンマ編集したり、金額なら円マークをつけたり、日付なら年/月/日に加工したりできます。
org.opengion.hayabusa.db.CellRenderer インターフェースを実装したクラスで、org.opengion.plugin.column.Renderer_XXX.java の XXX 部を指定します。(パラメーター)には、このレンダラーに対する引数を指定できます。
詳細は、第11章 レンダラー、エディター、DBタイプ にて、述べます。
- 編集種別、編集パラメーター(エディター)
カラムの値をどのように編集するか、指定します。
テキストフィールドやプルダウンメニューなど、値のセットの方法を指定します。
org.opengion.hayabusa.db.CellEditor インターフェースを実装したクラスで、org.opengion.plugin.column.Editor_XXX.java の XXX 部を指定します。(パラメーター)には、このエディターに対する引数を指定できます。
詳細は、第11章 レンダラー、エディター、DBタイプ にて、述べます。
- 文字種別、文字パラメーター(DB タイプ)
カラムの値の属性を決定します。
この属性に応じて、値の整合性をチェックします。
例えば、数字タイプであれば、それに使用できる文字やルール(カンマや少数点の位置、マイナス符号の位置など)をチェックします。
org.opengion.hayabusa.db.DBType インターフェースを実装したクラスで、org.opengion.plugin.column.DBType_XXX.java の XXX 部を指定します。(パラメーター)には、このタイプに対する引数を指定できます。
詳細は、第11章 レンダラー、エディター、DBタイプ にて、述べます。
- データのデフォルト値
カラム自身のデフォルト値を指定します。
これは、アプリケーション全体で共通に使用されます。通常はデータベースのテーブルごとに初期値が変わる可能性がありますので、そのようなケースには適用できませんので、ご注意ください。
- ラベルカラム、コードカラム
カラムオブジェクトは、内部に言語別ラベルデータや、コードデータを持っています。通常は、カラム ID から、ラベルデータやコードデータを取り込みますが、ラベルカラムやコードカラムを指定すると、指定した ID で、ラベルやコードを作成します。たとえば、Yes/No のコードや、日付(日や、休日フラグなど)同じ属性が大量に存在する場合は、ラベルカラムやコードカラムを指定する事で、無駄なリソース登録を避けることが可能です。

第 11 章 コードリソースとカラムオブジェクト

この章では、カラムオブジェクトを修飾する機能として、コードリソースがあります。コードリソースとカラムオブジェクトの関係を説明します。

コードリソースは、キー(コードID)と選択枝からなります。
選択枝は、選択値 選択ラベル という組を複数指定することができます。

この、コードリソースを、カラムリソースの、レンダラー、エディターと組み合わせることで、検索結果から、プルダウンメニューや、選択値に対応するラベルを自動作成する事ができます。
コードリソースのラベルは、ラベルリソースで管理しています。国別にコードリソースを持つことはできません。(名称は、国別に表示できます。)
コード ID.選択値 をキーに、ラベルリソースに登録されます。登録は、コードリソースの登録画面から自動的にラベルが作成されます。(カラムリソースのラベルや画面リソースのラベルも同様に、登録画面から自動的にラベルリソースに出力されます。)

表示用レンダラーが、MENU の場合、データベースの検索結果の値と、コードリソースの選択値が比較され、合致した選択ラベルが表示されます。
これにより、選択値が、1、2、3や A,B,C などの記号(コード)の場合でも、その記号に応じた表示をさせることが可能になります。(例えば、1⇒要求、2⇒受付、3⇒完成など)
また、編集用エディターでは、これをプルダウンメニューで表示できます。ユーザーは、ラベルで選択できるため、非常に判りやすくなります。
また、コードリソースに使用可能な選択値を指定しておくことより、データベース定義をより完全に定義することが可能になります。(つまり、DB 設計時点の情報が、そのままノンコーディングで利用できます。)
さらに、SEQMENU などの編集用エディターを用いれば、1⇒2⇒3 というように、一度設定した値以降しか選択できない(先の例では、受付後に、要求に戻せない)などの制御を、リソース上で設定することが可能になります。

第12章 レンダラー、エディター、DBタイプ

この章では、カラムオブジェクトに対して、データを適用するときに、表示(レンダラー)、編集(エディター)、物理特性(DBタイプ)の取扱を説明します。

9. 表示用レンダラー

表示用レンダラーは、カラムの値をどのように表示するか、指定します。

例えば、数字なら、カンマ編集したり、金額なら円マークをつけたり、日付なら年/月/日に加工したりできます。

org.opengion.hayabusa.db.CellRenderer インターフェースを実装したクラスで、

org.opengion.plugin.column.Renderer_XXX.java の XXX 部を指定します。

カラムリソースの構築時にパラメーターとして引数を指定できます。

表示用レンダラーには、以下のクラスが用意されています。

属性クラス	概要説明
AUTOAREA	AUTOAREA レンダラは、カラムのデータをテキストエリアで表示する場合に使用するクラスです。
CHBOX	CHBOX レンダラーは、カラムのデータをチェックボックス文字情報として表示する場合に使用するクラスで、“0”と、“1”のみ使用できます。
CODE39	英数字をバーコードで使用する CODE39 のチェックデジット付き文字列に変換するレンダラークラスです。
COLUMN	COLUMN レンダラーは、データの値をカラム名と認識して、動的カラムを表示するクラスです。
CRYPT	パスワード情報など、重要な情報の暗号化された情報を表示する場合に使用するクラスです。
DATE	DATE レンダラーは、カラムのデータを表示パラメータで指定されたフォーマットで日付表示する場合に使用するクラスです。
DBLABEL	DBLABEL レンダラーは、値をラベルリソースの表示ラベルに変換するクラスです。
DBMENU	DBMENU レンダラーは、表示パラメータで指定された SQL 文を実行し、プルダウンメニューで表示する場合に使用するクラスです。
DECIMAL	DECIMAL レンダラーは、カラムのデータを Decimal(10進数、少数)表示する場合に使用するクラスです。
FILTER	特定の HTML タグのエスケープ文字を元のタグに戻して表示するクラスです。
FORM	FORM レンダラーは、表示パラメータで指定された FORM を表示するクラスで、元の Value を、\$1 として、使用可能です。
HM	HM レンダラーは、カラムのデータを時:分に分けて表示する場合に使用するクラスです。
HMS	HMS レンダラーは、カラムのデータを時:分:秒に分けて表示する場合に使用するクラスです。
HTML	HTML レンダラーは、HTML タグを含むデータを表示する場合に使用するクラスです。

KANA	KANA レンダラーは、カラムのデータに対し、半角カナを全角カナに変換して表示する場合に使用するクラスです。
LABEL	LABEL レンダラーは、カラムのデータをそのまま文字列として表示する場合に使用するクラスです。
MD	YMD レンダラーは、カラムのデータを日付(月/日)表示する場合に使用するクラスです。
MENU	MENU レンダラーは、カラムのデータをコードリソースに対応したラベルでプルダウンメニュー表示する場合に使用するクラスです。
MONEY	MONEY レンダラーは、カラムのデータを金額表示する場合に使用するクラスです。
MULTIQUERY	MULTIQUERY レンダラーは、表示パラメータで指定された SQL 文の実行結果を表示するクラスです。
NBSP	NBSP レンダラーは、内部のスペースを、 という文字列に置き換えます。
NUMBER	NUMBER レンダラーは、カラムのデータを数字表示する場合に使用するクラスです。
PASSWD	PASSWD レンダラーは、カラムのデータをパスワード情報(*****)として表示する場合に使用するクラスです。
PN	PN レンダラーは、カラムのデータを品番情報(11桁の文字列を3-5-3表示)として表示する場合に使用するクラスです。
PN2	PN2 レンダラーは、カラムのデータを品番情報(11桁の文字列を3-5-3)に対して、それぞれ、“PN_1”、“PN_2”、“PN_3”というクラスを付加します。
PRE	PRE レンダラーは、カラムのデータをそのまま PRE 表示する場合に使用するクラスです。
QUERY	QUERY レンダラーは、表示パラメータで指定された SQL 文の実行結果を表示するクラスで、元の Value を、\$1 として使用可能です。
RADIO	RADIO レンダラーは、カラムのデータをコードリソースに対応したラジオボタンの代替ラベルで表示する場合に使用するクラスです。
SLABEL	SLABEL レンダラーは、桁数の長いデータをコンパクトに表示させる LABEL レンダラーの類似クラスです。
TEXTAREA	TEXTAREA レンダラーは、カラムのデータをテキストエリアで表示する場合に使用するクラスです。
TMSTMP	TMSTMP レンダラーは、日付ネイティブのカラムのデータから、数字部分だけをピックアップし、日時(年/月/日 時:分:秒)表示する場合に使用するクラスです。
WRITABLE	先頭1文字目が、アンダーバー“_”の場合は書込み禁止属性を考慮するレンダラークラスです。
XXXX	XXXX レンダラーは、パラメータで指定された XXXX フォーマットに対して、値を変換します。
YM	YM レンダラーは、カラムのデータを日付(年/月)表示する場合に使用するクラスです。
YMD	YMD レンダラーは、カラムのデータを日付(年/月/日)表示する場合に使用するクラスです。
YMD31	YMD レンダラーは、カラムのデータを日付(年/月/日)表示する場合に使用するクラスです。
YMDH	YMDH レンダラーは、カラムのデータを日時(年/月/日 時:分:秒)表示する場合に使用するクラスです。

10. 編集用エディター

編集用エディターは、カラムの値をどのように編集するか、指定します。
 テキストフィールドやプルダウンメニューなど、値のセットの方法を指定します。
 org.opengion.hayabusa.db..CellEditor インターフェースを実装したクラスで、
 org.opengion.plugin.column.Editor_XXX.java の XXX 部を指定します。
 カラムリソースの構築時にパラメーターとして引数を指定できます。

編集用エディターには、以下のクラスが用意されています。

属性クラス	概要説明
AUTOAREA	AUTOAREA エディターは、カラムのデータをテキストエリアで編集する場合に使用するクラスです。
CHBOX	カラムのデータをチェックボックスで編集する場合に使用するエディタークラスです。
CHBOX2	CHBOX2 エディターは、カラムのデータをチェックボックスで編集する場合に使用するクラスです。
COLUMN	動的カラムのデータを編集する場合に使用するエディタークラスです。
CRYPT	パスワード情報など、重要な情報の暗号化された情報を編集する場合に使用するクラスです。
DBMENU	カラムの編集パラメーターのSQL文の実行結果より、プルダウンメニューを作成して編集する場合に使用するエディタークラスです。
DBRADIO	DBRADIO エディターは、カラムの編集パラメーターのSQL文の実行結果より、動的にラジオボタンを作成して編集する場合に使用するエディタークラスです。
DECIMAL	<p>DECIMAL エディターは、カラムのデータを Decimal(10進数、少数)表示する場合に使用するクラスです。
ENTCLM	動的カラムの Entry カラムを編集する場合に使用するエディタークラスです。
HIDDEN	カラムのデータを HIDDEN で編集する場合に使用するエディタークラスです。
HTML	HTML タグを含むデータを編集する場合に使用するエディタークラスです。
INDBMENU	INDBMENU エディターは、カラムの表示パラメーターのSQL文を実行結果より、作成したプルダウンメニューと、テキストフィールドによる入力の両方をサポートする、編集に使用するクラスです。
INMENU	INMENU エディターは、コードリソースに対応したプルダウンメニューと、テキストフィールドによる入力の両方をサポートする、編集に使用するクラスです。
MENU	MENU エディターは、カラムのデータをコードリソースに対応したプルダウンメニューで編集する場合に使用するクラスです。
NUMBER	NUMBER エディターは、カラムのデータを数字編集する場合に使用するクラスです。
PASSWD	PASSWD エディターは、パスワード情報(*****)として編集する場合に使用するクラスです。
PN	PN エディターは、カラムのデータをカラムのデータを品番情報(11桁の文字列を3-5-3編集)する場合に使用するクラスです。
QUERY	QUERY エディターは、編集パラメータで指定された SQL 文の実行結果をテキストエリアに表示するクラスで、元の Value を、\$1 として使用可能です。
RADIO	RADIO エディターは、カラムのデータをコードリソースに対応したラジオボタンで編集する場合に使用するクラスです。

RADIO2	RADIO2 エディターは、カラムのデータをチェックボックスで編集する場合に使用するクラスです。
TEXT	TEXT エディターは、カラムのデータをテキストフィールドで編集する場合に使用するクラスです。
TEXTAREA	TEXTAREA エディターは、カラムのデータをテキストエリアで編集する場合に使用するクラスです。
UPLOAD	UPLOAD エディターは、ファイルアップロードを行う場合に使用する編集用クラスです。
WRITABLE	先頭1文字目が、アンダーバー() の場合に、書込み禁止属性()を強制的に付与するクラスです。
YM	YM エディターは、カラムのデータを日付(年/月)編集する場合に使用するクラスです。
YMD	YMD エディターは、カラムのデータを日付(年/月/日)編集する場合に使用するクラスです。
YMD2	YMD エディターは、カラムのデータを日付(年/月/日)編集する場合に使用するクラスです。
YMDH	YMDH エディターは、カラムのデータを日時(年/月/日 時:分:秒)編集する場合に使用するクラスです。

