

FINALUNA API

【詳細設計書】

第 3.2.0 版

2011 年 12 月 31 日

(株)NTT データ

改編履歴

改編日付	作成者/更新者	版数	更新内容
2006年3月31日	NTT データ	0.5	新規作成
2006年6月30日	NTT データ	0.7	StringType 型の StringBuffer 対応 Context にメソッドを追加 CodeType の実装クラス名を変更
2006年10月10日	NTT データ	1.0	Java 標準化対応 DTO に保持する型を Finaluna 型のみではなく、Java 標準 API も保持出来るように変更
2006年11月1日	NTT データ	1.1	出口処理前後処理クラス分割対応 2.5 業務処理及び 4.1.7 業務処理の記述を修正。
2006年11月30日	NTT データ	1.1.0	DateAndTimeType 型の追加による修正。
2007年7月25日	NTT データ	1.2.2	UpdatedDAO のメソッド追加。 public IntegerType update (QueryCode id)
2008年10月17日	NTT データ	2.0.0	全体的に最新版の API に修正。
2008年10月22日	NTT データ	2.0.0	全体的に最新版の API に修正。
2010年2月24日	NTT データ	3.0.0	RS のインタフェースに合わせた修正を実施。
2011年3月22日	NTT データ	3.1.0	RS のインタフェースに合わせた修正を実施。
2011年12月31日	NTT データ	3.2.0	Terasoluna ライブラリを最新化。

目次

0. はじめに.....	1
0.1. 本書について.....	1
1. API 概要.....	2
2. 機能概要.....	3
2.1. FINALUNA 型 (jp.finaluna.api.type).....	3
2.1.1. 文字列系.....	5
2.1.2. コード系.....	5
2.1.3. 数値系.....	7
2.1.4. 日付系.....	8
2.1.5. 配列系.....	9
2.1.6. その他.....	10
2.1.7. ファクトリ.....	12
2.1.8. FINALUNA 型ユーティリティ (jp.finaluna.api.type.util).....	13
2.1.9. FINALUNA 型 値チェック (jp.finaluna.api.type.util.validator).....	18
2.2. 電文 I/O (jp.finaluna.api.dto).....	20
2.2.1. DTO (DTO).....	21
2.2.2. リクエスト DTO (RequestDTO).....	21
2.2.3. レスポンス DTO (ResponseDTO).....	22
2.2.4. パラメータ DTO (ParameterDTO).....	22
2.2.5. 処理結果ステータス保持インタフェース (ResponseStatusHolder).....	23

2.2.6. メッセージ保持インタフェース (MessageHolder)	23
2.2.7. 応答ステータスコード (ResponseStatusCode)	23
2.3. メッセージ (jp.finaluna.api.dto.message)	25
2.3.1. 業務メッセージ (Message)	25
2.3.2. 複数業務メッセージ (Messages)	26
2.3.3. 使用例	27
2.4. DB、ファイル I/O 、共有変数 (jp.finaluna.api.dao)	28
2.4.1. データアクセスオブジェクト (DAO)	30
2.4.2. 照会用 DAO (QueryDAO)	30
2.4.3. 更新用 DAO (UpdateDAO)	30
2.4.4. ページ照会用 DAO (PageQueryDAO)	30
2.4.5. カーソル照会用 DAO (CursorDAO)	31
2.4.6. ストアドプロシージャ実行用 DAO (StoredProcedureDAO)	31
2.4.7. エンティティ照会用 DAO (EntityQueryDAO)	31
2.4.8. エンティティ更新用 DAO (EntityUpdateDAO)	31
2.4.9. SQL 実行前チェック (SqlPreCheck)	32
2.4.10. クエリコード (QueryCode)	33
2.4.11. FINALUNA 論理排他例外 (FinalunaOptimisticLockFailureException)	33
2.5. 業務処理 (jp.finaluna.api.blogic)	34
2.5.1. 業務ロジック (Logic)	35
2.5.2. コンテキスト (Context)	36
2.5.3. 業務コンテキスト (ProcessContext)	37
2.5.4. 起動時実行ロジック (StartupLogic)	37
2.5.5. 業務停止時実行ロジック (StopLogic)	37
2.5.6. 業務ロジックコード (BLogicCode)	38
2.5.7. DAO コード (DAOCode)	38

2.5.8. DTO コード (DTOCode)	38
2.5.9. 使用例.....	39
2.6. 出口処理 (jp.finaluna.api.exit).....	40
2.6.1. ロジック後出口 (LogicPostExit).....	40
2.6.2. ロジック前出口 (LogicPreExit).....	40
2.7. 例外処理 (jp.finaluna.api.exception).....	41
2.7.1. Finaluna 業務例外 (FinalunaApplException).....	41
2.7.2. Finaluna システム例外 (FinalunaSysException).....	41
2.7.3. Unsupport 例外 (FinalunaUnsupportedOperationException).....	42
2.7.4. エラーメッセージホルダー (ErrorMessageHolder).....	42
2.8. ロガー (jp.finaluna.api.log).....	43
2.8.1. ロガーファクトリ (LogFactory).....	43
2.8.2. ロガー生成ファクトリインタフェース (LogFactoryDelegate).....	44
2.8.3. ロガーインタフェース (Logger).....	44
3. FINALUNA API クラス構成.....	45
3.1. FINALUNA API クラス構成.....	45
4. 補足.....	46
4.1. JavaDoc.....	46
4.1.1. FINALUNA 型 (jp.finaluna.api.type).....	47
4.1.2. FINALUNA 型ユーティリティ (jp.finaluna.api.type.util).....	61
4.1.3. FINALUNA 型入力チェック (jp.finaluna.api.type.util.validator).....	68
4.1.4. 電文 I/O (jp.finaluna.api.dto).....	71
4.1.5. メッセージ (jp.finaluna.api.dto.message).....	73
4.1.6. DB、ファイル I/O 、共有変数(jp.finaluna.api.dao).....	74

4.1.7. 業務処理 (jp.finaluna.api.blogic)	77
4.1.8. 出口処理 (jp.finaluna.api.exit)	80
4.1.9. 例外処理 (jp.finaluna.api.exception)	81
4.1.10. ログ一 (jp.finaluna.api.log)	83
4.2. 用語集	85

0. はじめに

0.1. 本書について

本書「詳細設計書 (FINALUNA API)」は、本システムが提供するAPIの機能仕様について、基本的な実現内容を記述したものである。

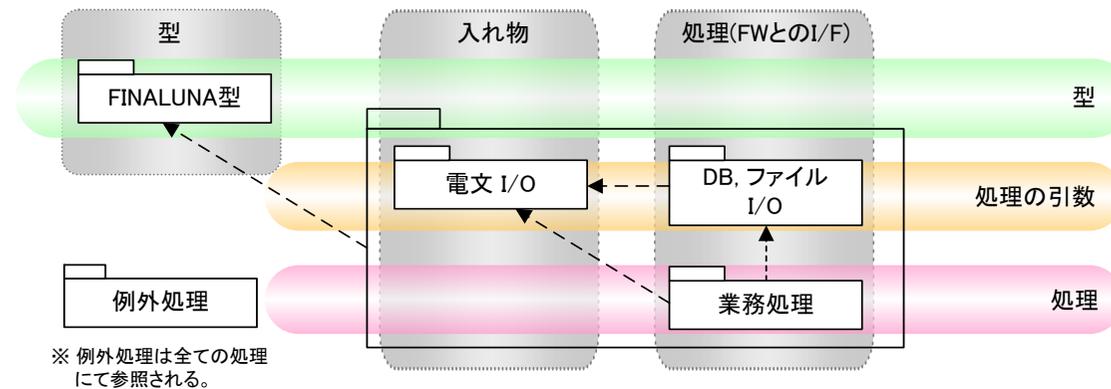
1. API 概要

FINALUNA APIは、業務ロジックを開発する際に使用するAPIである。

FINALUNA APIは以下の特徴を持つ。

- FINALUNA API は、大部分が Java のインタフェースで構成されており、その実装はフレームワークが提供する為 API は恒久的に不変である。そのため、作成済みの業務ロジックに改修をかけなければならないのは、原則として仕様変更が発生した場合のみとなる。そのため、保守性の高い業務ロジックを開発することが可能である。
- FINALUNA API は、Java の標準API の型を内包した独自の型を提供する。この独自の型は、業務ロジック開発の観点で必要となる型のみを提供するため、Java の細かい仕様を知ることなく、スムーズに型の適用を行うことが可能である。
- 業務処理への入出力インタフェースが規定されているため、業務ロジックでは入力リソースと出力先を意識することなく、開発を行うことが可能である。