11. データのタイプ

データのタイプは、カラムの値の属性を決定します。

この属性に応じて、値の整合性をチェックします。

例えば、数字タイプであれば、それに使用できる文字やルール(カンマや少数点の位置、マイナス符号の位置など)をチェックします。

mis.pdm.hayabusa.db.column.DBType_XXX.java の XXX 部を指定します。

カラムリソースの構築時にパラメーターとして引数を指定できます。

データのタイプには、以下のクラスが用意されています。

属性クラス	概要説明
ALL	半角/全角混在の一般的な制限のない文字列を扱う為の、カラム属性を定義します。
CRYPT	半角/全角混在の一般的な制限のない暗号化された文字列を扱う為の、カラム属性を定義します。
DATE	文字列の厳密な日付属性(年/月/日)の半角の日付を扱う為の、カラム属性を定義します。
DD	DATA_DEFAULT カラムで、内容の整合性を整えます。
HMS	文字列の時間属性(時:分:秒)の半角の時間を扱う為の、カラム属性を定義します。
K	全角のみで構成される文字列を扱う為の、カラム属性を定義します。
KCL	全角+改行 C(CR,LF)のみで構成される文字列を扱う為の、カラム属性を定義し

	ます。
KX	<p>DBType_KX は、全角/半角混在の Char または Varchar2 属性に対応するクラスで一般的な制限のない文字列カラム属性を定義します。
MD5	パスワード情報など、重要な情報のハッシュコード(MD5)を扱う為の、カラム属性を定義します。
NVAR	Unicode 文字列の値を HTML のエスケープ記号(&#xZZZZ;)に変換する、カラム属性を定義します。
OASNM	旧 OAS で実装していた、半角文字(カナ含む)名称用記号を扱う為の、カラム属性を定義します。
PN	情報機器事業部向け、品番情報の文字列を扱う為の、カラム属性を定義します。
R	半角少数付き数字の NUMBER を扱う為の、カラム属性を定義します。
S9	半角数字の NUMBER を扱う為の、カラム属性を定義します。
X	一般的な半角文字列を扱う為の、カラム属性を定義します。
X9	半角数字の NUMBER を扱う為の、カラム属性を定義します。
XH	半角文字+半角カタカナを扱う為の、カラム属性を定義します。
XHU	半角文字+半角カタカナの大文字のみに制限された文字列を扱う為の、カラム属性を定義します。
XK	半角/全角混在の一般的な制限のない半角優先文字列を扱う為の、カラム属性を定義します。
XKZ	半角/全角混在のクラスですが、半角カタカナのみを通さない文字列を扱う為の、カラム属性を定義します。
XL	半角小文字の英数字の文字列を扱う為の、カラム属性を定義します。
XLU9	半角英数字のみの文字列を扱う為の、カラム属性を定義します。
XU	半角大文字の英数字の文字列を扱う為の、カラム属性を定義します。
XU9	半角英数大文字のみの文字列を扱う為の、カラム属性を定義します。
YM	文字列の日付属性(年/月)の半角の日付を扱う為の、カラム属性を定義します。
YM01	文字列の開始日付属性を規定する半角文字列を扱う為の、カラム属性を定義します。
YM31	文字列の終了日付属性を規定する半角文字列を扱う為の、カラム属性を定義します。
YMD	文字列の日付属性(年/月/日)の半角の日付を扱う為の、カラム属性を定義します。
YMDH	文字列の日付属性(年/月/日 時:分:秒)の半角の日付を扱う為の、カラム属性を定義します。

第13章 その他リソースファイル

この章では、ラベル、メッセージ、ユーザー、画面、の各リソースについて説明します。

1. ラベルリソース

ラベルリソースは、キー(ラベルID)と値(ラベル値)からなります。

キーは、カラムIDと連動しており、カラムに対して表示ラベルを設定するイメージになります。

ラベルリソースは、国別に複数のリソースを持つことができ、かつ同時に使用することが可能なため、同一アプリケーションによる国際化対応が可能になります。

国別リソースの切替は、ユーザー情報の言語で指定できますので、ログインする端末やIPアドレス等による切替でないため、使用者がどこに居ても自分の言語で表示することが可能になります。

ラベルリソースでは、従来持っていた、メッセージリソースを統合しました。また、画面名称、コードリソースのコード表示もラベルリソースで管理しています。

ラベルリソースには、名前(長)、名前(短)を持っています。名前(長)は、検索条件の入力画面や、カラムタグで作成する画面部品などのリソース名に使われます。名前(短)は、表形式の一覧表示を行う場合に使われます。表形式表示の場合は、略称を用いたり、折返しラベルを用いたりするケースがあり、それらをリソースで登録しておけます。この考えは、コードリソースにも適用されます。

2. メッセージリソース

メッセージリソースは、ラベルリソースに統合されました。

従来のメッセージリソースと同様に、ラベルリソースに引数を渡すことが可能です。これは、`java.text.MessageFormat` クラスと同等の使用方法が可能です。ただし、引数には、`String` (文字列)しか渡すことはできません。

3. ユーザーリソース

ユーザーリソースは、キー(ユーザーID)とユーザー属性からなります。

ユーザー属性は、ログイン時のユーザーIDから、`UserInfo` オブジェクトを作成し、JSP 上では、`UserInfo` タグでアクセスすることが可能です。

ユーザーリソースは、国別にリソースを持っていません。

・ユーザー

ユーザーリソースのキーになります。ログインユーザーは、セッションごとに作成されるため、1人が同時にサーバーに接続して、別々の処理を行うことが可能になります。ただし、物理的な操作(ファイルダウンロードなど)は、ユーザーIDごとのワークフォルダをテンポラリーとして使用するため、同一ログインユーザーで異なる人がログインする運用は避けてください。

- パスワード
ユーザー認証用のパスワードです。本フレームワークでは、このパスワードは使用していません。ユーザー認証には、Tomcat の認証を利用しています。
- 言語
ユーザーごとに言語を持っており、この言語に応じて国別リソースが使用されます。そのため、ユーザーがどこからログインしようとも、自分の言語に応じた画面で操作することが可能になります。
- 日本語名称
母国語での氏名情報です。画面上にログインユーザーとして表示します。
- 英語名称
英語での氏名情報です。基本的に、英語での氏名情報は必須項目として使用できるように登録しておいてもらいます。
- メールアドレス
メール等を利用する場合のメールアドレスを指定します。
- ロールズ
ユーザーのロール(役割)を指定します。これは、画面制御用の権限を付与することが可能です。通常は、役職(部長、管理者、などの役職)を指定し、その権限に応じた画面のアクセス制御を行います。
このロールが、root の場合は、すべての画面にアクセスすることが可能です。
- データロールズ
ユーザーのデータ単位のロールを指定します。
画面が利用できても、データを見る権限が与えられていないケースで使用します。現時点では、アプリケーション側での制御が必要です。
- 有効日,パスワード変更日時
ログイン有効日とパスワードの変更日付です。有効日を過ぎるとログインできなくなります。有効日とパスワード変更日付は連動していません。

4. GUIリソース

GUIリソースは、キー(画面ID)と画面属性からなります。
画面属性は、各 JSP 画面のフォルダ名と連動しています。※
画面属性は、GUIInfo オブジェクトより、guiInfo タグでアクセスすることが可能です。
GUI リソースとユーザーリソースの関係より、どのユーザーがどの画面をアクセスできるか設定する事が可能です。
GUIリソースは、キー(画面ID)とロールズでユニークになります。(データベース上では、システムID,作成区分も加わります)。これは、同一画面IDでロール違いのレコードを作れるという事を意味し、物理アドレスを分けることができるという事です。

- 画面 ID

GUI リソースのキーになります。(レコードとしては、画面 ID+ロールでユニーク)
画面 ID と実アドレスは、基本的には1対1の関係ですが、異なる画面 ID から同一実行アドレスを実行することも可能です。ロール違いの同一画面 ID で、異なる実アドレスを指定する事も可能です。
- 実アドレス

メニューから呼ばれるディレクトリです。
一般には、画面 ID と同じです。
http で始まるアドレスを指定した場合は、フォルダではなく、指定の HTTP アドレスにアクセスします。
- 表示順

画面の表示順を指定します。
この表示順で、画面が上から順番に表示されます。大分類として、下記の画面レベルでブレイクすると、次の画面レベルが同一のグループであるという認識をしますので、表示順でが重要です。
- 画面レベル

メニューを大分類、小分類、通常、など区別します。
先の表示順でも述べましたように、表示順が重要です。
- 名称、名称(長)、名称(短)

名称は、画面 ID に対する、設計時の名称です。実際にメニューフレーム上に表示される画面名称は、名称(短)です。
名称(短)が登録されていない場合は、名称(長)がメニューに使用されます。
名称(長)は、QUERY 部に表示される画面名称で、必須です。
- ロールズ

ユーザーロールに対する権限を複数指定できます。
ユーザーのロールは、この画面のロールに含まれていないか、文字列での検索を行います。画面側には、AAA|BBB|CCC と『|』で区切られた複数のロールを記述することが可能です。
- アクセスモード

メニュー表示、アクセス、ファイル出力などの画面ごとの制限を指定します。
情報は、各ロールに対して、3つの項目で表しています。

1:"-"	メニュー非表示	"m"	メニュー表示	"p"	(強制プルダウン表示
2:"-"	アクセス拒否	"r"	読取許可	"w"	読取、書込許可
3:"-"	未対応	"d"	ファイル出力	"u"	ファイル入力
	"f"	ファイル入出力	"e"	画面編集可	

第13章 その他リソースファイル

この3項目を順次つないで、“--”, “-r”, “-w”, “mr”, “mw” などの設定を行います。モードが設定されている場合は、共通モードとして、すべてのロールに同じモードを割り当てます。個別に割り当てる場合は、ROLES 情報の後ろに () 付きで追記します。例えば、AAA|BBB(-r)|CCC とすれば、BBB ロールの場合は、読取専用になります。

ロールをぶつける場合は、AND 演算になります。

・ターゲット

メニューフレームから指定されるリンクのターゲットを指定します。

通常は、CONTENTS を指定しておき、各画面ホルダの inde.JSP が呼ばれ、そこから、QUERY と RESULT フレームに別れます。

ここに、それら以外の名称(例えば、_new)を指定すると、ポップアップに、_top で、自分自身のフレームを入れ替えます。

第 V 部 データ・アクセス

『定義は、定めるよりも反論するほうがやさしい』
アリストテレス

ここでは、データのアクセス方法について説明します。
構成は、次のとおりです。

第 14 章 SQL、PL/SQL および Java
データを取得するために使用する SQL (Structured Query Language) と、手続き型言語機能拡張である PL/SQL について説明します。

第 15 章 Query と Update
データの取り出し方、更新の仕方について、説明します。なお、取り出すストレージは、データベースだけではなく、ファイルやメールサーバーでも構いません。

第 16 章 View、WriteTable、ReadTable
取り出したデータの出力方法について、説明します。通常は、標準出力 (画面表示) ですが、ファイルへの入出力方法についても、説明します。

第 17 章 tableFilter、process、schedule
あまり一般的ではありませんが、タグから利用できる各種クラスについて説明します。

第 18 章 filter (Servlet のフィルター機能の実装)
Servlet の標準機能である、フィルターの実装クラスについて説明します。

第 19 章 高度なデータアクセス
通常の SQL によるデータ取り出しと、カラムオブジェクトの機能を組み合わせることによって、高速で汎用性の高いデータ取り出し方法について、説明します。

第14章 SQL、PL/SQL および Java

この章では、データを取得するために使用するSQL (Structured Query Language)と、手続き型言語機能拡張であるPL/SQL について説明します。

1. 共通オブジェクトの定義

PL/SQL の共通オブジェクト定義は、次のとおりです。

① ARG_ARRAY

```
CREATE OR REPLACE TYPE ARG_ARRAY AS VARRAY(100) OF VARCHAR2(100);
```

- ・引数の受け渡しは、配列タイプで行います。
- ・すべて文字列タイプとし、必要であれば、PL/SQL 内でNUMBER型等に変換します。
- ・引数の最大数は100個とし、最大文字数は、100桁とします。
- ・引数の順番は、query タグの args 属性で設定します。
- ・もちろん、受け取り側は、その順番で受け取る必要があります。

② ERR_MSG_ARRAY

```
CREATE OR REPLACE TYPE ERR_MSG AS OBJECT  
( NO NUMBER,                /* 行番号 */  
  KEKKA OUT NUMBER ,        /* 結果 0:正常 1:警告 2:異常 */  
  ID VARCHAR2(10),          /* エラーメッセージID */  
  MSG1 VARCHAR2(100),       /* メッセージの引数1 */  
  MSG2 VARCHAR2(100),       /* メッセージの引数2 */  
  .....  
  MSG5 VARCHAR2(100) );     /* メッセージの引数5 */  
/
```

```
CREATE OR REPLACE TYPE ERR_MSG_ARRAY AS VARRAY(500) OF ERR_MSG;
```

- ・エラーメッセージは、オブジェクトタイプで宣言し、最大500件まで登録することができます。
- ・それ以上のエラーが出た場合は、表示を打ち切ることとします。
- ・エラーメッセージに引数を5個まで登録できます。

③ SYSARG_ARRAY

```
CREATE OR REPLACE TYPE SYSARG AS OBJECT  
( NO NUMBER,                /* 行番号 */  
  CDKH VARCHAR2(1) );       /* 改廃コード A:追加 C:変更 D:削除 */  
/
```

```
CREATE OR REPLACE TYPE SYSARG_ARRAY
AS VARRAY(1000) OF SYSARG;
```

```
/
```

- ・プロシジャーに渡すシステム制御情報は、行番号と改廃コードとします。
- ・エラーメッセージや処理の振り分けは、これらの情報を用いて PL/SQL 側で記述します。
- ・これらは、1000件までの情報を渡しますが、それ以上の登録が画面から必要な場合は、別途専用の PL/SQL と、登録用 Java プログラムが必要になります。
- ・それ以上の情報の一括登録は、画面情報をインプットとするのではなく、別に、要求テーブルや、ファイルなどを利用して DB 登録する事を推奨します。

④ PROCEDURE SET_ERRMSG

```
CREATE OR REPLACE PROCEDURE SET_ERRMSG
(P_ERRMSG IN OUT ERR_MSG_ARRAY,
 P_NO     IN     NUMBER := NULL,
 P_KEKKA IN     NUMBER := NULL,
 P_ID     IN     VARCHAR2 := NULL,
 P_MSG1  IN     VARCHAR2 := NULL,
 P_MSG2  IN     VARCHAR2 := NULL,
 P_MSG3  IN     VARCHAR2 := NULL,
 P_MSG4  IN     VARCHAR2 := NULL,
 P_MSG5  IN     VARCHAR2 := NULL) IS
```

```
TOO_MANY_ERRORS EXCEPTION; --エラー件数最大数オーバーエラー
V_ERRMSG      ERR_MSG;    -- エラーメッセージオブジェクト
```

エラーメッセージ配列を設定する場合の初期化などを行うプロシージャ

2. 検索系のコール条件

検索系の PL/SQL の procedure のコール条件は、次のとおりです。

① 検索専用 PL/SQL

条件を与えて、検索結果をカーソルで返します。

検索におけるエラーは、入力値の不正な値だけなので、基本的にはエラーメッセージは返しません。

なお、この PL/SQL 内でデータベースの更新を行うことは、PL/SQL 作成者の責任の元、行うことができます。

```
CREATE OR REPLACE PACKAGE HAYABUSA IS
TYPE CUST_CURSOR IS REF CURSOR ;
PROCEDURE XXXXXX (
KEKKA OUT NUMBER,
ERRMSG OUT ERR_MSG_ARRAY,
```

```

RC1 OUT CUST_CURSOR,
ARGS IN ARG_ARRAY);
END;
/

CREATE OR REPLACE PACKAGE BODY HAYABUSA IS
PROCEDURE XXXXXX (
KEKKA OUT NUMBER ,           /* 結果 0:正常 1:警告 2:異常 */
ERRMSG OUT ERR_MSG_ARRAY,   /* エラーメッセージ配列 */
RC1 OUT CUST_CURSOR,        /* 結果はカーソルタイプで返す。*/
ARGS IN ARG_VARRAY,        /* 引数は、文字列の配列タイプで
                             受け取る。 */
) AS
BEGIN
OPEN RC1 FOR
SELECT A.COLUMN_ID,A.TABLE_NAME,A.COLUMN_NAME,B.NAME_JA,
B.DATA_TYPE,B.USE_LENGTH,A.FGADD
FROM DB05 A, DB03 B
WHERE A.COLUMN_NAME = B.COLUMN_NAME
ORDER BY A.SYSTEM_ID,A.TABLE_NAME,A.COLUMN_ID ;
END XXXXXX;
END HAYABUSA;
/

```

3. 登録系のコール条件

登録系の PL/SQL の procedure のコール条件を以下のように定めます。

① 登録専用 PL/SQL

条件を与えて、検索結果をメッセージ配列で返します。
登録においては、結果として、OK/NG を返し、NG の場合にメッセージを表示します。なお、この PL/SQL 自体で全ての処理を完結させる必要があります(トランザクション制御は、この PL/SQL で自分で対応する)。登録結果を再度検索したい場合は、JSP 画面上で queryタグを用いて再検索してください。

```

CREATE OR REPLACE PROCEDURE YYYY(
KEKKA OUT NUMBER ,           /* 結果 0:正常 1:警告 2:異常 */
ERRMSG OUT ERR_MSG_ARRAY,   /* エラーメッセージ配列 */
NAMES IN VARCHAR2,         /* カラム名チェック用文字列 */
SYSARGS IN SYSARG_ARRAY,   /* 登録条件配列 */
USERARGS IN USERARG_ARRAY, /* 登録データ配列 */
) IS
.....

```

```

CREATE OR REPLACE TYPE USERARG AS OBJECT ( ..... );

```

/

```
CREATE OR REPLACE TYPE USERARG_ARRAY AS VARRAY(XXX) OF US  
ERARG;
```

/

- ユーザー定義オブジェクトの配列は、各プロシージャごとに作成します。
- これは、各引数の個数等は、個別に設定することとします。
- NAMES はカラム名チェック用の文字列です。ここでは、USERARGSで定義されているカラム名の順番を、カンマ区切りで設定します。
例 'OYA,KO,HOJO,DYSTR,DYEND'
- エンジン側は、上記のカラム名を元にデータを配列に設定しますので、受け側 (PL/SQL) では、自分自身の定義とチェックするように作ってください。

第15章 Query と Update

この章では、データの取り出し方、更新の仕方について、説明します。なお、取り出すストレージは、データベースだけではなく、ファイルやメールサーバーでも構いません。

query タグは、データベースからデータを検索して、DBTableModel を作成します。同様に、PL/SQLUpdate タグ、または、tableUpdate は、DBTableModel の値を、データベースに書き込みます。

どちらも、DBTableModel の値をやり取りしますので、query タグで検索した結果を、そのまま tableUpdate タグで登録したり、ファイルから読み取った、DBTableModel をデータベースに、update したり、query で検索した DBTableModel をファイルに書き出したり相互運用可能です。

query タグも、update タグも、派生クラスがしますので、以下に説明いたします。

1. query タグ

query タグは、データベース検索に使用されるタグです。queryType の指定により、実際に呼び出す Query クラスを指定します。通常は、queryType = “JDBC” が良く使われますが、PL/SQL によりカーソルを返す場合は、”JDBCErrMsg” を指定します。

```
<og:query
  tableId          : (通常は使いません)結果を DBTableModel に書き込んで、
                    session に登録するときのキーを指定します。
  queryType       : 検索を実行する手段を指定します。
  dbid            : (通常は使いません)Query オブジェクトを作成する時の
                    DB 接続 ID を指定します。
  scope          : キャッシュする場合の範囲を指定します。
  command        : 処理コマンドをセットします。
  maxRowCount    : (通常は使いません)データの最大読み込み件数を指定します。
  SkipRowCount   : (通常は使いません)データの読み始めの初期値を指定します。
  debug          : デバッグ情報(タグのボディ部)を 出力する/しないを
                    指定します。
  displayMsg     : 検索結果をの件数や登録された件数を画面上に表示するか
                    どうかを指定します。
  names          : PL/SQL を利用する場合の引数にセットすべき データの名称
                    を指定します。
  StopZero       : 検索結果が0件のとき処理を続行するかどうかを指定します。
>
  SQL 文
</og:query>
```

2. PL/SQLUpdate タグ

PL/SQLUpdate タグは、PL/SQL を使ってデータベース登録に使用されるタグです。queryType の指定により、実際に呼び出す Query クラスを指定します。通常は、queryType = “JDBCPL/SQL” が良く使われますが、引数付き SQL 文を直接実行する場合は、”JDBCPrepared” を指定します。

```
<og:PL/SQLUpdate
  command      = "{@command}"
  names        = "UNIQ, USRID, ECNO, EDBN" → 引数 (配列) のカラム名
  dbType       = "RK0271ARG"           → 引数 (配列) の定義ファイル名
  queryType    = "JDBCPL/SQL" >
  { call RK0271E.RK0271E( ?, ?, ?, ?, ? ) } →CALL する PL/SQL
</og:PL/SQLUpdate>
```

3. tableUpdate タグ

tableUpdate タグは、SQL 文でデータベース登録に使用されるタグです。queryType の指定により、実際に呼び出す Query クラスを指定します。通常は、queryType = “JDBCTableUpdate” を使用します。このタグは、INSERT 文や UPDATE 文を直接 BODY 部に記述しますが、tableUpdateParam タグを使用すれば、それらの SQL 文を動的に作成してくれます。

- ・引数で、SQL 文を渡して使用する場合

```
<og:tableUpdate
  command      = "{@command}"
  queryType    = "JDBCTableUpdate" >
  {@SQL}
</og:tableUpdate>
```

- ・tableUpdateParam を使用する場合

```
<og:tableUpdate
  command      = "{@command}"
  queryType    = "JDBCTableUpdate"
  sqlType      = "{@sqlType}" // tableUpdateParam の sqlType と一致
>
<og:tableUpdateParam
  sqlType      = "{@sqlType}" // INSERT, COPY, UPDATE, MODIFY, DELETE
  table        = "{@TABLE_NAME}" // 処理対象のテーブル名
  names        = "{@names}" // 処理対象のカラム名
  omitNames    = "{@omitNames}" // 処理対象外のカラム名
  where        = "{@where}" // 処理対象を特定するキー
  constKeys    = "{@constKeys}" // 処理カラム名の中の固定情報カラム名
  constVals    = "{@constVals}" // 処理カラム名の中の固定情報設定値
/>
</og:tableUpdate>
```

4. queryType

queryType 属性は、実際に起動される Query インターフェースの派生クラス名を指定します。

基本的に、query タグは、リクエスト情報から、update タグは、DBTableModel 情報から、値を取り出して、queryType 属性で指定のクラスに引数を渡します。

queryType	説明	使用可能タグ
JDBCCallable	Query インターフェースを継承した実装クラスです。java.sql.CallableStatement を用いて、データベース検索処理を行います。引数は、従来の PL/SQL の実行が可能のように、第一引数はエラーコード、第二引数は、エラーメッセージを返してきます。第三引数以降は、自由に指定できます。	query/ update
JDBCErrMsg	Query インターフェースを継承した実装クラスです。java.sql.CallableStatement を用いて、データベース検索処理を行います。引数を配列指定で渡すことができ、エラー時には、DBErrMsg オブジェクトにエラー情報を格納して返すことが可能です。	query カーソル を返します。
JDBCKeyEntry	EntryQuery タグで使用します。java.sql.CallableStatement を用いて、データベース検索処理を行います。引数に、キーと値をセットで配列指定で渡すことができ、エラー時には、DBErrMsg オブジェクトにエラー情報を格納して返すことが可能です。	entryQuery
JDBCPL/SQL	Query インターフェースを継承した実装クラスです。java.sql.CallableStatement を用いて、データベース検索処理を行います。引数に、SYSARG_ARRAY と、USERARG_ARRAY を配列指定で渡すことができ、エラー時には、DBErrMsg オブジェクトにエラー情報を格納して返すことが可能です。	update
JDBCPrepared	Query インターフェースを継承した実装クラスです。java.sql.PreparedStatement を用いて、データベース検索処理を行います。引数に、指定した値を配列で渡します。	query/ update
JDBC	Query インターフェースを継承した実装クラスです。java.sql.Statement を用いて、データベース検索処理を行います。引数は無しです。(与えられた SQL 文を実行します。)	query
JDBCUpdate	Query インターフェースを継承した実装クラスです。java.sql.CallableStatement を用いて、データベース登録処理を行います。引数は、そのまま配列に格納して処理を行います。エラー時の処理や、検索結果の取り出しはできません。	query/ update

第 16 章 View 、WriteTable 、ReadTable

この章では、取り出したデータの出力方法について、説明します。通常は、標準出力(画面表示)ですが、ファイルへの入出力方法についても、説明します。

1. view タグ

view タグは、DBTableModel を表示するためのタグです。通常は、query タグにより、DBTableModel が作成され、指定の scope に登録されるので、そこから取り出して、表示します。表示する場合に、表示方法を、viewFormId 属性で指定された派生クラスを使用して表示します。

<og:view

viewFormId	: session から取得する ViewForm オブジェクトの ID。
viewFormType	: ViewForm オブジェクトを作成する ViewFormFactory に与える ID。
scope	: キャッシュする場合のスコープを指定します。
command	: 処理コマンド
startNo	: 表示データを作成する場合の表示の開始行番号をセットします。
pageSize	: 表示データを作成する場合の1ページの行数をセットします。
columnWritable	: 書き込み可能カラム名を、カンマ区切りで与えます。
noWritable	: 書き込み不可カラム名を、カンマ区切りで与えます。
columnDisplay	: 表示可能カラム名を、カンマ区切りで与えます。
noDisplay	: 表示不可カラム名を、カンマ区切りで与えます。
language	: 言語コードを指定します。
tableId	: session から取得する DBTableModel オブジェクトの ID。
writable	: rowWritable (行が書き込み可能かどうか)を設定します。
checked	: 書き込み可能な行(rowWritable == true)のチェックボックスに対して初期値を 選択済みにするか、非選択済みにするかを指定します。
pagePlus	: 1ページの行数の増加分をセットします。
rowspan	: 表示データを作成する場合のフォーマットの行数をセットします。
skip	: 書き込み可能な行(rowWritable == true)のみを表示対象とするかどうかを指定します。
viewLinkId	: request から取得する ViewLink に対応する Attributes オブジェクトの ID。
selectedType	: 表示時の選択用オブジェクトのタイプを指定します。
optionTypeAttributes	: テーブル等のチェックボックスに属性を付加します。JavaScript などの HTML 基本タグ以外の属性を、そのままチェックボックス/ラジオボタン等に使用します。
noMessage	: 検索結果メッセージを表示する/しないを設定します。
backLinkCount	: ページの先頭へのリンクの間隔をセットします。

```
</>
```