FINALUNA APIの構成概要図は以下のとおり。



■ 図 1. 1 FINALUNA API 構成概要図

2. 機能概要

2.1. FINALUNA型 (jp.finaluna.api.type)

FINALUNA型とは、Javaの標準APIの型を内包した独自の型である。

業務ロジック内では、FINALUNA型を使用することで、より標準化を図った業務ロジックを記述することができる。

FINALUNA型は、ベースとなるFINALUNA型、もしくは既存のFINALUNA型を拡張することで、プロジェクト毎に追加することが可能である。

以下にFINALUNA APIが提供するFINALUNA型の一覧を示す。

表 2.1 FINALUNA型一覧

分類	クラス名		説明
文字列系	StringType	文字列型	可変文字列を保持する型
	StringCode	文字列型	固定文字列を保持する型
	CharSequenceType	文字シーケンス型	文字シーケンスを保持する型
コード系	CodeType	コード型	コード値、区分を表す型
	KeyCodeType	キーコード型	更新可能なコード型
数値系	IntegerType	整数型	整数を保持する型
	DecimalType	小数型	小数を保持する型
日付系	DateType	日付型	年月日を保持する型
	TimeType	時刻型	時分秒を保持する型
	TimestampType	日時型(ナノ秒)	年月日時分秒 ナノ秒を保持する型
	DateAndTimeType	日時型	年月日時分秒を保持する型
配列系	ListType	リスト型	リストを保持する型
	MapType	マップ型	マップを保持する型
	SetType	セット型	セットを保持する型
	CollectionType	コレクション型	コレクションを保持する型
その他	BaseType	FINALUNA基底型	全ての型の基となる型
	CursorSet	カーソルセットインタフェース	データベースから検索結果を一件ずつ取得するためのインタフェース
	FinalunaFormatException	フォーマット例外	NumberFormatUtil、DateFormatUtil で、フォーマット、解析失敗時にスローされる
	Copiable	コピー	インスタンスのコピーを戻すインタフェース
	Rcsid	RCSIDアノテーション	リビジョン番号、日付情報を保持するアノテーション
ファクトリ	ConstantTypeFactory	ファクトリインタフェース	Finaluna型を生成するファクトリインタフェース
	ConstantFactory	ファクトリクラス	Finaluna型を生成するファクトリクラス
	CollectionFactoryDelegate	ファクトリインタフェース	Finaluna型を生成するファクトリインタフェース
	CollectionFactory	ファクトリクラス	Finaluna型を生成するファクトリクラス

2.1.1. 文字列系

文字列型を保持するクラス。

分類	クラス名		説明
文字列系	StringType	文字列型	可変文字列を保持する型
	StringCode	文字列型	固定文字列を保持する型
	CharSequenceType	文字シーケンス型	文字シーケンスを保持する型

2.1.1.1. 文字列型 (StringType)

文字列を保持する型。

可変文字列を保持する。デフォルトの最大文字数は50*1024*1024文字。最大文字数を越えた場合、FinalunaSysExceptionをthrowする。

2.1.1.2. 文字列型 (StringCode)

文字列を保持する型。

固定文字列を保持する。デフォルトの最大文字数は50*1024*1024文字。最大文字数を越えた場合、FinalunaSysExceptionをthrowする。

2.1.1.3. 文字シーケンス型 (CharSequenceType)

文字シーケンスを保持する。

2.1.2. コード系

コード系はコード値、区分など定数/メソッドパラメータを表す型。

分類	クラス名		説明
コード系	CodeType	コード型	コード値を表す型
	KeyCodeType	キーコード型	更新可能なコード型

2.1.2.1. コード型 (CodeType)

コード値を表す型。
処理における識別情報などを保持する。

2.1.2.2. キーコード型 (KeyCodeType)

更新可能なコード型。
コード型と同様の位置づけの型。値の更新参照を許容している。

2.1.3. 数値系

数値を保持する型。この数値の比較演算のメソッドを持つ。

演算結果が小数になる場合、小数部を破棄出来る演算を提供している。ユーザが明示的にスケールと丸めモード(RoundCode)を指定して小数部を破棄することを可能とする。

分類	クラス名		説明
数値系	IntegerType	整数型	整数を保持する型
	DecimalType	小数型	小数を保持する型

2.1.3.1. 数値型 (IntegerType)

整数を保持する型。

最大値 $((2-2^{-52}) \cdot 2^{1023})$ ・最小値 $(-(2-2^{-52}) \cdot 2^{1023})$ の境界値を超える値を代入した場合は、FinalunaSysExceptionをthrowする。

2.1.3.2. 小数型 (DecimalType)

小数を保持する型。

最大値 $((2-2^{-52}) \cdot 2^{1023})$ ・最小値 $(-(2-2^{-52}) \cdot 2^{1023})$ の境界値を超える値を代入した場合は、FinalunaSysExceptionをthrowする。

2.1.4. 日付系

年月日時分秒の情報を保持する型。

分類	クラス名		説明
日付系	DateType	日付型	年月日を保持する型
	TimeType	時刻型	時分秒を保持する型
	TimestampType	日時型(ナノ秒)	年月日時分秒 ナノ秒を保持する型
	DateAndTimeType	日時型	年月日時分秒を保持する型

2.1.4.1. 日付型 (DateType)

年月日を保持する型。

日付の比較、編集メソッドを持つ。内部に保持する情報は年月日。この型が保持する情報は年月日であり、それより単位が小さい時間フィールドの情報は初期化される。つまり、このクラスの利用者は時間フィールドを気にすることなく、純粋な日付(年月日)のみを比較、または計算した結果を取得することができる。この型の文字列表現はyyyy-MM-dd形式です。

2.1.4.2. 時刻型 (TimeType)

時分秒を保持する型。

時刻の比較、編集メソッドを持つ。内部に保持する情報は時分秒。年月日の値は、Java.util.Calendarクラスの基準時間を保持する。この型が保持する情報は時刻フィールドであり、それより単位が大きい日付フィールドの情報は初期化される。つまり、このクラスの利用者は日付フィールドを気にすることなく、純粋な時刻(時分秒)のみを比較、または計算した結果を取得することができる。この型の文字列表現はHH:mm:ssです。

2.1.4.3. 日時型(ナノ秒) (TimestampType)

年月日時分秒 ナノ秒を保持する型。

日時の比較、編集メソッドを持つ。内部に保持する情報は年月日時分秒ナノ秒。この型の文字列表現はyyyy-MM-dd HH:mm:ss.fffffffです。

2.1.4.4. 日時型 (DateAndTimeType)

年月日時分秒を保持する型。

日時の比較、編集メソッドを持つ。内部に保持する情報は年月日時分秒。この型の文字列表現はyyyy-MM-dd HH:mm:ss形式です。

2.1.5. 配列系

配列の情報を保持する型。

分類	型名		説明
配列系	ListType	リスト型	リストを保持する型
	MapType	マップ型	マップを保持する型
	SetType	セット型	セットを保持する型
	CollectionType	コレクション型	コレクションを保持する型

2.1.5.1. リスト型 (ListType)

リストを保持する。

2.1.5.2. マップ型 (MapType)

マップを保持する。

2.1.5.3. セット型 (SetType)

セットを保持する。

2.1.5.4. コレクション型 (CollectionType)

コレクションを保持する。

2.1.6. その他

上記FINALUNA型に属さないタイプの型。

分類	型名	説明
その他	BaseType	FINALUNA基底型
	CursorSet	カーソルセットインタフェース
	FinalunaFormatException	フォーマット例外
	Copiable	コピー
	Rcsid	RCSIDアノテーション

2.1.6.1. FINALUNA基底型 (BaseType)

全ての型の基となる型。

FINALUNA型の基となる。

2.1.6.2. カーソルセットインタフェース (CursorSet)

インスタンスのコピーを戻すインタフェース。

インスタンスのコピーを戻す。

2.1.6.3. フォーマット例外 (FinalunaFormatException)

フォーマット・解析処理例外。

NumberFormatUtil、DateFormatUtilで、フォーマット、解析失敗時にスローされる。

2.1.6.4. コピー (Copiable)

インスタンスのコピーを戻す。

2.1.6.5. RCSIDアノテーション (Rcsid)

リビジョン番号、日付情報を保持するアノテーション。

2.1.7. ファクトリ

上記FINALUNA型を生成するためのファクトリクラス。

分類	型名		説明
ファクトリ	ConstantFactoryDelegate	ファクトリインタフェース	Finaluna型を生成するファクトリインタフェース
	ConstantFactory	ファクトリクラス	Finaluna型を生成するファクトリクラス
	CollectionFactoryDelegate	ファクトリインタフェース	Finaluna型を生成するファクトリインタフェース
	CollectionFactory	ファクトリクラス	Finaluna型を生成するファクトリクラス

2.1.7.1. ファクトリインタフェース (ConstantFactoryDelegate)

Finaluna型を生成する為のファクトリインタフェース。

Finaluna型を生成するファクトリ実装クラスは、このConstantFactoryDelegateインタフェースを実装する。

2.1.7.2. ファクトリクラス (ConstantFactory)

Finaluna型を生成する為のファクトリ。

業務ロジッククラスのFinaluna型フィールドに格納するインスタンス生成は、このConstantFactoryクラスを用いて生成する。

2.1.7.3. ファクトリインタフェース (CollectionFactoryDelegate)

Finaluna型を生成する為のファクトリ。

業務ロジッククラスのFinaluna型フィールドに格納するインスタンス生成は、このCollectionFactoryDelegateクラスを用いて生成する。
(ListType, SetType, MapTypeに用いる。)

2.1.7.4. ファクトリクラス (CollectionFactory)

Finaluna型を生成する為のファクトリ。

業務ロジッククラスのFinaluna型フィールドに格納するインスタンス生成は、このCollectionFactoryクラスを用いて生成する。
(ListType, SetType, MapTypeに用いる。)