Body 部分の存在する場合(主に、Format 系クラス)は、以下のようになります。

例:

```
<og:view
  viewFormType = "HTMLCustomTable"
  .....
>
  <thead> ..... </thead>
  <tbody> ..... </tbody>
  <tfoot> ..... </tfoot>
</og:view>
```

2. viewFormType 属性

viewFormType 属性は、org.opengion.hayabusa.html.ViewForm インターフェースの実装である、org.opengion.plugin.view.ViewForm_XXXX.java クラスのXXXX部分を指定します。このクラスを切り替えることで、DBTableModel を色々な形で表示させることが可能になります。

属性クラス	概要説明
CustomData	ヘッダ、フッタ、ボディを指定して作成する、自由レイアウトが可能な、カスタムテーブル表示クラスです。
HTMLAjaxTreeTable	JavaScript のツリー階層を持ったテーブル表示を行う、ツリーテーブル表示クラスです。
HTMLCalendar	1ヶ月分のカレンダー形式で、検索結果を表示する、カレンダー表示クラスです。
HTMLCrossTable	クロス集計テーブル作成クラスです。
HTMLCustomTable	ヘッダ、フッタ、ボディを指定して作成する、自由レイアウトが可能な、カスタムテーブル表示クラスです。
HTMLCustomTreeBOM	JavaScript のツリー階層を持ったテーブル表示を行う、ツリーテーブル表示クラスです。
HTMLDynamic	各フィールド情報から、動的にカラムを作成する動的カラム一覧表示クラスです。
HTMLEntry	エントリ形式フォーム作成クラスです。
HTMLFormatTable	ヘッダー部分のフォーマットに応じたテーブルを自動作成する、フォーマットテーブル作成クラスです。
HTMLFormatTextField	フォーマットを外部から指定して作成する自由レイアウトの、テキストフィールド表示クラスです。

HTMLGanttTable	ガントチャート(テーブル形式)を作成する、ガントチャート表示クラスです。
HTMLRotationTable	行と列を入れ替えて表示する、テーブル回転表示クラスです。
HTMLSeqClmTable	検索結果を自動的に表形式に変換する、テーブル作成クラスです。
HTMLSimpleList	検索結果を単純なリスト形式で表示するクラスです。
HTMLTable	検索結果を自動的に表形式に変換する、テーブル作成クラスです。
HTMLTextField	検索結果から、テキストフィールドタグを自動生成する、テキストフィールド作成クラスです。
HTMLTimeTable	時間軸を持つタイムテーブルの表示を行うクラスです。
HTMLTreeBOM	JavaScript のツリー階層を持ったテーブル表示を行う、ツリーテーブル表示クラスです。

3. writeTable タグ

writeTable タグは、DBTableModel をファイルに書き出すためのタグです。このタグにより書き出された、DBTableModel は、readTable タグにより、もとの DBTableModel に戻すことが可能になります。query タグ、update タグ等と組み合わせて、データベース、ファイルをシームレスに連携させることが可能になります。

また、writerClass 属性に、DBTableWriter インターフェースの派生クラスを適用することで、各種形式(TSV 形式、CSV 形式、XML 形式、フラットファイル形式等)で、セーブできます。

```
<og:writeTable
  separator      : 可変長ファイルを作成するときの項目区切り文字をセットします。
  HeaderSequence
                  : Label,Name,Size,Class,Data の各フィールドの頭文字のアルファベットで出力順をセットします。
  fileURL        : fileURL を セットします。
  filename       : ファイルを作成するときのファイル名をセットします。
  encode         : ファイルを作成するときのファイルエンコーディング名をセットします。
  writerClass    : 実際に書き出すクラス名(の略称)をセットします。
  fileAppend     : 追加モードで書き込むかどうかをセットします。
  direct         : 結果をダイレクトに EXCEL ファイルとして出力するかどうかをセットします。
  zip            : 結果をファイルに出力するときに、ZIP で圧縮するかどうかをセットします。
  language       : 言語コードを指定します。
  tableId        : session から所得する DBTableModel オブジェクトの ID。
  scope          : キャッシュする場合の範囲を指定します。
/>
```

4. writerClass 属性を、以下に示します。

属性クラス	概要説明
CSV	カンマ区切りダブルクォートゼロカンマファイル(CSV)形式書き込みクラスです。
CSV2	加工なしカンマ区切りダブルクォートファイル(CSV)形式書き込みクラスです。
Calc	Calc ファイルの書き出しクラスです。
CalcDef	Calc ファイルの書き出しクラスです。
CalcDefAno	Calc ファイルの書き出しクラスです。
Data	加工なし区切り文字指定データの書き出しクラスです。
Data2	加工なしダブルクォート区切り文字指定データの書き出しクラスです。
Default	区切り文字指定(初期値:タブ)ゼロカンマファイルの書き出しクラスです。
Excel	ネイティブ EXCEL ファイルの書き出しクラスです。
Fixed	固定長文字ファイルの書き出しクラスです。
Properties	プロパティファイル形式(エンジン専用特殊形式)の書き出しクラスです。
T	Excel での文字変換関数 =T("値") という文字列で書き出すクラスです。
XML	TableWriter を XML 形式で出力する為の実装クラスです。

コラム ORACLE XDK 形式

writerClass の XML について、簡単に説明しておきます。

これは、ORACLE XDK(XML Development Kit) で使用できるXML形式でファイルをセーブすることで、XDKのユーティリティを用いてデータベースへ登録することが可能になります。

また、この形式は、XSQLサブレットの出力形式でもあるため、各種文献やサンプル等で、XSLTファイルや、CSSファイルなどの流用が可能と思われます。

このXML は、下記のような形式をしています。

ROWSET と ROW があり、ROW の属性の num が行番号に相当します。

そして、各カラムがタグになり、データがBODY部に書かれています。

```
<?xml version = '1.0' encoding = 'Shift_JIS'?>
<ROWSET>
  <ROW num="1">
    <UNIQ>2</UNIQ>
    <SYSTEM_ID>DBDEF</SYSTEM_ID>
    .....
  </ROW>
  <ROW num="2">
    <UNIQ>1</UNIQ>
    <SYSTEM_ID>DBDEF</SYSTEM_ID>
    .....
  </ROW>
  <ROW num="3">
    <UNIQ>3</UNIQ>
```

```

    <SYSTEM_ID>DBDEF</SYSTEM_ID>
    .....
  </ROW>
  .....
</ROWSET>

```

この形式のファイルをデータベースに登録するには、以下のコマンドが使えます。

```
java OracleXML putXML -user "HYBS/HYBS" -conn "jdbc:oracle:thin:@hn50g5:1521:HYBS"
-commitBatch 0 -fileName "%1.xml" "%1"
```

同様に、データベースから抜き出すには、以下のコマンドが使えます。

```
java OracleXML getXML -user "HYBS/HYBS" -conn "jdbc:oracle:thin:@hn50g5:1521:HYBS"
-encoding "Shift_JIS" "select * from %1" > %1.xml
```

もちろん、Java で開発ができるため、自分でこれらのAPIを使いながら、登録/更新などを行うクラスを独自に作成することも可能です。

参考:<http://otn.oracle.co.jp/software/tech/xml/xdk/index.html>

5. readTable タグ

readTable タグは、DBTableModel をファイルから読み出すためのタグです。通常は、writeTable タグにより、書き込まれた DBTableModel を再現するために使用されます。ファイルから読み出す場合に、読み出し方法を、readerClass 属性で指定された派生クラスを使用して表示します。

```

<og:readTable
  separator      : 可変長ファイルを作成するときの項目区切り文字をセットします。
  fileURL        : fileURL を セットします。
  filename       : ファイルを作成するときのファイル名をセットします。
  encode         : ファイルを作成するときのファイルエンコーディング名をセットします。
  readerClass    : 実際に読み出すクラス名 (の略称) をセットします。
  maxRowCount    : 読取時の最大取り込み件数をセットします。
  language       : 言語コードを指定します。
  tableId        : session から所得する DBTableModel オブジェクトの ID。
  command        : 処理コマンド
  modifyType     : ファイル取り込み時の モディファイタイプ (A,C,D 属性) を指定します。
/>

```

6. readClass 属性を、以下に示します。

属性クラス	概要説明
Calc	XML パーサによる、OpenOffice.org Calc の表計算ドキュメントファイルを読み取る実装クラスです。
Default	指定の区切り記号(初期値:タブ区切り)ファイルの読み取りクラスです。
Excel	POI による、EXCEL バイナリファイルを読み取る実装クラスです。
Fixed	固定長ファイルの読み取りクラスです。
JExcel	JExcel による EXCEL バイナリファイルを読み取る実装クラスです。

第17章 tableFilter 、process 、schedule

この章では、あまり一般的ではありませんが、タグから利用できる各種クラスについて説明します。

1. tableFilter タグ

tableFilter タグは、TableFilter のサブクラス(classId で指定)に渡して処理を実行します。クラスを作成する場合は、org.opengion.hayabusa.db.TableFilter インターフェースを継承したクラスにする必要があります。また、classId 属性には、システムリソース で設定した TableFilter.XXXX の XXXX を指定します。

```

<og:tableFilter
  classId      = "WL_LOGICSET"           :TableFilter のサブクラス(実行クラス)
  tableId      = "WL0000"               :登録元の DBTableModel の
                                       session/request 変数内の取得キー
  keys         = "AA, BB, CC"           :実行クラスへの引数のキー
  vals         = "{@AA}, {@BB}, {@CC}"  :実行クラスへの引数の値
  selectedAll  = "false/true"           :処理対象の行を全行選択するかどうか
                                       (初期値:false)
  modifyType  = "A/C/D"                 :処理の方法(A:追加 C:更新 D:削除)を
                                       指定します。初期値は自動です。
/>

```

2. mainProcess タグ、process タグ

mainProcess タグは、HybsProcess を継承した、ParamProcess,FirstProcess,ChainProcess の実装クラスを実行する MainProcess を起動するクラスです。

LoggerProcess は、最初に定義するクラスで、画面ログ、ファイルログ、を定義します。

また、エラー発生時に、指定のメールアドレスにメール送信できます。

Process_Logger は、なくても構いませんが、指定する場合は、最も最初に指定しなければなりません。

ParamProcess は、一つだけ定義できるクラスで、データベース接続情報を定義します。

(データベース接続しなければ)なくても構いません。

FirstProcess は、処理を実行する最初のクラスで、このクラスでデータが作成されます。

ループ処理は、この FirstProcess で順次作成された LineModel オブジェクトを1行ずつ下位の ChainProcess に流していきます。

ChainProcess は、FirstProcess で作成されたデータを、受け取り、処理します。

処理対象から外れる場合は、LineModel を null に設定する為、下流には流れません。

フィルタチェーンの様に使用します。なくても構いませんし、複数存在しても構いません。

第17章 tableFilter 、process 、schedule

process タグは、org.opengion.fukurou.process パッケージの HybsProcess インターフェースを実装したクラスの、Process_XXXX.java の XXXX 部分を指定します。

共通的なパラメータは、この Tag クラスに実装しますが、それぞれ、個別に必要なパラメータは、ParamTag を使用して指定します。

このタグは、MainProcess タグの内部にのみ、記述可能です。

```
<og:mainProcess
  useJspLog="[true/false]"
  useDisplay="[true/false]" >
  <og:process processID="ZZZ" >
    <og:param key="AAA" value="111" />
  </og:process >
</og:mainProcess >
```

processID	概要説明
BulkQuery	データベースから読み取った内容を、一括処理するために、ParamProcess のサブクラス(Process_DBParam)にセットしたり、加工したりする FirstProcess と、ChainProcess のインターフェースを両方持った、実装クラスです。
DBCountFilter	データベースの存在件数でフィルタリングする ChainProcess インターフェースの実装クラスです。
DBMerge	UPDATE と INSERT を指定し データベースを追加更新する、ChainProcess インターフェースの実装クラスです。
DBParam	他のプロセスへ共通のデータベース接続を割り当てる為の、ParamProcess インターフェースの実装クラスです。
DBReader	データベースから読み取った内容を、LineModel に設定後、下流に渡す、FirstProcess インターフェースの実装クラスです。
DBWriter	上流から受け取ったデータをデータベースに書き込む GainProcess インターフェースの実装クラスです。
FileCopy	上流から受け取った FileLineModel を処理する、ChainProcess インターフェースの実装クラスです。
FileFtp	上流から受け取った FileLineModel を処理する、ChainProcess インターフェースの実装クラスです。
FileSearch	指定のフォルダ以下のファイルを一覧する、FirstProcess インターフェースと、ChainProcess インターフェースの実装クラスです。
Grep	上流から受け取った FileLineModel から、文字列を見つけ出す ChainProcess インターフェースの実装クラスです。
GrepChange	上流から受け取った FileLineModel から、語句を置換する、ChainProcess インターフェースの実装クラスです。
GrepChangeExcel	上流から受け取った FileLineModel から、語句を置換する、ChainProcess インターフェースの実装クラスです。
LDAPReader	LDAP から読み取った内容を、LineModel に設定後、下流に渡す、FirstProcess インターフェースの実装クラスです。
Logger	画面出力、ファイルログ、エラーメールを管理する、ロギング関係の

	LoggerProcess インターフェースの実装クラスです。
StringUtil	上流から受け取ったデータを StringUtil クラスの特定のメソッドでデータ変換する、CainProcess インターフェースの実装クラスです。
TableDiff	ファイルから読み取った内容を、LineModel に設定後、下流に渡す、FirstProcess インターフェースの実装クラスです。
TableFilter	上流から受け取ったデータをフィルタする、ChainProcess インターフェースの実装クラスです。
TableReader	ファイルから読み取った内容を、LineModel に設定後、下流に渡す、FirstProcess インターフェースの実装クラスです。
TableWriter	上流から受け取ったデータをファイルに書き込む CainProcess インターフェースの実装クラスです。
XSLT	XSLT 変換結果を指定のファイルに出力します。

3. schedule タグ

アプリケーション共有のタイマー機能を用いて、繰り返しスケジュールを設定します。タイマースケジュールは、帳票デーモンや、再編成処理、定期的バッチ処理など、エンジン上のスレッドで動作するスケジュールを登録します。スケジュールすべきクラスは、timerTask 属性に、フルクラス名で指定します。そして、org.opengion.fukurou.util.HybsTimerTask を拡張 (extends) しておく必要があります。

設定例

```
<og:schedule
  command      = "{@CMD}"
  scope        = "request"
  timerTask    = "org.opengion.hayabusa.report.ReportDaemon"
  name         = "{@NAME}"
  comment      = "Start-up By {@USER.INFO}"
  singleton    = "true"
  delay        = "0"
  period       = "{@PERIOD}000"
  fixedDelay   = "true"
  keys         = "SYSTEM_ID, DMN_GRP"
  vals         = "{@SYSTEM_ID}, {@DMN_GRP}"
/>
```

属性の説明

<og:schedule	
command	コマンド (SET, VIEW, REMOVE, CANCEL) をセットします (初期値: SET)。
timerTask	処理を実行するクラス名 (HybsTimerTask のサブクラス) を指定します。(必須)
name	名称 (ユニークキー) を設定します。(必須)
comment	タイマータスクの説明を設定します。
singleton	Name 属性が同一のタスクを 2 重登録出来ないよう (true/false) にします (初期値: true [出来ない])。

第17章 tableFilter 、process 、schedule

delay	ミリ秒単位の遅延時間を設定します(初期値:0)。
period	ミリ秒単位の繰り返し間隔を設定します(初期値:60000)。
fixedDelay	固定遅延実行 (true)/固定頻度実行(false)を設定します (初期値:true)。
keys	HybsTimerTask に渡す為のキー情報、CSV 形式で指定します。
vals	HybsTimerTask に渡す為の値を、CSV 形式で複数指定します。
startTime	24 時間制 (YYMMDD) の開始時刻を設定します(初期値:000000)。
stopTime	24 時間制 (YYMMDD) の終了時刻を設定します(初期値:000000)。
tableId	(通常は使いません) 結果を DBTableModel に書き込んで、 session に登録するときのキーを指定します。
scope	キャッシュする場合のスコープ (local, request, page, session, applicaton)を指定します (初期値:local)。

> ... Body ...

</og:schedule >

org.opengion.plugin.daemon 以下のクラス。

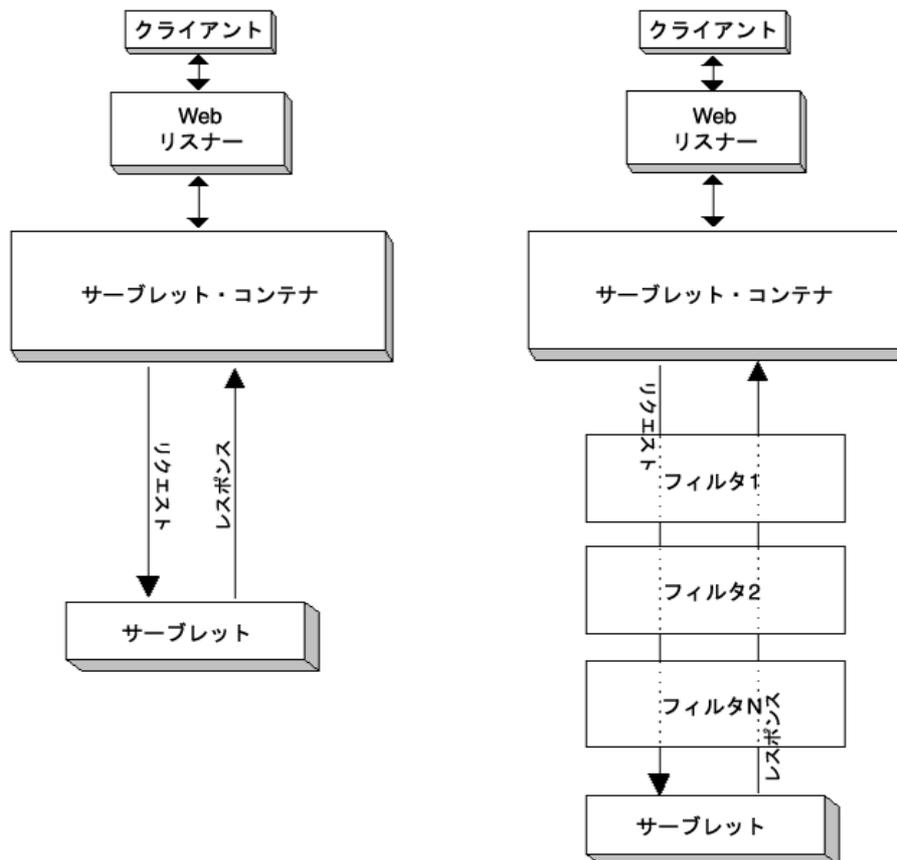
timerTask 属性に設定する場合は、フルパス指定が必要です。

属性クラス	概要説明
MailReceive	【メールデーモン】メールサーバーを監視して、EXCEL ファイルの DB 登録処理のデーモンです。
Report	【レポート出力】帳票要求テーブルを監視して、帳票処理プログラムを呼び出します。
Report2	【レポート出力】帳票要求テーブルを監視して、帳票処理プログラムを呼び出します。
Transfer	【伝送システム】各読取方法、実行方法に応じて伝送処理を行うためのデーモンです。
Transfer_CB01	【伝送システム】旧伝送 DB(CB01)を監視して、実行方法に応じた処理プログラムを呼び出します。
URLConnect	【URL アクセス】 指定したパラメータで URL に接続します。

第18章 filter (Servlet のフィルター機能の実装)

この章では、Servlet の標準機能である、フィルターの実装クラスについて説明します。フィルターについては、タグリブリファレンスではなく、JavaDoc で出力された API リファレンスの `org.opengion.hayabusa.filter` パッケージを参照してください。

参考文献: **Oracle Containers for J2EE サブレット開発者ガイド 10g(10.1.3.1.0)**
(http://docs.oracle.com/cd/E18355_01/Web.1013/B31859-01/filters.htm#554770)



1. Filter の設定

filter を使用するには、opengion/uap/Webapps/gf/WEB-INF/Web.xml にて、<filter>と、<filter-mapping> の設定をする必要があります。
 フレームワークでは、あらかじめ、利用可能なフィルターの設定サンプルが、コメントアウトされて記述されていますので、コメントを外せば、初期設定状態ですぐに使用できます。

例:

```
<filter>
  <filter-name>AccessStopFilter</filter-name>
  <filter-class>org.opengion.hayabusa.filter.AccessStopFilter</filter-class>
  <init-param>
    <param-name>startTime</param-name>
    <param-value>070000</param-value>
  </init-param>
  <init-param>
    <param-name>stopTime</param-name>
    <param-value>220000</param-value>
  </init-param>
  <init-param>
    <param-name>filename</param-name>
    <param-value>jsp/custom/stopFile.html</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>AccessStopFilter</filter-name>
  <url-pattern>/jsp/*</url-pattern>
</filter-mapping>
```

org.opengion.hayabusa.filter 以下のクラス。

filter	概要説明
AccessStopFilter	AccessStopFilter は、Filter インターフェースを継承したアクセス制御クラスです。
FileFilter	Filter インターフェースを継承した HTML デモ画面を作成するフィルタクラスです。
GZIPFilter	GZIPFilter は、Filter インターフェースを継承した ZIP 圧縮クラスです。
URLCheckFilter	URLCheckFilter は、Filter インターフェースを継承した URL チェッククラスです。
URLHashFilter	URLHashFilter は、Filter インターフェースを継承した URL チェッククラスです。

第19章 高度なデータアクセス

この章では、通常のSQLによるデータ取り出しと、カラムオブジェクトの機能を組み合わせることによって、高速で汎用性の高いデータ取り出し方法について、説明します。

一般に、Web アプリケーションは、Web サーバーとデータベースサーバーという2階層か、それ以上の階層で構築されています。

この多階層化のメリットのうち、容易にスケーラビリティを確保できるということがあります。データベースサーバーのクラスタ化は、非常に高度な技術が要求され、かつ高価になる傾向があるため、Web サーバー側への負荷分散をまずは考慮すべきでしょう。

そうする場合に、Web サーバー側とデータベースサーバー側の負荷の振分においては、できるだけ Web サーバー側での処理を行い、データベースサーバー側は必要最小限、つまりデータの検索に特化するようにシステムを構築しておけば、負荷が増えてきた場合には、まず、Web サーバーの増設を行うだけで簡単に対応できます。

本フレームワークでは、検索結果のデータに対して、カラムオブジェクトを適用させることで各種チェックや表示形式の変換などを、データベースサーバーではなく、Web サーバーで実行しています。また、検索結果は、Web サーバーの DBTableModel にキャッシュし、管理しているため、次ページ操作などは、データベースにアクセスすることなく処理できます。同様に、検索結果に対する追加、修正、削除も、Web サーバー上で行った後、変更データ分のみをデータベースサーバーに登録するため、データベースサーバー側の負荷を低減できます。

ただし、Web サーバーには、十分なメモリを確保しておく必要があります。

次に、一般的な検索と表示の例を示します。

例)

```
<og:query
  command          = "{@command}"
  maxRowCount      = "10000"
>
  SELECT * FROM CUSTOMER
</og:query>

<og:view
  viewFormType = "HTMLTable"
  command      = "{@command}"
  pageSize     = "100"
/>
```

query タグの `maxRowCount = "10000"` は、データベースから 最大10000行を検索することを指定しています。これは、誤って大量のデータ検索を行った場合に、Web サーバー側のメモリ不足を引き起こさないための配慮です。

デフォルト設定は、`SystemResource.properties` の `DB_MAX_ROW_COUNT` パラメータで指定できます。

view タグの `pageSize = "100"` は、画面上に表示する1ページ分の行数です。つまり、先に最大件数分を検索した場合(10000件検索)100件ずつ画面に表示されます。このとき、画面を、NEXT, PREVした場合でも、データベースへの検索を行わずに、Web サーバー側のメモリ上に存在している `DBTableModel` を再表示しているだけです。

また、画面に表示する場合に、数字であればカンマ編集や、日付であれば、YYYY/MM/DD の表示形式の変換は、データベースの検索結果に Web サーバーのカラムオブジェクトの表示用レンダラーを適用して処理しています。

例えば、10000件検索したとしても、画面上に100件検索しているだけです。この100件分のみ、表示用レンダラーが適用されます。残りの9900件は、表示されるまで変換されません。

この特性を利用して、検索結果の表示処理(表示用レンダラー)で、特殊処理を行うことにより、パフォーマンスを上げることが可能になります。

以下に、その例となるレンダラーを取り上げます。

1. FORM レンダラー

FORM レンダラーは、パラメータで指定された FORM を表示するクラスです。元の Value を、`$1` として、使用することが可能です。

例)

`DBCOLUMNResource.properties` に、FORM レンダラーの指定とパラメータに下記の様に指定する。

```
<a href=" ../TEST14/index.JSP?           //1行で記述
      command=NEW&                       //1行で記述
      SYSTEM_ID=DBDEF&                   //1行で記述
      TABLE_NAME=$1"                   //1行で記述
      target="CONTENTS" >$1</a>
```

これは、検索結果の値を、`$1` の箇所に埋め込んで、リンク情報を作成します。先に述べましたように、この変換処理は、画面に表示するときのみ、行われます。

FORMレンダラーのパラメーターは何でも良く、`image` タグと組み合わせたリンクや定型の文書などでも構いません。

2. QUERY レンダラー

QUERY レンダラーは、パラメータで指定された SQL 文を実行し、その結果を表示するクラスです。

元の Value を、\$1 として、使用することが可能です。

例)

DBColumnResource.properties に、QUERY レンダラーの指定とパラメータに下記の様に指定する。

```
Select count(*) from $1
```

これは、検索結果の値を、\$1 の箇所に埋め込んで、テーブルのカウントを取ります。先に述べましたように、この検索処理は、画面に表示するときのみ、行われます。

例)