2.1.8. FINALUNA型ユーティリティ (jp.finaluna.api.type.util)

FINALUNA型を操作する上でのユーティリティ機能。

分類	クラス名		説明
ユーティリティ	DateUtil	日付操作ユーティリティクラス	日付系のユーティリティクラス
	DateFormatUtil	日付操作ユーティリティクラス	日付文字列の出力、解析を行うユーティリティクラス
	TimestampUtil	タイムスタンプユーティリティクラス	現在タイムスタンプ取得ユーティリティクラス
	MathUtil	数値操作ユーティリティクラス	数値系のユーティリティクラス
	NumberFormatUtil	数値操作ユーティリティクラス	数値を文字列に変換し整形するユーティリティクラス
	NumerationSystemUtil	数値操作ユーティリティクラス	命数法を用いた数値表現の文字列を、命数法を用いない数値表現の文字列に置き換えるユーティリティクラス
	MessageUtil	メッセージ操作ユーティリティクラス	メッセージ系のユーティリティクラス
	OutPrinterUtil	帳票出力ユーティリティクラス	帳票を印刷するユーティリティクラス
	PropertyUtil	プロパティユーティリティクラス	プロパティの操作を行うユーティリティ
	ConvertUtil	型変換ユーティリティクラス	java⇔FINALUNA型相互変換ユーティリティクラス。
	ConvertUtilDelegate	型変換ユーティリティインタフェース	java⇔FINALUNA型相互変換ユーティリティインタフェース
	FinalunaTypeConvertUtil	型変換ユーティリティクラス	FINALUNA型間の相互変換を行うための型変換ユーティリティクラス
	FinalunaTypeConvertUtilDelegate	型変換ユーティリティインタフェース	FINALUNA型間の相互変換を行うための型変換ユーティリティインタフェース
	BlankStringUtil	空文字列ユーティリティクラス	空文字列、空白文字列を判定するためのユーティリティクラス
BlankStringUtilDelegate	空文字列ユーティリティインタフェース	空文字列、空白文字列を判定するためのユーティリティインタフェース	

2.1.8.1. 日付操作ユーティリティ (DateUtil)

日付系のユーティリティクラス。

2.1.8.2. 日付操作ユーティリティ (DateFormatUtil)

日付文字列の出力、解析を行うユーティリティクラス。
パターンを指定して、日付文字列の出力、解析を行う。
パターンは以下の文字列を組み合わせて指定する。

G	元号	和暦用ロケールが設定されている場合、G の数で和暦情報プロパティファイルに 定義されているどの和暦表記を使用するかを決定する。 (例)和暦情報プロパティが「19890108=h,H,平成」の場合、G→「h」、GG→「H」、GGG→「平成」
y	年	(例:2009 年) yyyy→「2009」、yy→「09」、(和暦用ロケールで)yy→「21」
M	月	(例:3 時) MM→「03」
d	日	(例:3 日) dd→「03」
H	時(24h表記)	(例:午後1時) HH→「13」
m	分	(例:15 分) mm→「15」
s	秒	(例:20 秒) ss→「20」
S	ミリ秒	(例:123 ミリ秒) SSS→「123」ミリ秒未満は切り捨て
N	ナノ秒	(例:123456789 ナノ秒) NNNNNNNNN→「123456789」※ミリ秒を含む
Z	タイムゾーン	(例:日本時間) Z→「+0900」

2.1.8.3. タイムスタンプユーティリティ (TimestampUtil)

現在タイムスタンプ取得ユーティリティクラス。
現在タイムスタンプを取得する。 TimestampAdjustUtilで、取得するタイムスタンプを補正することが出来る (テスト用)。

2.1.8.4. 数値操作ユーティリティ (MathUtil)

計算処理を行うユーティリティクラス。
区分値として以下の区分を保持する。

- 丸めモード区分
数値の丸めモードを表す。
- 範囲区分
境界値の範囲を表す。

- 境界値区分

境界値を含むか否かを表す。

2.1.8.5. 数値操作ユーティリティ (NumberFormatUtil)

数値を文字列に変換し整形する。 数字を表す整形された文字列を解釈し数値に変換する。
パターンを指定して、数字文字列の出力、解析を行う。
パターンは以下の文字列を組み合わせて指定する。

0	指定桁数に満たない場合はゼロ埋め (例:123) 00000→「00123」
#	ゼロだと表示されない (例:123) #,###→「123」
.	数値桁区切り子 (例:12.3) ##0.00→「12.30」
'	グループ区切り子 (例:123456) #,###→「123,456」
-	マイナス記号 (例:-123456) #,###;- #,###→「-123,456」
%	接尾辞(100 倍してパーセントを表す) (例:1.234) ###.###%→「123.4%」
;	サブパターン境界(正と負のサブパターンを区切る) (例:123456) #,###;#CR→「123,456」 (例:-123456) #,###;#CR→「123,456CR」

2.1.8.6. 数値操作ユーティリティ (NumerationSystemUtil)

命数法を用いた数値表現の文字列を、命数法を用いない数値表現の文字列に置き換える。

指定文字列

t/T	Thousand 10 の 3 乗	1t→1000 2T→2000
m/M	Million 10 の 6 乗	3m→3000000 4M→4000000
b/B	Billion 10 の 9 乗	5b→5000000000 6B→6000000000

変換例

	対象文字列	replace	toPlainNumberStringCode
通常	1,234t	1,234000	1234000
指定文字の重複	1tb	1000000000000	1000000000000
指定文字のみ	t	000	0
少数表示	1.1t	1.1000	1.1000
0のみ	0t	0000	0
0のみ(少数表示)	0.t	0.000	0.000
マイナス	-1t	-1000	-1000
末尾以外の指定文字	10m0	100000000	100000000
少数表示	1m.1	1000000.1	1000000.1
指定文字以外が入ってきた場合	Thoge	000hoge	FinalunaApplException
全角のTMBが入ってきた場合	TMB	TMB	FinalunaApplException
全角の数字が入ってきた場合	123	123	123
通常	1,234t	1,234000	1234000
指定文字の重複	1tb	1000000000000	1000000000000

2.1.8.7. メッセージ操作ユーティリティ (MessageUtil)

メッセージ系のユーティリティクラス。
メッセージ編集：指定されたメッセージキーを元にメッセージ文字列を作成する。

2.1.8.8. 帳票出力ユーティリティ (OutPrinterUtil)

帳票を印刷するユーティリティクラス。
DTOを帳票データとして扱い帳票のプリンター出力を行う。

2.1.8.9. プロパティユーティリティ (PropertyUtil)

プロパティの操作を行うユーティリティクラス。
業務ロジックで使用するプロパティに関する処理を提供する。プロパティ情報の取得：指定したキーを基に、プロパティファイルから情報を取得する。。

2.1.8.10. 型変換ユーティリティ (ConvertUtil)

java⇔FINALUNA型相互変換ユーティリティクラス。
java⇔FINALUNA型相互変換のユーティリティ処理を実装する。

2.1.8.11. 型変換ユーティリティインタフェース (ConvertUtilDelegate)

java⇔FINALUNA型相互変換ユーティリティインタフェース。

java⇔FINALUNA型相互変換のユーティリティ実装クラスはこのConvertUtilDelegateインタフェースを実装する。

2.1.8.12. 型変換ユーティリティクラス (FinalunaTypeConvertUtil)

FINALUNA型⇔FINALUNA型相互変換ユーティリティクラス。

FINALUNA型⇔FINALUNA型相互変換のユーティリティ処理を実装する。

2.1.8.13. 型変換ユーティリティインタフェース (FinalunaTypeConvertUtilDelegate)

FINALUNA型⇔FINALUNA型相互変換ユーティリティインタフェース。

FINALUNA型⇔FINALUNA型相互変換のユーティリティ実装クラスはこのFinalunaTypeConvertUtilDelegateインタフェースを実装する。

2.1.8.14. 空文字列ユーティリティクラス (BlankStringUtil)

空文字列、空白文字列を判定するためのユーティリティクラス。

2.1.8.15. 空文字列ユーティリティインタフェース (BlankStringUtilDelegate)

空文字列、空白文字列を判定するためのユーティリティインタフェース。

2.1.9. FINALUNA型 値チェック (jp.finaluna.api.type.util.validator)

FINALUNA型を対象に値のチェックを行うユーティリティ機能。
この機能は、業務ロジック処理実行前に値をチェックする為に使用される。

分類	クラス名		説明
バリデータ	DateValidator	DateType型入力チェック	DateType型の入力チェックを行う機能
	NumericValidator	IntegerType型及びDecimalType型入力チェック	IntegerType型及びDecimalType型の入力チェックを行う機能
	StringValidator	StringType型入力チェック	StringType型及びStringCode型の入力チェックを行う機能
	TimestampValidator	TimestampType型入力チェック	TimestampType型の入力チェックを行う機能
	TimeValidator	TimeType型入力チェック	TimeType型の入力チェックを行う機能
	DateAndTimeValidator	DateAndTimeType型入力チェック	DateAndTimeType型の入力チェックを行う機能