DBColumnResource.properties に、DBCOUNT というカラムを定義する。

```
DBCOUNT=VARCHAR2 128 true QUERY TEXT XK _ "select count(*) from $1"
```

検索は、TABLE_NAME に別名として DBCOUNT をつける。

元のテーブル名も必要なので、検索しておく。

<og:query

```
command="{@command}"
SELECT TABLE_NAME ,
TABLE_NAME DBCOUNT
FROM USER_TABLES
ORDER BY TABLE_NAME
```

</og:query>

<og:view

```
viewFormType = "HTMLTable"
command = "{@command}"
```

/>

No	テーブル名	DBCOUNT
<input type="checkbox"/> 1	DB01	4
<input type="checkbox"/> 2	DB02	12
<input type="checkbox"/> 3	DB03	125
<input type="checkbox"/> 4	DB04	263
<input type="checkbox"/> 5	DB05	302
<input type="checkbox"/> 6	DB06	24
<input type="checkbox"/> 7	DB07	49
<input type="checkbox"/> 8	DB08	184
<input type="checkbox"/> 9	DB09	30
<input type="checkbox"/> 10	DB10	11
<input type="checkbox"/> 11	DB11	140
<input type="checkbox"/> 12	DB12	302
<input type="checkbox"/> 13	DB13	4
<input type="checkbox"/> 14	GE50_OLD	7
<input type="checkbox"/> 15	PLAN_TABLE 2	
<input type="checkbox"/> 16	USERS	1
<input type="checkbox"/> 17	USER_ROLES	1

第 VI 部 ダイレクト・パス・インポートとダイレクト・パス・エクスポート

『うまく使えば、時間はいつも十分にある』
ゲーテ

ここでは、ダイレクトパスについて説明します。
構成は、次のとおりです。

第 20 章 ダイレクト・パス・インポート

通常の DBTableModel に一旦キャッシュする方式では、巨大なデータをすべてメモリ中に管理することができないため、直接ファイルからデータベースに登録する方法について説明します。

第 21 章 ダイレクト・パス・エクスポート

同様に、直接データベースからファイルに抜き出す方法について説明します。

第20章 ダイレクト・パス・インポート

この章では、通常の DBTableModel に一旦キャッシュする方式では、巨大なデータをすべてメモリ中に管理することができないため、直接ファイルからデータベースに登録する方法について説明します。

通常の readTable などは、DBTableModel オブジェクトを介して全件メモリにロードしてから表示させる為、大量データ処理ができません。このタグでは、直接ファイルを読み取りながらデータベース登録するので大量データをバッチ的に登録する場合に使用します。

```
<og:directTableInsert
  dbid          = "ORCL"          接続データベース ID(初期値: DEFAULT)
  separator     = ","           ファイルの区切り文字 (初期値: タブ)
  fileURL       = "{@USER. ID}"  読み取り元ディレクトリ名
  filename      = "{@filename}"  読み取り元ファイル名
  encode        = "Shift_JIS"    読み取り元ファイルエンコード名
  displayMsg    = "MSG0040"      登録完了後のメッセージ
  columns       = "CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG"
                                     #NAME の代わりに使用するカラム列名
  commitBatch  = "100"          この件数ずつコミットを発行(初期値: 無制限)
  useColumnCheck = "true"       カラムチェックを行うかどうか(初期値: false)
  useColumnAdjust = "true"      カラム変換を行うかどうか(初期値: false)
  nullCheck     = "CLM, SYSTEM_ID" NULL チェックを実行します。
>
  INSERT INTO GE41
    (CLM, NAME_JA, LABEL_NAME, KBSAKU, SYSTEM_ID, LANG,
     FGJ, DYSET, DYUPD, USRSET, USRUPD, PGUPD)
  VALUES
    ([CLM], [NAME_JA], [LABEL_NAME], [KBSAKU], [SYSTEM_ID], [LANG],
     ' 1', '{@USER. YMDH}', '{@USER. YMDH}', '{@USER. ID}', '{@USER. ID}',
     '{@GUI. KEY}')
</og:directTableInsert >
```

第21章 ダイレクト・パス・エクスポート

この章では、同様に、直接データベースからファイルに抜き出す方法について説明します。中間の、データ(DBTableModel)を作成しないため、余計なメモリを取らず、高速にデータを抜き出すことが可能です。一方、抜き出すデータは生データのため、データの再利用等、システム的な使用を想定しています。

JDBCErrMsg 形式の PL/SQL をコールして、その検索結果(カーソル)を抜きすることもできます。

```

<og:directWriteTable
  dbid          = "ORCL"          接続データベース ID(初期値 : DEFAULT)
  separator     = ", "           ファイルの区切り文字 (初期値 : タブ)
  fileURL       = "{@USER. ID}"  保存先ディレクトリ名
  filename      = "{@filename}"  保存ファイル名
  encode        = "UnicodeLittle" 保存ファイルエンコード名
  useHeader     = "true"         保存ファイルにヘッダーを出力するか
  zip           = "true"         ZIP ファイルに圧縮するかどうか
  zipFilename   = "Sample. zip"  ZIP ファイルのファイル名
  fileAppend    = "true"         ファイルを追加モードで登録するかどうか
  displayMsg    = "MSG0033"     実行後の表示メッセージ
  fetchSize     = "200"         DB 検索する場合のフェッチするサイズ
>
      SELECT * FROM ZYXX
</og:directWriteTable >

<og:directWriteTable
  fileURL       = "{@USER. ID}"  保存先ディレクトリ名
  filename      = "{@filename}"  保存ファイル名
  names         = "AAA, BBB, CCC, ..." 指定のキーに対応するリクエスト値を
                                         ARG_ARRAY にセットします。
  queryType     = "JDBCErrMsg"  JDBCErrMsg 形式の PL/SQL をコールします。
>
  { call PL/SQL(?, ?, ?, ? ) }
</og:directWriteTable >

```

第 VII 部 アプリケーションの保護

『弓の弦は二本持っていなければならぬ』
シャルル・ド・ボヴェル

ここでは、アプリケーションの保護について説明します。
構成は、次のとおりです。

第 22 章 アプリケーション・アクセスの制御
アプリケーション・リソースへのユーザー・アクセスを制御する方法について説明します。

第 23 章 権限、ロールおよびセキュリティ・ポリシー
ユーザーの設定と画面の設定方法について説明します。

第 24 章 アプリケーション監査
使用メモリ、ログインユーザーの監視、および、キャッシュクリア、接続の破棄等管理画面の操作を中心に説明します。

第22章 アプリケーション・アクセスの制御

この章では、アプリケーション・リソースへのユーザー・アクセスを制御する方法について説明します。

アプリケーションアクセスに関する制御には、IPアドレスによる制限や、ドメインユーザーなどの認証に関する制限などの、半固定にちかい制限と、アプリケーションレベルの流動性の高い、いわば、その都度変更できる制限があります。

ここでは、アプリケーションレベルでの制限について、述べます。

Web アプリケーションフレームワークでは、すべてのログインユーザーを管理する必要があります。それは、例えば、ユーザーが自らログインせずに、GUESTユーザーとして自動的に扱われたとしても、同様です。

ユーザー情報は、セッションごとに作成されます。よって、同一ユーザー（例えば、GUEST）が同時に複数ログインしたとしても、セッションが別であれば、別人として扱われます。

（ただし、GUESTユーザーの属性は同一です。また、テンポラリファイル等を使用する場合は、一般的なアプリケーションの作りをしている場合は、同一になります。よって、そのような使い方……つまり、書込みを行う場合は、きちんとログインユーザーを分けるべきです。）

本フレームワークでは、ユーザー認証と、画面等のアクセス制限を分けて管理しています。ユーザー認証は、Tomcat 等のサーブレットコンテナに任せて、アクセス制限を、ユーザーリソースと GUI リソースを用いて処理しています。

1. ユーザー認証

ユーザー認証は、Tomcat 等のサーブレットコンテナで行っています。

以下、Tomcat を例に説明します。

設定ファイルは、opengion/apps/tomcat7.0.27/conf/server.xml の Realm で設定します。

デフォルトの UserDatabaseRealm (conf/tomcat-users.xml) を使用するのではなく、JDBCRealm を用いてデータベース上のユーザーで認証を行います。

このテーブルと、後ほど説明するアクセス制限用のユーザーテーブルを同一にすることで、メンテナンス工数の削減が可能になります。

```
<Realm className="org.apache.catalina.realm.JDBCRealm"
        driverName="${connectionDriver}"
        connectionURL="${connectionUrl}"
        connectionName="${connectionName}"
        connectionPassword="${connectionPassword}"
        userTable="GEA10V01" userNameCol="USERID" userCredCol="PASSWD"
        userRoleTable="GEA10V01" roleNameCol="SYSTEM_ID"
/>
```

ここでは、GEA10V01 というビューを作成して、ユーザーテーブルを Realm の形式に合わせます。こうすることで、ユーザーテーブルが変更されても、または、システム毎に異なっても、sarver.xml を書き直す必要がなくなります。

2. セキュリティ制限

セキュリティ制限は、JSP/Servlet 準拠の Web.xml で対応しています。

先のユーザー認証は、サーブレットコンテナ (例 Tomcat) ごとに異なりますが、Web.xml は JSP の規格に準拠していますので、共通に使用できます。

下記の例は、gf というサンプルシステムの Web.xml です。

opengion/uap/Webapps/gf/WEB-INF/Web.xml より抜粋

```
<security-constraint>
  <Web-resource-collection>
    <Web-resource-name>Protected JSP Area</Web-resource-name>
    <url-pattern>/JSP/*</url-pattern>
    <url-pattern>/help/*</url-pattern>
    <url-pattern>/filetemp/*</url-pattern>
    <http-method>DELETE</http-method>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
    <http-method>PUT</http-method>
  </Web-resource-collection>
  <auth-constraint>
    <role-name>*</role-name>
  </auth-constraint>
</security-constraint>

<security-constraint>
  <Web-resource-collection>
    <Web-resource-name>Protected Other Area</Web-resource-name>
    <url-pattern>/*</url-pattern>
  </Web-resource-collection>
  <auth-constraint>
    <role-name></role-name>
  </auth-constraint>
</security-constraint>

<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Web App</realm-name>
</login-config>

<security-role>
  <role-name>*</role-name>
```

```

</security-role>
<security-role>
  <role-name></role-name>
</security-role>
<security-role>
  <role-name>manager</role-name>
</security-role>

```

3. directory listings

Web サーバーで、あるディレクトリにアクセスしたときに、そのディレクトリ一覧を表示させるのか、または、表示させないのかを指定するには、default servlet の設定を、行う必要があります。

listings 属性を false に設定すると、ディレクトリの表示を行わないようになります。

なお、ディレクトリ表示を行う場合、welcome-file の設定があると、そのファイルを表示してしまいますので、ご注意ください。

これらの設定は、Tomcat 側の共通設定で行います。

```
opengion/apps/tomcat7.0.27/conf/Web.xml
```

```

<servlet>
  <servlet-name>default</servlet-name>
  <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class> ※1
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <init-param>
    <param-name>listings</param-name>
    <param-value>>false</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

```

※1. openGion フレームワークでは、この Web.xml をカスタマイズしています。

1. org.apache.catalina.servlets.DefaultServlet の代わりに、org.opengion.tomcat.CacheDefaultServlet を指定。
2. <error-page> で、/JSP/common/error.JSP を指定。

以前のバージョンでは、<mime-mapping> も必要な項目を追加していました。

4. Digest 認証

先の認証の例は、BASIC 認証といわれる方式で、ユーザー名とパスワードを Base64 形式でエンコードして送信しますので、転送中に傍受されると容易に解読されてしまいます。

DIGEST 認証は、無作ために生成した文字列(ナンス)をクライアントに返し、それを用いてURL, HTTP方式、ナンスから一方向ハッシュ(MD-5)を作成し、サーバーは、独自のチェックサムで比較して判定します。

例: BASIC 認証時のヘッダー情報から、ユーザー/パスワードの取得方法

```
String authorization = request.getHeader("Authorization");
String userInfo = authorization.substring(6).trim();
BASE64Decoder decoder = new BASE64Decoder();
String nameAndPassword = new String(decoder.decodeBuffer(userInfo));
int index = nameAndPassword.indexOf(":");
String user = nameAndPassword.substring(0, index);
String password = nameAndPassword.substring(index+1);
```

Digest 認証の設定は、システム側の Web.xml で記述します。

opengion/uap/Webapps/gf/WEB-INF/Web.xml より抜粋

```
<login-config>
  <auth-method>DIGEST</auth-method>
  <realm-name>Web App</realm-name>
</login-config>
```

5. HTTPS クライアント認証

SSL(Secure Sockets Layer)は、公開キーと対象キー暗号を組み合わせてセキュアセッションを確立します。

opengion/apps/tomcat7.0.27/conf/server.xml の下記のコメントアウトされている箇所を使用します。

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
  maxThreads="150" scheme="https" secure="true"
  clientAuth="false" SSLProtocol="TLS" />
```

さらに UAP¥Webapps¥システムID ¥WEB-INF¥Web.xml の security-constraint に、user-data-constraint を追加します。

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Protected JSP Area</web-resource-name>
```

```
<url-pattern>/JSP/*</url-pattern>
</Web-resource-collection>
<auth-constraint>
  <role-name>*</role-name>
</auth-constraint>
<user-data-constraint>
  <transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
</security-constraint>
```

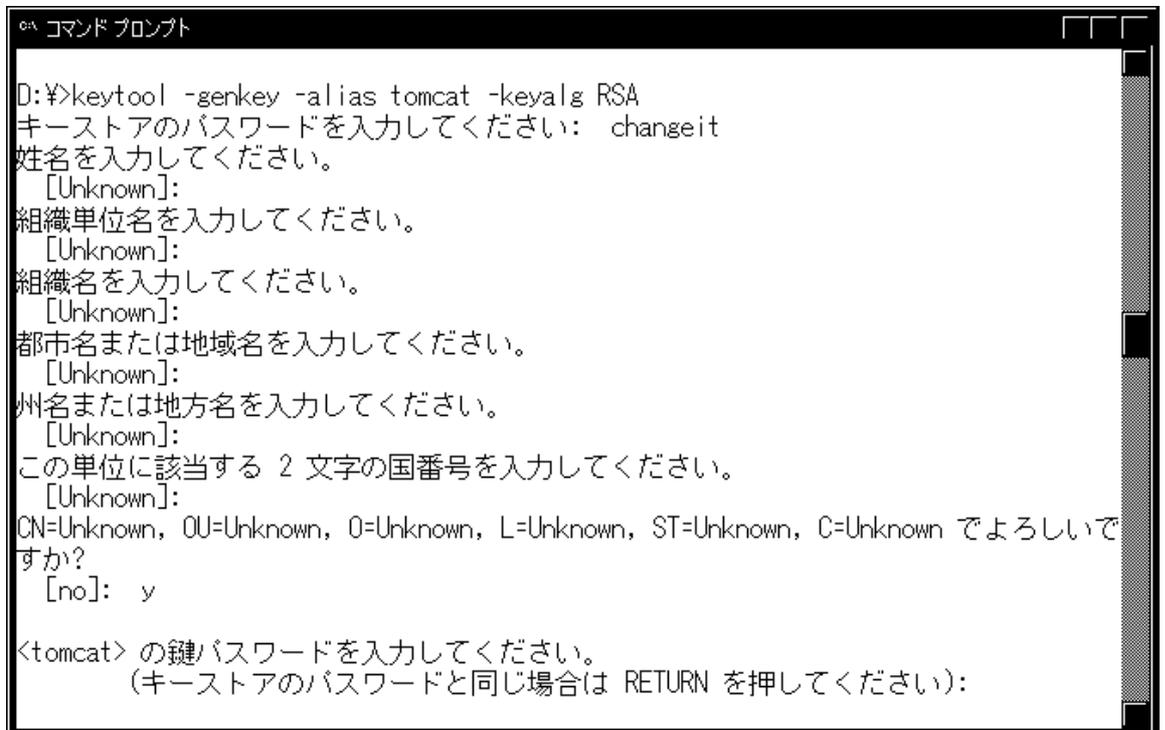
SSL は、JDK1.4 より標準パッケージになりましたので、そのまま使用できます。それ以前のJDKでは、JavaTM Secure Socket Extension (JSSE) をインストールする必要があります。

参考:<http://java.sun.com/products/jsse/>

注意:Web エンジン Ver 3.0 では、JDK1.4 以上となっています。

エイリアス『tomcat』に公開キーと秘密キーを生成する必要があります。これは、JDKの keytool を使用して、作成してください。

```
keytool -genkey -alias tomcat -keyalg RSA
```



```
コマンド プロンプト
D:\>keytool -genkey -alias tomcat -keyalg RSA
キーストアのパスワードを入力してください: changeit
姓名を入力してください。
 [Unknown]:
組織単位名を入力してください。
 [Unknown]:
組織名を入力してください。
 [Unknown]:
都市名または地域名を入力してください。
 [Unknown]:
州名または地方名を入力してください。
 [Unknown]:
この単位に該当する 2 文字の国番号を入力してください。
 [Unknown]:
CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown でよろしいで
すか?
 [no]: y

<tomcat> の鍵パスワードを入力してください。
(キーストアのパスワードと同じ場合は RETURN を押してください):
```

第23章 権限、ロールおよびセキュリティ・ポリシー

この章では、ユーザーの設定と画面の設定方法について説明します。

1. 画面メニューのアクセス制限

画面メニューのアクセス制限は、ユーザーリソースとGUIリソースの設定により、個々の画面に対して、読取許可、書込み許可を与えることが可能です。

ユーザーリソースのロール、グループ、プロジェクトと、GUI リソースのロール、グループ、プロジェクト、アクセスモードで判断されます。

以下に、アクセスチェックの手順を示します。

1. ユーザー情報のロールが、"root" であれば、アクセスモードの rw 属性のみで判定
2. アクセスモードのその他 (rwrwrwrw の最後の rw) のチェックで判定
3. 以下の判定は、全てが成立している必要がある。
 - (ア) ロール、グループ、プロジェクトのロールのどれかに r 属性または w 属性のいずれかが設定されている。
 - (イ) 画面ロールがあり、かつ、ユーザーロールを含む。
 - (ウ) 画面グループがあり、かつ、ユーザーグループを含む。
 - (エ) 画面プロジェクトがあり、かつ、ユーザープロジェクトを含む。

上記判定で、アクセス許可が与えられます。

2. 画面の登録ボタンのアクセス制限

アクセスモードの rw 属性は、読取許可 (r) と書込許可 (w) で指定します。
制限をかける場合は、- を記入してください。

例)	rwrwrwrw	フルアクセス(その他モードも rw のため)
	rwrwrw--	指定のユーザーのみアクセス可能
	r-r-r-r-	すべてで、読取専用
	-w-w-w-w	書き込み専用・・・メニューには現れません。

個々の画面のロール、グループ、プロジェクトと、ユーザーのロール、グループ、プロジェクトを制御することで、さらに画面単位にユーザーごとのアクセス制限をかけられます。

また、読取権限のみに制限されている場合は、登録ボタンが表示されません。
登録ボタンの制限は、JSP 画面の、writeCheck タグで行っています。

例) writeCheck による書込制限判定

<!-- 追加、複写、変更、削除ボタンを作成します。 -->

```
<mis:writeCheck>
  <mis:input type="submit" name="command" value="COPY"
    msg="MSG0035" accesskey="C" td="false" />
  <mis:input type="submit" name="command" value="MODIFY"
    msg="MSG0036" accesskey="M" td="false" />
  <mis:input type="submit" name="command" value="DELETE"
    msg="MSG0037" accesskey="D" td="false" />
  <JSP:directive.include file="../../common/Excel_direct.JSP" />
</mis:writeCheck>
```

このタグで囲われた範囲は、書込み許可が与えられない場合は、表示されません。

第24章 アプリケーション監査

この章では、使用メモリ、ログインユーザーの監視、および、キャッシュクリア、コネクションの破棄等管理画面の操作を中心に説明します。

稼働状況は、org.opengion.hayabusa.servlet.HybsAdmin サンプルで表示しています。これに、引数を与えることで、各種情報を表示させることが可能です。

http://localhost:8824/gf/JSP/admin?COMMAND=コマンド

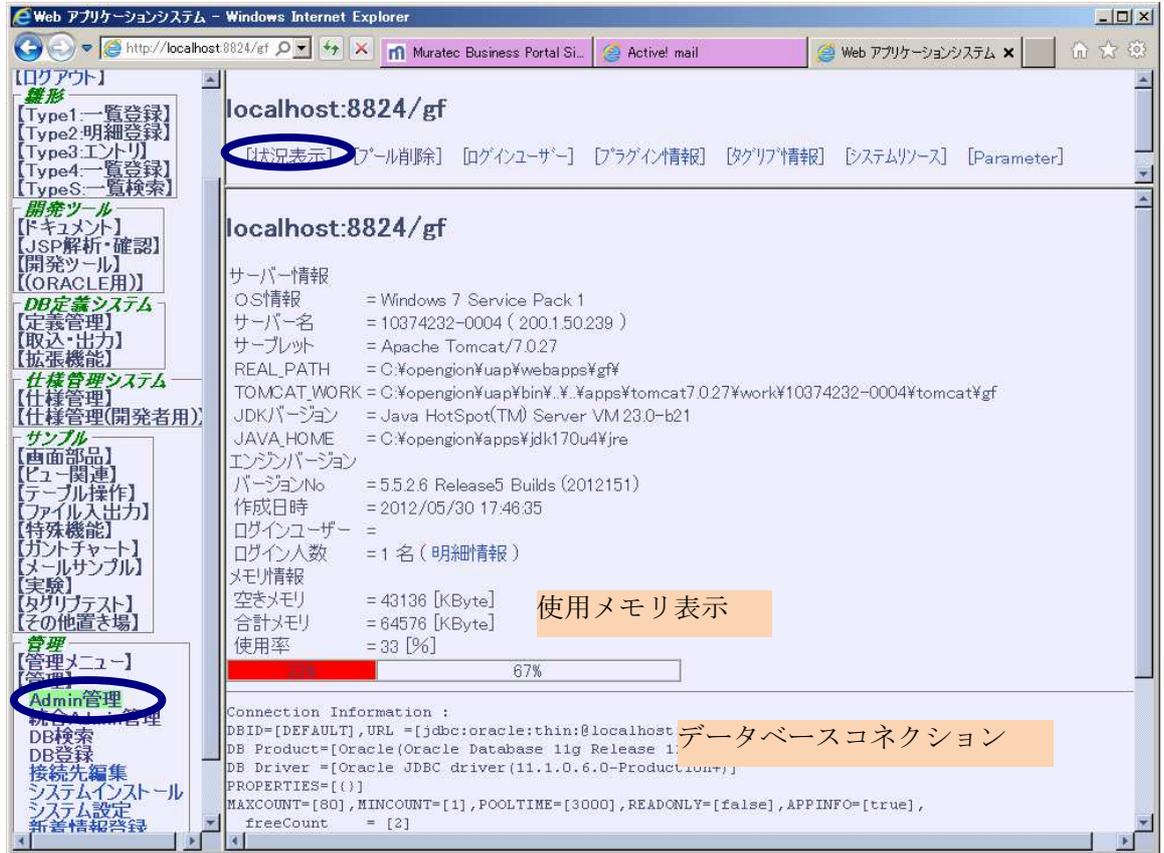
名称	COMMAND	説明
状況表示	infomation	サーバー情報、エンジンバージョン、メモリ情報など、システムの各種情報を表示します。
プール削除	close	リソース情報のキャッシュを全てクリアします。
ログインユーザー	loginUser	現在のログインユーザーの明細情報を表示します。
プラグイン情報	plugin	現在のプラグインのバージョン情報を表示します。
タグリブ情報	taglib	現在のタグリブのバージョン情報を表示します。
システムリソース	systemResource	現在のシステムリソースの設定情報を表示します。
Parameter	(admin サンプルでない)	システムで使われる{@XXXX}の実データをサンプル表示します。
アクセスストップ	AccessStop	アクセスストップフィルターの制御(停止、許可)を行います。

1. 状況表示

稼働しているサーバーの情報、Tomcat や Java のバージョン、使用メモリ、および、使用データベースコネクションについての情報は、リソース管理メニューの状況表示より検索できます。

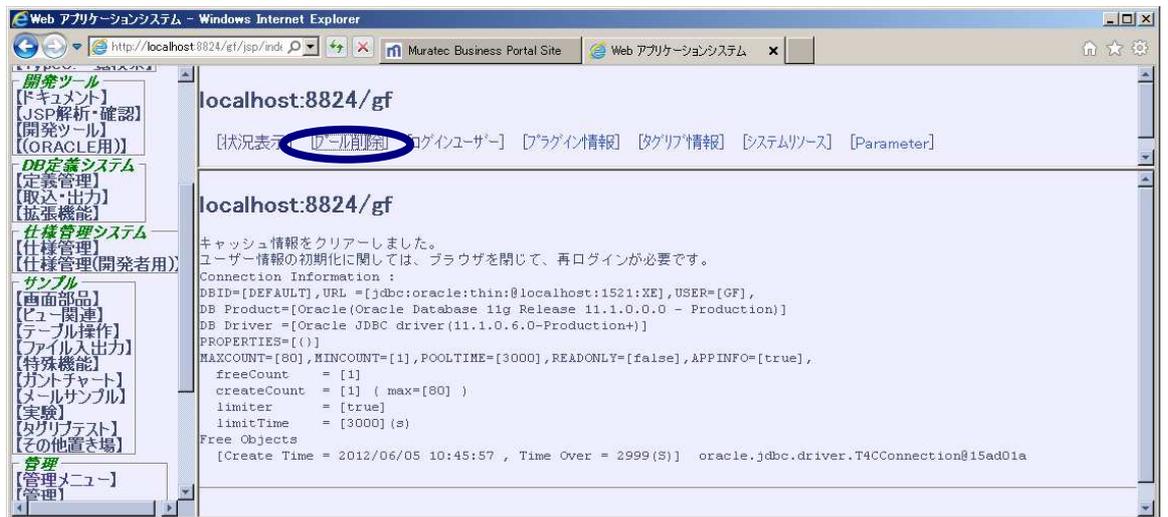
使用メモリの算出方法は、Runtime オブジェクトより取得します。

```
int freeMemory = (int) (Runtime.getRuntime().freeMemory()/1024) ;
int totalMemory = (int) (Runtime.getRuntime().totalMemory()/1024);
int useMemoryRatio = (int) (((totalMemory - freeMemory) *100)/totalMemory) ;
```