2.1.9.1. Date型入力チェック (DateValidator)

DateType型の入力チェックを行う機能。
日付の前後関係のチェック、日付の範囲チェックなどを行う。

2.1.9.2. IntegerType型及びDecimalType型入力チェック (NumericValidator)

IntegerType型及びDecimalType型の入力チェックを行う機能。
値の桁数チェック、数値の範囲チェックなどを行う。

2.1.9.3. StringType型入力チェック (StringValidator)

StringType型及びStringCode型の入力チェックを行う機能。
値のNULLチェック、正規表現によるチェックなどを行う。

2.1.9.4. Timestamp型入力チェック (TimestampValidator)

TimestampType型の入力チェックを行う機能。
日時の前後関係のチェック、日時の範囲チェックなどを行う。

2.1.9.5. Time型入力チェック (TimeValidator)

TimeType型の入力チェックを行う機能。
時刻の前後関係のチェック、時刻の範囲チェックなどを行う。

2.1.9.6. DateAndTime型入力チェック (DateAndTimeValidator)

DateAndTimeType型の入力チェックを行う機能。
日時の前後関係のチェック、日時の範囲チェックなどを行う。

2.2. 電文I/O (jp.finaluna.api.dto)

業務ロジック - フレームワーク間のデータ入出力は、全てDTO(データ転送オブジェクト)を介して行う。DTOが保持する値はFINALUNA型/Java標準APIとし、一意な項目名をキーにして保持する。フレームワークが入出力対象となるリソースの入出力機能を担う為、業務ロジックでは、このDTOの入出力対象となるリソースを意識せず実装出来る。

DTOが使用される箇所を以下に示す。

- 業務ロジックの入出力
- DAOの入出力

分類	クラス名		説明
電文I/O	DTO	DTOマーカーインタフェース	業務ロジックの入出力値を保持するデータ転送オブジェクトのマーカーインタフェース
	RequestDTO	リクエストDTO	入力情報を格納するデータ転送オブジェクト
	ResponseDTO	レスポンスDTO	出力情報を格納するデータ転送オブジェクト
	ParameterDTO	パラメータDTO	入出力情報を格納するデータ転送オブジェクト
	ResponseStatusHolder	処理結果ステータス保持インタフェース	処理結果ステータスを保持するインタフェース
	MessageHolder	メッセージ保持インタフェース	メッセージを保持するインタフェース
コード	ResponseStatusCode	応答ステータスコード	応答ステータスを一意に識別するコード

2.2.1. DTO (DTO)

【マーカーインタフェース】

業務ロジックの入出力値を保持するデータ転送オブジェクトのマーカーインタフェース。 リクエストDTO、パラメータDTO、レスポンスDTOはこのインタフェースを継承している。

業務ロジック、DAO(データアクセスオブジェクト)間の入出力は、DTO(データ転送オブジェクト)を介して行う。

DTOが保持する値はすべてFINALUNA型とし、フレームワークに依存しないオブジェクトとする。

BaseTypeインタフェースを継承したDTOインタフェースをマーカーインタフェースとして、下位インタフェースはこれを継承する。

DTOの他にRequestDTO、ParameterDTO、ResponseDTOなどのインタフェースも提供するが、

DTOならびに、必要な外部インタフェース(Copiable、MessageHolder等)を直接継承して業務DTOを作成してもよい。

2.2.2. リクエストDTO (RequestDTO)

入力情報を格納するデータ転送オブジェクト。

フレームワークから業務ロジックへの入力情報を格納する。

リクエストDTOは業務ロジックからの変更を許さない為、読取専用とする。 よって、このDTOに値を設定出来るのはフレームワークのみとする。

業務ロジックからは、項目名をキーにして値(FINALUNA型)を取得することが出来る。

Note:

DTOはデータ取得メソッド(BaseType get(String name))を持っているが、取得した結果を抽象的なFinaluna型クラスを返す。

業務ロジック内でFinaluna型へのキャストをしなければならなくなる為、RequestDTOを継承し、項目毎のgetメソッドを持つクラスを作成することを推奨する。

2.2.3. レスポンスDTO (ResponseDTO)

出力情報を格納するデータ転送オブジェクト。

業務ロジックからフレームワークへの出力情報を格納する。

レスポンスDTOは業務ロジックの処理結果をフレームワークに渡す為、業務ロジックからの更新を許す。

レスポンスDTOにはFINALUNA型以外に、業務ロジックの応答ステータス、出力メッセージ、出力エラーメッセージを保持する。

業務ロジックに渡すRequestDTOとResponseDTOの構造が同じ場合、フレームワーク側でRequestDTOに設定されている値を ResponseDTOに 設定して処理することが出来る。

メッセージをResponseDTOに設定することによって、フレームワーク側で メッセージを出力することが出来る。

業務ロジックの処理結果ステータスを設定することによって、バッチの場合フレームワーク側で取引の終了コードを 制御することが出来る。オンラインの場合、論理遷移先を制御することが出来る。設定しない場合はデフォルトが適用される。

このResponseDTOは、フレームワーク側のデータ転送オブジェクト出力処理にて処理される。

2.2.4. パラメータDTO (ParameterDTO)

入出力情報を格納するデータ転送オブジェクト。

業務ロジックからDAOまたは、サブ業務ロジックへの入力情報を格納する。

パラメータDTOは業務ロジックからの更新を許す。 よって、業務ロジックよりこのDTOに値を設定出来る。

業務ロジックからは、項目名をキーにして値 (FINALUNA型) を取得することが出来る。

Note:

DTOはデータ取得メソッド(BaseType get (String name))を持っているが、取得した結果を抽象的なFinaluna型クラスを返す。業務ロジック内でFinaluna型へのキャストをしなければならなくなる為、ParameterDTOを 継承し、項目毎のgetメソッドを持つクラスを作成することを推奨する。

2.2.5. 処理結果ステータス保持インタフェース (ResponseStatusHolder)

処理結果ステータスを保持する。

2.2.6. メッセージ保持インタフェース (MessageHolder)

処理結果ステータスを保持する。

2.2.7. 応答ステータスコード (ResponseStatusCode)

応答ステータスを一意に識別するコード。

業務ロジックの処理結果を保持するFINALUNA型。ステータスコードとステータス重要度の保持・返却を行う。

フレームワークでは処理形式に応じて、処理結果ステータスの扱いが異なる。

【OLTP】

フレームワークが次遷移先画面の識別に用いる。

ステータス” SUCCESS” の場合、成功画面へ、ステータス” WARNING” の場合、失敗画面へと次遷移先を振り分ける。

【バッチ】

終了コード (数値)として扱われる。

フレームワークがステータスの値を判定し、バッチの終了コードを返す。

また、終了コード判定処理を使用することによってステータス毎に 終了コードを設定することも可能としている。

2.2.7.1. 使用例

【業務ロジック実装例】

```
// 値の判定
if (requestDto.getGendarCode().isMan()) {
    // 男性の為、ステータス SUCCESS を返す。
    responseDto.setResponseStatusCode(context.creatResponseStatusCode("SUCCESS"));
} else {
    // 男性以外の為、ステータス WARNING を返す。
    responseDto.setResponseStatusCode(context.creatResponseStatusCode("WARNING"));
}
return;
```

応答ステータスによって、フレームワークの処理の振る舞いを通知する。

(例) 以下のようなフレームワーク制御に用いられる。

【オンライン】

フレームワークが次遷移先画面の識別に用いる。

ステータス” SUCCESS” の場合、成功画面へ、ステータス” WARNING” の場合、失敗画面へと次遷移先を振り分ける。

【バッチ】

終了コード (数値) として扱われる。

フレームワークがステータスの値を判定し、バッチの終了コードを返す。また、終了コード判定処理を使用することによってステータス毎に終了コードを設定することも可能としている。

2.3. メッセージ (jp.finaluna.api.dto.message)

業務ロジックから出力するメッセージ、エラーメッセージを保持する。

このメッセージをレスポンスDTOに設定することによって、フレームワークがメッセージの編集/出力処理を担う。

メッセージはエラーコードと、埋め込み文字を保持している。出力されるメッセージはメッセージコードにひもづいた文字列に埋め込み文字が埋め込まれる。

分類	クラス名		説明
メッセージ	Message	業務メッセージ	メッセージを保持するインタフェース
	Messages	複数業務メッセージ	メッセージを複数保持するインタフェース。

2.3.1. 業務メッセージ (Message)

メッセージを保持するインタフェースは、以下のようなフレームワーク制御に用いられる。

【OLTP】

画面に表示するメッセージとして設定する。

【バッチ】

業務ロジック用ログとして出力される。

【Rich】

業務ロジック用ログとして出力される。

2.3.2. 複数業務メッセージ (Messages)

メッセージを複数持つメッセージクラスのインタフェースは、以下のようなフレームワーク制御に用いられる。

【OLTP】

画面に表示するメッセージとして設定する。

【バッチ】

業務ロジック用ログとして出力される。

【Rich】

業務ロジック用ログとして出力される。

2.3.3. 使用例

【業務ロジック実装例】

```
// 年齢のチェック
if (requestDto.getAge().isLess(context.createIntegerType(“20”))) {
    // 20歳未満の為、メッセージを設定
    responseDto.addMessage(context.createMessage(“mes.age”, “20”));
}
return;
```