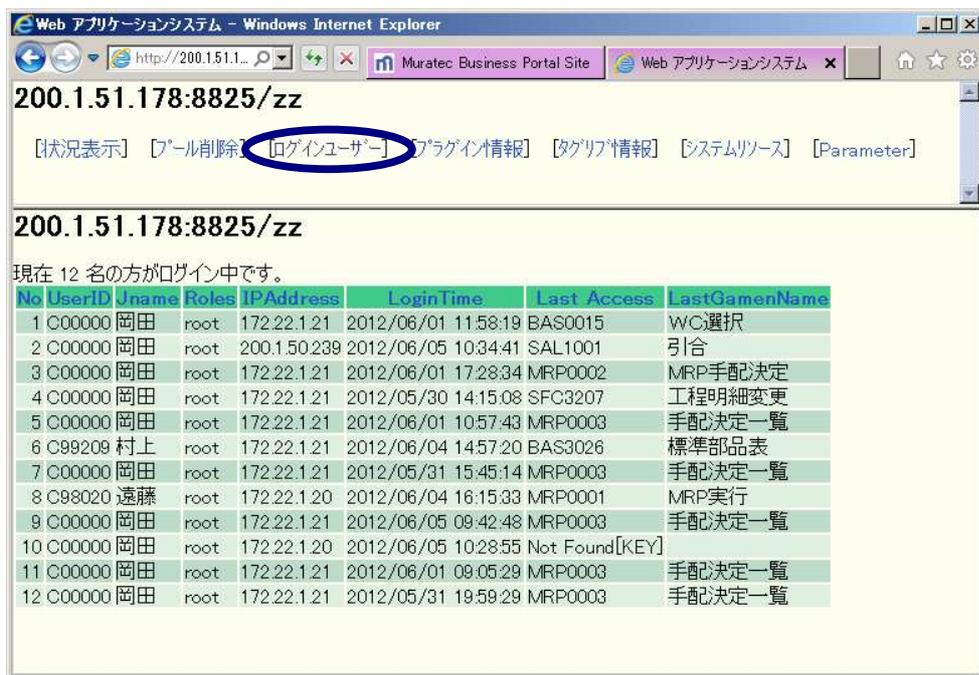
2. プール削除

内部でキャッシュしているリソース情報や、データベース接続キャッシュを破棄します。データベースコネクション上でエラーが発生した場合は、自動的に破棄しますが、想定ケース以外の状況で、不正なセッションが残ったままの場合は、エラーになります。その場合は、セッション破棄で、クリアする必要があります。



3. ログインユーザー

現在ログインしているユーザーの一覧を表示します。
 ログインしている状態とは、セッションが残っている状態のセッション数を指します。
 つまり、同一人物が複数セッションでログインしていても、セッション分(複数)接続とみなします。



4. プラグイン情報

org.opengion.plugin 以下のパッケージに存在するプラグイン一覧を表示します。
 これは、各種タグ上から呼び出すことができるクラス名と関連付いています。
 詳細は、開発ツールのドキュメントより、確認できます。

5. タグリブ情報

org.opengion.hayabusa.taglib 以下のパッケージに存在するタグリブ一覧を表示します。
 実際に利用できるのは、このカスタムタグ化されたタグだけです。
 詳細は、開発ツールのドキュメントより、確認できます。

6. システムリソース

システムリソースに登録されている状態を一覧表示します。
 データそのものは、管理メニューのシステム定数から検索・登録できます。
 現時点のシステムの設定情報が表示されます。
 これらの値は、JSP 上では、{@SYS.XXXX} で取得可能です。

7. パラメーター

パラメーターは、実行環境での各種設定値を一覧表示します。

これは、{@SYS.XXXX} の実際の取得値（抜粋）や、{@USER.XXXX}、{@GUI.XXXX}、{@SESSION.XXXX}、{@LBL.XXXX}、{@DATE.XXXX AA BB} などの JSP 開発時に使用できる変数や、ServletConfig、ServletContext、ServletRequest、PageContext、HttpSession、Java System Properties などの値も表示しています。

8. アクセスストップ

画面上からは操作できませんが、COMMAND=AccessStop で、一般ユーザーのアクセスを停止させることが可能です。

直接、以下の URL を打ち込みます。（ログインユーザーは、SYSTEM か、admin に限定）

http://localhost:8824/gf/JSP/admin?COMMAND=AccessStop

この処理を利用するには、opengion/uap/Webapps/gf/WEB-INF/Web.xml の AccessStopFilter のコメントを外してください。（<filter>の箇所と、<filter-mapping>の2か所あります）

これは、Servlet の Filter を有効にすることで、各ユーザーからのアクセスを停止させます。通常の設定では、時間指定で自動的にアクセスストップが実施されますが、COMMAND=AccessStop をセットすることで、リアルタイムなシステム停止が可能です。

なお、ログインユーザーが、SYSTEM または、admin の場合は、このフィルターで止まりません。つまり、他のユーザーのアクセスを停止しながら、これらのユーザーはシステムにアクセスできます。逆に、停止したシステムを再度、アクセス許可を与える場合は、再び、COMMAND=AccessStop で接続すれば、トグルになっているので、許可されます。当然、許可できるのは、アクセスが止められていない、SYSTEM か admin のユーザーだけという事になります。

索引

A

AccessStopFilter	89
ALL	22, 30, 60, 73
ARG_ARRAY	68, 69, 70, 71, 75, 97
AUTOAREA	55, 58

B

Base64 形式	103
BASIC 認証	12, 103
BulkQuery	84
business	16

C

Calc	79, 82
CalcDef	79
CalcDefAno	79
calendar	19
ChainProcess	83, 84, 85
CHBOX	55, 58
CODE39	55
column	19, 76, 96
COLUMN	52, 55, 58, 70
common	14, 17, 101, 107
CRYPT	55, 58, 60
CSS	11
CSV	79
CSV2	79
CustomData	77

D

daemon	16, 19
Data	80
Data2	80
DATE	55, 60, 73, 74, 112
db 16, 17, 27, 72, 73, 81, 96, 97	
DBCountFilter	84
DBLABEL	55
DBMENU	56, 58, 59
DBMerge	84
DBParam	84

DBReader	84
DBTableModel .. 11, 16, 17, 19, 45, 46, 72, 74, 76, 77, 79, 81, 83, 86, 90, 91, 95, 96, 97	
DBWriter	84
DD	59, 60, 70, 91
DECIMAL	56, 58
Default	80, 82
develop	17, 19
DIGEST 認証	103

E

ENTCLM	59
ERR_MSG_ARRAY	68, 69, 70, 71
Excel	80, 82

F

FileCopy	85
FileFilter	89
FileFtp	85
FileSearch	85
filter	17, 112
f ilter	89
FILTER	56
FirstProcess	83, 84, 85
Fixed	80, 82
FORM	2, 56, 91, 92
FORM レンダラー	91
fukurou	14, 16, 19

G

Grep	85
GrepChange	85
GrepChangeExcel	85
GUI リソース	65, 106
GZIPFilter	89

H

HIDDEN	59
HM	56, 61
HMS	56, 61
html	17

HTML	10, 11, 17, 21, 46, 52, 56, 59, 61, 77, 90, 93
HTMLAjaxTreeTable	77
HTMLCalendar	78
HTMLCrossTable	78
HTMLCustomTable	78
HTMLCustomTreeBOM	78
HTMLDynamic	78
HTMLEntry	78
HTMLFormatTable	78
HTMLFormatTextField	78
HTMLGanttTable	78
HTMLRotationTable	78
HTMLSeqCImTable	78
HTMLSimpleList	78
HTMLTable	78
HTMLTextField	78
HTMLTimeTable	78
HTMLTreeBOM	78
HybsProcess	83, 84

I

io	14, 16, 17, 19, 24, 27, 46, 72, 76, 77, 80, 99, 100, 101, 103, 104, 108, 109, 111, 112
----	--

J

JDBC	24, 27, 72, 73, 74, 75, 97, 99
JDBCCallable	74
JDBCErrorMsg	72, 74, 97
JDBCKeyEntry	75
JDBCPL/SQL	73, 75
JDBCPrepared	73, 75
JDBCUpdate	75
JExcel	82
JSP	29

K

K	61
KANA	56
KCL	61
KX	61

L

LABEL	56, 57, 96
LDAPReader	85
Logger	85

M

mail	16, 17
MailReceive	87
mainProcess	83, 84
maxRowCount	46, 72, 81, 90, 91
MD	56, 57, 60, 61, 62, 96
MENU	54, 56, 59
model	16
MONEY	56
MULTIQUERY	56
MVC	11, 12

N

NBSP	56
NUMBER	50, 51, 56, 59, 61, 68, 69, 70, 71
NVAR	61

O

OASNM	61
openGion	9
ORACLE XDK 形式	80

P

Parameter	109
ParamProcess	83, 84
PASSWD	27, 56, 59, 100
PL/SQL	29
PL/SQLUpdate	72, 73
PN	16, 50, 57, 59, 61
PN2	57
PRE	57
PROCEDURE SET_ERRMSG	69
process	16, 83, 84
Properties	80

Q

query	14, 19, 45, 46, 68, 72, 73, 74, 75, 76, 79, 90, 91, 93, 97
QUERY	42, 57, 59, 66, 93
query タグ	72
QUERY レンダラー	93
queryType	74
query タグ	91
QUERY 画面	42

R

R 61	
RADIO.....	57, 58, 59
readTable タグ	81
remote.....	17
report.....	14, 17
Report.....	87
Report2.....	87
resource	17, 100, 104
ResultSetエラー! ブックマークが正しくありません。 11	
RESULT 画面.....	42

S

S921, 61	
scope.....	46, 72, 76
security.....	16
servlet.....	17, 101, 108
shutdown.bat	40
SSL.....	103
startup.bat	40
StringUtil.....	85

T

T 80	
table.....	19, 46, 72, 73, 74, 76, 77, 81
TableDiff	85
tableFilter	83, 89
TableFilter.....	85
TableReader	85
tableUpdate	72, 73
TableWriter.....	85
taglet	16
taglib.....	18, 109, 111
TEXT	52, 57, 59, 93
TEXTAREA.....	57, 59
TMSTMP.....	57
transfer	16
Transfer	87
Transfer_CB01	87

U

UPLOAD	14, 60
URLCheckFilter	89
URLConnect.....	87
URLHashFilter	89

UserInfo オブジェクト	64
util.....	16

V

view.....	19, 76, 77, 90, 91, 93
view タグ	76, 83
viewFormType	77
view タグ	91

W

workdelete.bat	40
WRITABLE.....	57, 60
writeTable タグ	79

X

X 61	
X961	
XH.....	11, 61
XK.....	61, 93
XL21, 61	
xml.....	16
XML	80
XSLT.....	85
XU.....	21, 62
XXXX	57, 70, 109, 111, 112

Y

YM	56, 57, 60, 62, 96
----------	--------------------

あ

アクセスストップ	109
アクセス制限.....	10
アクセスモード	66, 106

か

開発/単体テスト	22
カスタムタグ.....	33
カラム	49
カラムオブジェクト .37, 49, 50, 51, 53, 54, 55, 67, 90, 91	
カラム名.....	21, 37, 50, 51, 55, 71, 73, 74, 76
簡易ワークフロー	10

き	
基本設計	21, 22, 31
共通オブジェクト定義	68

く	
グループ	106

こ	
コードカラム	51, 53
コードリソース	54
国際化対応	10
個人ポータルサイト	10
コネクションプーリング	12
コンテキスト画面	42

し	
システムリソース	109
状況表示	109
使用桁数	51
詳細設計	21, 22

す	
ストアドプロシージャ	11, 30

せ	
セキュアセッション	103
セキュリティ制限	100

そ	
総合テスト	22

た	
ターゲット	42, 66
ダイレクト・パス・インポート	95
ダイレクト・パス・エクスポート	95
タグライブラリ	18, 25, 30
タグリブ情報	109

て	
ディレクトリー一覧	101
ディレクトリ構成	14

データの属性	51
データのタイプ	52, 60
データのデフォルト値	51, 52
データベースコネクション	109
データロールズ	64

の	
納入仕様書	21, 31

は	
パスワード	10, 27, 28, 55, 56, 58, 59, 61, 64, 65, 103
汎用オブジェクト	29, 33, 36, 50

ひ	
表形式オブジェクト	35, 36, 37
表示桁数	51, 52
表示種別	51, 52
表示用レンダラー	52, 54, 55, 91

ふ	
フル削除	109
フォーム認証	12
プラグイン情報	109
プロジェクト	106

へ	
編集種別	51, 52
編集用エディター	52, 54, 58

ほ	
本番導入	22

め	
メッセージリソース	63
メニュー画面	42

も	
文字種別	21, 51, 52

ゆ

ユーザー数の制限	12
ユーザー認証	99
ユーザーリソース	63, 64, 65, 99, 106

ら

ラベルカラム	51, 53
ラベルリソース	54, 55, 63

り

リソースファイル	11, 21, 31, 49, 63
----------------	--------------------

ろ

ロール	106
ロールズ	64, 65, 66
ログイン有効日	65
ロケイユザ-	109
ログレベル	12