【メッセージの生成方法】

```
context.createMessage({メッセージコード}, {埋め込み文字});
```

【メッセージリソース】

(例)

```
mes.age=年齢制限チェック {0}
```

{0}・・・埋め込み文字の記述。複数指定することが可能、0から始まる。

フレームワークではメッセージに設定されたメッセージコードと、埋め込み文字を元にメッセージリソースからメッセージ文字列を作成する。

上記の場合、「年齢制限チェック 20」のメッセージ文字列が作成される。

フレームワークでは処理形態に応じて、メッセージの扱いが異なるような実装を推奨する。

【オンライン】

画面に表示するメッセージとして設定する。

【バッチ】

業務ロジック用ログとして出力される。

2.4. DB、ファイルI/O、共有変数(jp.finaluna.api.dao)

業務ロジックから、外部リソース(ファイル、DB、共有変数など)へアクセスし照会/更新を行う。開発者が外部リソースへのアクセスを意識することなく、実装出来ることを目的にしている為、象的なAPIを提供する。この機能をDAOと呼称する。対象とする外部リソースの照会、更新の設定(実行するクエリ)は、メソッドパラメータに受け渡すクエリコードで識別する。QueryDAO、UpdateDAOはストアドプロシージャによる照会、更新も可能である。

外部リソースへのアクセス処理中に例外が発生した場合、例外をスローする。業務ロジックではこの例外を補足し処理の継続有無を判断することが必要になる。DAOが提供する機能を以下に示す。

分類	クラス名		説明
DB、ファイルI/O	DAO	データアクセスオブジェクト	外部データへアクセスするためのデータアクセスオブジェクトのマーカーインタフェース
	QueryDAO	照会用DAO	業務ロジックより外部リソースを取得するための検索用データアクセスオブジェクト
	UpdateDAO	更新用DAO	業務ロジックより外部リソースを更新するためのデータアクセスオブジェクト
	PageQueryDAO	ページ照会用DAO	業務ロジックより外部リソースを取得するための一覧検索用データアクセスオブジェクト
	CursorDAO	カーソル照会用DAO	業務ロジックより外部リソースをカーソルで取得するための検索用データアクセスオブジェクト
	StoredProcedureDAO	ストアドプロシージャ実行用DAO	業務ロジックよりストアドプロシージャにより外部リソースを取得、更新するためのデータアクセスオブジェクト
	EntityQueryDAO	エンティティ照会用DAO	外部データを主キー参照する為のデータアクセスオブジェクト
	EntityUpdateDAO	エンティティ更新用DAO	外部データに主キー更新する為のデータアクセスオブジェクト
チェック	SqlPreCheck	SQL実行前チェック	データベース処理を行う前にバインド値(DTO)の値チェックを行う
コード	QueryCode	クエリコード	クエリコード情報を保持する型のコード型

例外	FinalunaOptimisticLockFailureException	FINALUNA論理排他例外	参照した時点から更新時まで、他者によってレコードが更新されていた場合にスローされる例外
----	--	----------------	---

2.4.1. データアクセスオブジェクト (DAO)

【マーカーインタフェース】

外部データへアクセスするためのデータアクセスオブジェクトのマーカーインタフェース。

照会用DAO、更新用DAO、ページ照会用DAO、カーソルDAO、ストアドプロシージャDAO、エンティティ照会用DAO、エンティティ更新用DAOはこのインタフェースを継承している。

2.4.2. 照会用DAO (QueryDAO)

業務ロジックより外部リソースを取得するための検索用データアクセスオブジェクト。

クエリコードの設定に従い、外部リソースよりデータを取得する。取得したデータはDTOに変換し業務ロジックに返す。

2.4.3. 更新用DAO (UpdateDAO)

業務ロジックより外部リソースを更新するためのデータアクセスオブジェクト。

クエリコードの設定に従い、外部リソースのデータを更新する。更新した件数を業務ロジックに返す。

2.4.4. ページ照会用DAO (PageQueryDAO)

業務ロジックより外部リソースを取得するための一覧検索用データアクセスオブジェクト。

クエリコードの設定に従い、外部リソースより指定された範囲のデータを取得する。取得したデータはDTOに変換し業務ロジックに返す。

検索対象リソースより、指定ページ分の情報を取得する。

検索対象データの総件数は取得できないため、件数だけを取得する SQL を実行して取得する事。

2.4.5. カーソル照会用DAO (CursorDAO)

データベースへの検索を実施するDAOのインタフェース。
業務ロジックより外部リソースを取得するための検索用データアクセスオブジェクト。
クエリコードの設定に従い、外部リソースより指定された範囲のデータをカーソルで保持する。
業務ロジックからの要求に応じて順方向にデータを1レコードずつ取得し、DTOに変換して業務ロジックに返す。

2.4.6. ストアドプロシージャ実行用DAO (StoredProcedureDAO)

データベースへのストアドプロシージャを実行するDAOのインタフェース
業務ロジックからストアドプロシージャにより外部リソースを取得、更新するためのデータアクセスオブジェクト。
クエリコードの設定に従い、指定されたストアドプロシージャを実行する。

2.4.7. エンティティ照会用DAO (EntityQueryDAO)

外部データを主キー参照する為のデータアクセスオブジェクト。
主キーを条件にして、1レコードを格納したDTOを取得する。getメソッドは必ずキャッシュからデータを取得し、実際の外部データに対するアクセスを行わない。
主キークラスはhashCode及びequalsを適切に実装する必要がある。

2.4.8. エンティティ更新用DAO (EntityUpdateDAO)

外部データに主キー更新する為のデータアクセスオブジェクト。
主キーを条件にして、外部データ1レコードを更新する。更新は分散環境内の他JVMへ通知され、キャッシュが更新される。
主キークラスはhashCode及びequalsを適切に実装する必要がある。

2.4.9. SQL実行前チェック (SqlPreCheck)

DAO の更新処理 SQL 実行前チェック。

DAO の更新メソッドにて、処理を行う前に バインド値(DTO)の値のチェックを行う。

このインタフェースの check メソッドを実装し、設定ファイルの DAO に インジェクションすることで使用可能となる。

実装方法例：

```
// DATA2 項目のチェック長
private int data2Length = 0;

// SQL 実行前チェック。
public void check(ProcessContext context, DTO dto) throws FinalunaAppException {
    RequestDTO reqDto = (RequestDTO) dto;

    // DATA1 項目の必須チェック
    StringType data1 = (StringType) reqDto.get("DATA1");
    if ((data1 == null) || (data1.isNull())) {
        throw new CheckException(new IllegalArgumentException(), "SampleSqlCheck.01", "DATA1 is null");
    }

    // DATA2 項目の長さチェック
    StringType data2 = (StringType) reqDto.get("DATA2");
    if (data2.length() < this.data2Length) {
        throw new CheckException(new IllegalArgumentException(), "SampleSqlCheck.02", "Data2 is illegal to length");
    }
}

// DATA2 項目のチェック長を設定する。
public void setData2Length(int data2Length) {
    this.data2Length = data2Length;
}
```

2.4.10. クエリコード (QueryCode)

クエリコード情報を保持する型のコード型。

DAOの設定を識別する。

業務ロジックよりDAO(データアクセスオブジェクト)を実行する際のパラメータ。

このクエリコード型に応じてDAO(データアクセスオブジェクト)の実行を振り分ける。

2.4.11. FINALUNA論理排他例外 (FinalunaOptimisticLockFailureException)

参照した時点から更新時まで、他者によってレコードが更新されていた場合にスローされる例外

2.5. 業務処理 (jp.finaluna.api.blogic)

業務処理を実行する業務ロジックのAPI。

分類	クラス名		説明
業務ロジック	Logic	業務ロジック	業務ロジックが実装するインタフェース
	Context	コンテキスト	業務ロジックに必要な情報を保持するコンテキストインタフェース
	ProcessContext	プロセスコンテキスト	業務ロジックに必要な情報を保持するコンテキストインタフェース
	StartupLogic	業務起動時実行ロジック	業務起動時処理インタフェース
	StopLogic	業務停止時実行ロジック	業務停止時処理インタフェース ※OSS版FWでは未実装。
コード	BLogicCode	業務ロジックコード	業務ロジックを一意に識別するコード型
	DAOCode	DAOコード	データアクセスオブジェクト(DAO)を一意に識別するコード型
	DTOCode	DTOコード	データ転送オブジェクト(DTO)を一意に識別するコード型

2.5.1. 業務ロジック (Logic)

業務ロジックが実装するインタフェース。

【重要】FINALUNAフレームワークで実行する業務ロジックは、FINALUNA APIが提供するLogicインタフェースを実装する。フレームワークはこのLogicインタフェースを実装したクラスを業務ロジックと認識し、呼出す仕組みをとる。

業務ロジックは、FINALUNA APIが提供するAPIを使用して実装し、パラメータに入力値としてリクエストDTO、出力値としてレスポンスDTO、業務ロジック内で必要とする情報を保持するコンテキストを持つ。

インスタンスの生成は**context**機能を使用して取得する必要がある。

フレームワーク側で抽出したリクエストDTOを入力情報として処理を実装し、処理結果をレスポンスDTOに格納する。フレームワークがレスポンスDTOに格納されたデータを元に出力処理/制御処理を担う。

Note:

業務ロジック内では、処理中に発生する例外を意識しなくてよい。(例外処理はフレームワーク側で捕捉し処理する為)。

しかし、業務ロジック側で判断を有する例外は捕捉し、スローすべきである。

(例：トランザクションロールバックさせる為に、例外をキャッチし、ロールバック用例外としてスローする。)

2.5.2. コンテキスト (Context)

業務ロジックに必要な情報を保持するコンテキストインタフェース。

このオブジェクトには、業務ロジック内でFINALUNA型、DTO、DAOなどを生成するファクトリ機能と、サブ機能(サブ業務ロジック・パラメータ)を容易に使用する為の機能を持つ。業務ロジックはFINALUNA APIが提供するAPIを使用して実装する為、このコンテキストを使用してFINALUNA APIのインスタンスを生成する。

以下の情報を取得することが出来る。(ファクトリ機能)

No	名称	備考
1	Finaluna 型 文字列	値を指定して取得可能
2	Finaluna 型 整数	値を指定して取得可能
3	Finaluna 型 小数	値を指定して取得可能
4	Finaluna 型 日付	1970 年 1 月 1 の日付。値を指定して取得可能
5	Finaluna 型 時刻	時間(0 時 0 分 0 秒)の時刻。値を指定して取得可能
6	Finaluna 型 日時	1970 年 1 月 1 日 時間(0 時 0 分 0 秒)の日時。値を指定して取得可能
7	Finaluna 型 日時(ナノ秒)	1970 年 1 月 1 日時間(0 時 0 分 0 秒ナノ秒)の日時。値を指定して取得可能
8	DTO	指定されたコードに従い、コンテキスト情報より DTO を取得する。
9	DAO	指定されたコードに従い、コンテキスト情報より DAO を取得する。
10	メッセージ	業務メッセージ。
11	ステータス	応答ステータス。

以下の機能を有する。(ファサード機能)

No	名称	備考
1	業務ロジックの呼出し	
2	パラメータ取得	バッチでのみ使用可能

2.5.3. 業務コンテキスト (ProcessContext)

業務ロジックに必要な情報を保持するコンテキストインタフェース。

このオブジェクトには、業務ロジック内で FINALUNA 型、DTO、DAO などを生成するファクトリ機能を容易に使用する為の機能を持つ。業務ロジックは FINALUNA API が提供する API を使用して実装する為に、このコンテキストを使用して FINALUNA API のインスタンスを生成する。

以下の情報を取得することが出来る。(ファクトリ機能)

No	名称	備考
1	Finaluna 型 文字列	値を指定して取得可能
2	Finaluna 型 整数	値を指定して取得可能
3	Finaluna 型 小数	値を指定して取得可能
4	Finaluna 型 日付	1970 年 1 月 1 日の日付。値を指定して取得可能
5	Finaluna 型 時刻	時間(0 時 0 分 0 秒)の時刻。値を指定して取得可能
6	Finaluna 型 日時	1970 年 1 月 1 日 時間(0 時 0 分 0 秒)の日時。値を指定して取得可能
7	Finaluna 型 日時(ナノ秒)	1970 年 1 月 1 日 時間(0 時 0 分 0 秒ナノ秒)の日時。値を指定して取得可能

2.5.4. 起動時実行ロジック (StartupLogic)

業務起動時処理を実装する場合には StartupLogic インタフェースを実装する。

起動時処理には通常の業務ロジックと同様に FINALUNA API を利用して実装する。

また、トランザクション管理、例外処理についてもフレームワーク側で処理するため 意識する必要がない。

2.5.5. 業務停止時実行ロジック (StopLogic)

業務停止時処理を実装する場合には StopLogic インタフェースを実装する。

停止時処理には通常の業務ロジックと同様に FINALUNA API を利用して実装する。

また、トランザクション管理、例外処理についてもフレームワーク側で処理するため 意識する必要がない。

2.5.6. 業務ロジックコード (BLogicCode)

業務ロジックを一意に識別するコード型。

業務ロジックから更に業務ロジックを呼出す場合に、呼出す業務ロジックを指定する為のキー値を保持する。

2.5.7. DAOコード (DAOCode)

外部データへアクセスするためのデータアクセスオブジェクトのマーカインタフェース。

照会用DAO、更新用DAO、ページ照会用DAO、カーソルDAO、ストアードプロシージャDAOはこのインタフェースを継承している。

2.5.8. DTOコード (DTOCode)

データ転送オブジェクト(DTO)を一意に識別するコード型。

業務ロジックからDTO(データ転送オブジェクト)を使用する場合、使用するDTO(データ転送オブジェクト)を指定する為のキー値を保持する。

2.5.9. 使用例

コンテキストの使用例を以下に示す。

【インスタンスの生成】

```
// 値〇〇会社を保持した文字列型の生成
StringType company = context.createStringType(“〇〇会社”);

// リスト型の生成
ListType list = context.createListType();

// レスポンス DTO の生成 DTO コードを指定して取得する
ResponseDTO resDto = context.createResponseDTO(DTO_CODE);

// DAO の生成 DAO コードを指定して取得する
QueryDAO dao = context.createQueryDAO(DAO_CODE);

// サブ業務ロジック呼出し
context.execute(requestDto, responseDto, BLOGIC_CODE);
```

2.6. 出口処理 (jp.finaluna.api.exit)

出口処理を実行するAPI

分類	クラス名		説明
出口処理	LogicPostExit	ロジック後出口処理	業務ロジック後出口インタフェース
	LogicPreExit	ロジック前出口処理	業務ロジック前出口インタフェース

2.6.1. ロジック後出口 (LogicPostExit)

ロジック後の出口処理実行インタフェース。
業務ロジック後の出口処理はこのインタフェースを実装する。

2.6.2. ロジック前出口 (LogicPreExit)

ロジック前の出口処理実行インタフェース。
業務ロジック前の出口処理はこのインタフェースを実装する。

2.7. 例外処理 (jp.finaluna.api.exception)

処理内で発生した例外を特定する為に、FINALUNAフレームワークでは以下の分類の例外を使用する。

分類	クラス名		説明
例外処理	FinalunaApplException	Finaluna業務例外	業務ロジックがスローする例外(業務例外)
	FinalunaSysException	Finalunaシステム例外	システム例外を表す非検査例外
	FinalunaUnsupportedOperationException	Unsupport例外	サポートしないメソッドが呼び出された場合の例外。
メッセージインタフェース	ErrorMessageHolder	エラーメッセージインタフェース	エラーメッセージを保持するためのインタフェース

2.7.1. Finaluna業務例外 (FinalunaApplException)

業務ロジックがスローする例外(業務例外)。

業務ロジック実行中に発生した修復可能な問題を実行側に通知するための例外。この例外は検査例外の為、フレームワーク側でcatchする必要がある。

FinalunaApplException 例外は、業務ロジックまたはユーティリティクラスで スローされる検査例外である。

FinalunaApplException のインスタンス生成時には、以下のパラメータを設定する。

- (1) エラーとなった原因となる例外 (スローすべき例外)
- (2) エラーコード
- (3) メッセージID
- (4) メッセージ埋め込みオブジェクト

2.7.2. Finalunaシステム例外 (FinalunaSysException)

システム例外を表す非検査例外。

処理中の予期せぬ問題、または修復不可能な問題をシステム管理者に通知するための例外。この例外は非検査例外の為、業務ロジックでcatchする必要なく、フレームワークでのみcatchすればよい。

2.7.3. Unsupport例外 (FinalunaUnsupportedOperationException)

サブクラスがオーバーライドしたメソッドで、その機能をサポートしない場合にスローされる。

サポートしないメソッドが呼び出された場合の例外。

2.7.4. エラーメッセージホルダー (ErrorMessageHolder)

エラーメッセージを保持するためのインタフェース。

アプリケーションにおける例外クラスのメッセージ部分を共通化したインタフェース。

2.8. ロガー (jp.finaluna.api.log)

業務ロジック内で使用するロガーを提供する。

分類	クラス名		説明
ファクトリ	LogFactory	ロガーファクトリ	業務ロジックにて使用されるロガーのインスタンス生成を行うためのクラス
	LogFactoryDelegate	ロガー生成ファクトリインタフェース	業務ロジックにて使用されるロガーのインスタンス生成インタフェース
ロガー	Logger	ロガーインタフェース	FINALUNAでログ出力機能を使用するためのインタフェース

2.8.1. ロガーファクトリ (LogFactory)

ロガーを生成するためのファクトリ。

業務ロジッククラスにて使用されるロガーのインスタンス生成には、このロガー生成ファクトリクラスを用いる。

例)

```
public class Blogic implements logic {
    private static Logger log = LogFactory.getLogger(Blogic.class);
```

2.8.2. ロガー生成ファクトリインタフェース (LogFactoryDelegate)

ロガーを生成するのファクトリインタフェース。

ロガーを生成するファクトリ実装クラスは、このLogFactoryDelegateインタフェースを実装する。

例)

- spring設定ファイル
実装クラスをspringのbean定義として設定

```
<!-- ログFactory -->
<bean id="LogFactoryDelegate" class="jp.finaluna.fw.common.context.util.log.LogFactoryDelegateImpl" />
```

- LogFactoryDelegateの実装クラス
LogFactoryDelegateの実装クラスは、springのInitializedBeanを実装し、InitialaizedBeanのafterPropertiesSetメソッドにファクトリクラスに実体をセットする処理を用意する。

```
public class LogFactoryDelegateImpl implements LogFactoryDelegate, InitializingBean {
    public void afterPropertiesSet() throws Exception {
        LogFactory.setLogFactory(this);
    }
}
```

2.8.3. ロガーインタフェース (Logger)

Finalunaでログ出力機能を実現するためのインタフェース。

例)

```
public class Blogic implements logic {
    private static Logger log = LogFactory.getLogger(Blogic.class);

    public void execute(Context context, DTO requestDTO, DTO responseDTO) throws FinalunaApplException {
        log.info(this.getClass().toString() + "#execute() start.");
        log.info("log.message", requestDTO);
    }
}
```

3. FINALUNA API クラス構成

3.1. FINALUNA API クラス構成

FINALUNA APIのクラス構成は「APIクラス図.xls」参照

4. 補足

4.1. JavaDoc

FINALUNA APIのJavaDocは「FinalunaAPI_JavaDoc.zip」参照。

以下にFINALUNA APIのメソッド一覧を示す。

4.1.1. FINALUNA型 (jp. finaluna. api. type)

表4. 1. 1-1 StringType

クラス	StringType	
説明	可変文字列を保持する型	
戻り値	メソッド	説明
StringType	append(CharSequenceType value)	この文字列に CharSequenceType 引数の文字列を追加する。
StringType	capitalize()	この文字列の先頭文字のみを大文字に変換する。変換対象の文字はアルファベットのみとする。
StringType	delete(int start, int end)	この文字列の指定位置の文字列を削除する。元の値は指定位置の文字列が削除された値となる。
boolean	endsWith(CharSequenceType suffix)	この文字列が、指定された接尾語で終了しているかどうかを判定する。
boolean	equalsIgnoreCase(CharSequenceType compareStr)	この文字列が、比較対象の文字列と等しいかどうかを、大文字、小文字の区別なしに判定する。比較対象の文字はアルファベットのみとする。
int	indexOf(CharSequenceType searchStr)	この文字列で、指定された文字列が最初に出現する位置のインデックスを取得する。
int	indexOf(CharSequenceType searchStr, int fromIndex)	指定インデックス以降で、文字列が最初に出現する位置のインデックスを取得する。
StringType	insert(int index, CharSequenceType value)	この文字列の指定位置に、CharSequenceType 引数の文字列を追加する。
boolean	isAlpha()	この文字列が、英字のみで構成されているかどうかを判定する。
boolean	isNumeric()	この文字列が、数字のみで構成されているかどうかを判定する。
boolean	isNumericAlpha()	この文字列が、英数字のみで構成されているかどうかを判定する。
int	lastIndexOf(CharSequenceType searchStr)	この文字列で、指定された文字列が最後に出現する位置のインデックスを取得する。
int	lastIndexOf(CharSequenceType searchStr, int fromIndex)	指定インデックス以前で、文字列が最初に出現する位置のインデックスを取得する。
boolean	matches(CharSequenceType regex)	この文字列が、指定された正規表現と一致するかどうかを判定する。
boolean	search(CharSequenceType searchStr)	指定した文字列が、この文字列内に存在するかどうかを判定する。
boolean	startsWith(CharSequenceType prefix)	この文字列が、指定された接頭語で始まるかどうかを判定する。
StringType	substring(int beginIndex)	この文字列の部分文字列を返す。
StringType	substring(int beginIndex, int endIndex)	この文字列の部分文字列を返す。
StringType	toLowerCase()	この文字列を、すべて小文字に変換する。
StringType	toUpperCase()	この文字列を、すべて大文字に変換する。
StringType	trim()	この文字列の先頭と末尾の空白を取り除いた文字列型を返します。

表4. 1. 1-2 StringCode

クラス	StringCode	
説明	固定文字列を保持する型	
戻り値	メソッド	説明

クラス	StringCode	
説明	固定文字列を保持する型	
戻り値	メソッド	説明
StringCode	capitalize()	この文字列の先頭文字のみを大文字に変換する。変換対象の文字はアルファベットのみとする。
boolean	contains()	この文字列が指定された検索文字列を含む場合に限り true を返す。
boolean	endsWith(CharSequenceType suffix)	この文字列が、指定された接尾語で終了しているかどうかを判定する。
boolean	equalsIgnoreCase(CharSequenceType compareStr)	この文字列が、比較対象の文字列と等しいかどうかを、大文字、小文字の区別なしに判定する。比較対象の文字はアルファベットのみとする。
int	indexOf(CharSequenceType searchSequence)	この文字列で、指定された文字列が最初に出現する位置のインデックス を取得する。
int	indexOf(CharSequenceType searchSequence, int fromIndex)	指定インデックス以降で、文字列が最初に出現する位置のインデックス を取得する。
boolean	isAlpha()	この文字列が、英字のみで構成されているかどうかを判定する。
boolean	isNumeric()	この文字列が、数字のみで構成されているかどうかを判定する。
boolean	isNumericAlpha()	この文字列が、英数字のみで構成されているかどうかを判定する。
int	lastIndexOf(CharSequenceType searchSequence)	この文字列で、指定された文字列が最後に出現する位置のインデックス を取得する。
int	lastIndexOf(CharSequenceType searchSequence, int fromIndex)	指定インデックス以前で、文字列が最初に出現する位置のインデックス を取得する。
boolean	matches(CharSequenceType regex)	この文字列が、指定された正規表現と一致するかどうかを判定する。
StringCode	replace(CharSequenceType target, CharSequenceType replacement)	この文字列の指定された置換対象文字列を指定された置換文字列に置き換えた文字列を返す。
StringCode	replaceAll(CharSequenceType regex, CharSequenceType replacement)	指定された正規表現に一致する、この文字列の各部分文字列に対し、指定された置換を実行した文字列を返す。
boolean	search(CharSequenceType searchSequence)	指定した文字列が、この文字列内に存在するかどうかを判定する。
StringCode[]	split(CharSequenceType regex)	この文字列を、指定された正規表現に一致する位置で分割した文字列を返す。
boolean	startsWith(CharSequenceType prefix)	この文字列が、指定された接頭語で始まるかどうかを判定する。
StringCode	substring(int beginIndex)	この文字列の部分文字列を返す。
StringCode	substring(int beginIndex, int endIndex)	この文字列の部分文字列を返す。
StringCode	toLowerCase()	この文字列を、すべて小文字に変換する。
StringCode	toUpperCase()	この文字列を、すべて大文字に変換する。
StringCode	trim()	この文字列の先頭と末尾の空白を取り除いた文字列型を返します。

表4.1.1-3 CharSequenceType

クラス	CharSequenceType	
説明	文字シーケンスを保持する型	
戻り値	メソッド	説明
int	length()	この文字列の長さを取得する。
boolean	contentEquals(CharSequenceType value)	この文字列が、指定されたシーケンスと同じ char 値のシーケンスを表す場合にだけ、True を返す。
char	charAt(int index)	指定されたインデックス位置にある char 値を返す。インデックスは、0 から length() - 1 の範囲。シーケンスの最初の char 値のインデックスは 0、次の文字のインデックスは 1。

表4.1.1-4 CodeType

クラス	CodeType.	
説明	コード値を表す型	
戻り値	メソッド	説明
	なし	

表4.1.1-5 KeyCodeType

クラス	KeyCodeType	
説明	更新可能なコード型	
戻り値	メソッド	説明
StringType	get()	コード値の取得を行う。
void	set(StringType value)	コード値を設定する。

表4.1.1-6 IntegerType

クラス	IntegerType	
説明	整数を保持する型	
戻り値	メソッド	説明
IntegerType	abs()	この数値の絶対値を取得する。
DecimalType	add(DecimalType val)	この数値に、指定した値を加算した結果を取得する。
DecimalType	add(DecimalType val, int scale, MathUtil.Round round)	この数値に、指定した値を加算した結果を取得する。
IntegerType	add(int val)	この数値に、指定した値を加算した結果を取得する。
IntegerType	add(IntegerType val)	この数値に、指定した値を加算した結果を取得する。
DecimalType	divide(DecimalType val, int scale)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(DecimalType val, int scale, MathUtil.Round round)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(int val, int scale)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(int val, int scale, MathUtil.Round round)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(IntegerType val, int scale)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(IntegerType val, int scale, MathUtil.Round round)	この数値に、指定した値を除算した結果を取得する。
boolean	isGreater(IntegerType compareNum)	この数値が指定された数値より大きいかどうかを判定する。
boolean	isGreaterEqual(IntegerType compareNum)	この数値が指定された数値より大きいかどうかを判定する。
boolean	isInRange(IntegerType small, IntegerType large, MathUtil.InRange range)	この数値が、指定された値、境界値条件で範囲内にあるかどうか判定する。
boolean	isLess(IntegerType compareNum)	この数値が指定された数値より小さいかどうかを判定する。
boolean	isLessEqual(IntegerType compareNum)	この数値が指定された数値より小さいかどうかを判定する。
DecimalType	multiply(DecimalType val)	この数値に、指定した値を乗算した結果を取得する。
DecimalType	multiply(DecimalType val, int scale, MathUtil.Round round)	この数値に、指定した値を乗算した結果を取得する。
IntegerType	multiply(int val)	この数値に、指定した値を乗算した結果を取得する。
IntegerType	multiply(IntegerType val)	この数値に、指定した値を乗算した結果を取得する。
IntegerType	negate()	この数値と絶対値が等しい負の値を取得する。
IntegerType	remainder(int val)	この数値を、指定の値で除算した剰余を取得する。
IntegerType	remainder(IntegerType val)	この数値を、指定の値で除算した剰余を取得する。

クラス	IntegerType	
説明	整数を保持する型	
戻り値	メソッド	説明
int	signum()	この数値の符号要素を返す。
DecimalType	subtract(DecimalType val)	この数値に、指定した値を減算した結果を取得する。
DecimalType	subtract(DecimalType val, int scale, MathUtil.Round round)	この数値に、指定した値を減算した結果を取得する。
IntegerType	subtract(int val)	この数値に、指定した値を減算した結果を取得する。
IntegerType	subtract(IntegerType val)	この数値に、指定した値を減算した結果を取得する。

表4.1.1-7 DecimalType

クラス	DecimalType	
説明	小数を保持する型	
戻り値	メソッド	説明
DecimalType	abs()	この数値の絶対値を取得する。
DecimalType	add(DecimalType augend)	この数値に、指定した値を加算した結果を取得する。
DecimalType	add(DecimalType augend, int scale, MathUtil.Round round)	この数値に、指定した値を加算した結果を取得する。
DecimalType	add(int augend)	この数値に、指定した値を加算した結果を取得する。
DecimalType	add(int augend, int scale, MathUtil.Round round)	この数値に、指定した値を加算した結果を取得する。
DecimalType	add(IntegerType augend)	この数値に、指定した値を加算した結果を取得する。
DecimalType	add(IntegerType augend, int scale, MathUtil.Round round)	この数値に、指定した値を加算した結果を取得する。
DecimalType	divide(DecimalType divisor, int scale)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(DecimalType divisor, int scale, MathUtil.Round round)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(int divisor, int scale)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(int divisor, int scale, MathUtil.Round round)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(IntegerType divisor, int scale)	この数値に、指定した値を除算した結果を取得する。
DecimalType	divide(IntegerType divisor, int scale, MathUtil.Round round)	この数値に、指定した値を除算した結果を取得する。
IntegerType	getInteger()	この数値を、整数値に変換する。
IntegerType	getInteger(MathUtil.Round round)	この数値を、指定された丸めモードを適用した整数値で取得する。
boolean	isGreater(DecimalType compareNum)	この数値が指定された数値より大きいかどうかを判定する。
boolean	isGreaterEqual(DecimalType compareNum)	この数値が指定された数値より大きいかどうかを判定する。
boolean	isInRange(DecimalType small, DecimalType large, MathUtil.InRange range)	この数値が、指定された値、境界値条件で範囲内にあるかどうか判定する。
boolean	isLess(DecimalType compareNum)	この数値が指定された数値より小さいかどうかを判定する。
boolean	isLessEqual(DecimalType compareNum)	この数値が指定された数値より小さいかどうかを判定する。

クラス	DecimalType	
説明	小数を保持する型	
戻り値	メソッド	説明
DecimalType	movePointLeft(int point)	この数値の小数点を、指定された値だけ左へ移動した値を取得する。
DecimalType	movePointRight(int point)	この数値の小数点を、指定された値だけ右へ移動した値を取得する。
DecimalType	multiply(DecimalType multiplicand)	この数値に、指定した値を乗算した結果を取得する。
DecimalType	multiply(DecimalType multiplicand, int scale, MathUtil.Round round)	この数値に、指定した値を乗算した結果を取得する。
DecimalType	multiply(int multiplicand)	この数値に、指定した値を乗算した結果を取得する。
DecimalType	multiply(int multiplicand, int scale, MathUtil.Round round)	この数値に、指定した値を乗算した結果を取得する。
DecimalType	multiply(IntegerType multiplicand)	この数値に、指定した値を乗算した結果を取得する。
DecimalType	multiply(IntegerType multiplicand, int scale, MathUtil.Round round)	この数値に、指定した値を乗算した結果を取得する。
DecimalType	negate()	この数値に-1を乗じた値を取得する。
DecimalType	pow(int exponent, int precision, MathUtil.Round round)	この数値を、指定の値でべき乗した値を取得する。
int	precision()	この数値の「精度」を返します。精度とは、スケールなしの値の桁数のことです。0の精度は1です。
DecimalType	remainder(DecimalType divisor)	この数値を、指定の値で除算した剰余を取得する。
DecimalType	remainder(DecimalType divisor, int scale, MathUtil.Round round)	この数値を、指定の値で除算した剰余を取得する。
DecimalType	remainder(int divisor)	この数値を、指定の値で除算した剰余を取得する。
DecimalType	remainder(int divisor, int scale, MathUtil.Round round)	この数値を、指定の値で除算した剰余を取得する。
int	scale()	この数値の「スケール」を返します。0 または正の場合、スケールは小数点以下の桁数です。負の場合、スケールなしの数値に、スケールの正負を逆にした値を指数とする10の累乗を乗算します。たとえば、-3のスケールでは、スケールなしの値に1000が乗算されます。
DecimalType	setScale(int scale)	このスケールで設定された数値を返す。
DecimalType	setScale(int scale, MathUtil.Round round)	このスケール、丸めモードで設定された数値を返す。
int	signum()	この数値の符号要素を返します。
DecimalType	subtract(DecimalType subtrahend)	この数値に、指定した値を減算した結果を取得する。
DecimalType	subtract(DecimalType subtrahend, int scale, MathUtil.Round round)	この数値に、指定した値を減算した結果を取得する。
DecimalType	subtract(int subtrahend)	この数値に、指定した値を減算した結果を取得する。
DecimalType	subtract(int subtrahend, int scale, MathUtil.Round round)	この数値に、指定した値を減算した結果を取得する。
DecimalType	subtract(IntegerType subtrahend)	この数値に、指定した値を減算した結果を取得する。
DecimalType	subtract(IntegerType subtrahend, int scale,	この数値に、指定した値を減算した結果を取得する。

クラス	DecimalType	
説明	小数を保持する型	
戻り値	メソッド	説明
	MathUtil.Round round)	

表4.1.1-8 DateType

クラス	DateType	
説明	年月日を保持する型	
戻り値	メソッド	説明
DateType	add(DateUtil.Calendar field, int amount)	この日付の日付フィールドに対し、指定された値で加減算を行う。
DateType	add(DateUtil.Calendar field, IntegerType amount)	この日付の日付フィールドに対し、指定された値で加減算を行う。
int	get(DateUtil.Calendar field)	この日付の指定フィールドを取得する。
DateType	getMonthOfFirstDay()	この日付の月初の日付を返す。
DateType	getMonthOfLastDay()	この日付の月末の日付を返す。
DateUtil.DayOfWeek	getWeek()	この日付の曜日を返す。
boolean	isAfter(DateType compareDate)	この日付が、指定した日付より後にあるかどうかを判定する。
boolean	isBefore(DateType compareDate)	この日付が、指定した日付より前にあるかどうかを判定する。
boolean	isInRange(DateType startDate, DateType endDate, MathUtil.InRange range)	この日付が、指定された値、境界値条件で範囲内にあるかどうか判定する。
boolean	isOnAndAfter(DateType compareDate)	この日付が、指定した日付より後にあるかどうかを判定する。
boolean	isOnAndBefore(DateType compareDate)	この日付が、指定した日付より前にあるかどうかを判定する。
DateType	set(DateUtil.Calendar field, int amount)	この日付の指定フィールドに対し、指定された値を設定する。
DateType	set(DateUtil.Calendar field, IntegerType amount)	この日付の指定フィールドに対し、指定された値を設定する。

表4.1.1-9 TimeType

クラス	TimeType	
説明	時分秒を保持する型	
戻り値	メソッド	説明
TimeType	add(DateUtil.Calendar field, int amount)	この時刻の日付フィールドに対し、指定された値で加減算を行う。
TimeType	add(DateUtil.Calendar field, IntegerType amount)	この時刻の日付フィールドに対し、指定された値で加減算を行う。
int	get(DateUtil.Calendar field)	この時刻の指定フィールドを取得する。
boolean	isAfter(TimeType compareTime)	この時刻が、指定した時刻より後にあるかどうかを判定する。
boolean	isBefore(TimeType compareTime)	この時刻が、指定した時刻より前にあるかどうかを判定する。
boolean	isInRange(TimeType startTime, TimeType endTime, MathUtil.InRange range)	この時刻が、指定された値、境界値条件で範囲内にあるかどうか判定する。
boolean	isOnAndAfter(TimeType compareTime)	この時刻が、指定した時刻より後にあるかどうかを判定する。
boolean	isOnAndBefore(TimeType compareTime)	この時刻が、指定した時刻より前にあるかどうかを判定する。

TimeType	set(DateUtil.Calendar field, int amount)	この時刻の指定フィールドに対し、指定された値を設定する。
TimeType	set(DateUtil.Calendar field, IntegerType amount)	この時刻の指定フィールドに対し、指定された値を設定する。

表4.1.1-10 TimestampType

クラス	TimestampType	
説明	年月日時分秒 ナノ秒を保持する型。	
戻り値	メソッド	説明
TimestampType	add(DateUtil.Calendar field, int amount)	この日付時刻の日付フィールドに対し、指定された値で加減算を行う。
TimestampType	add(DateUtil.Calendar field, IntegerType amount)	この日付時刻の日付フィールドに対し、指定された値で加減算を行う。
int	get(DateUtil.Calendar field)	この日付時刻の指定フィールドを取得する。
boolean	isAfter(TimestampType compareTime)	この日付時刻が、指定した日付時刻より後にあるかどうかを判定する。
boolean	isBefore(TimestampType compareTime)	この日付時刻が、指定した日付時刻より前にあるかどうかを判定する。
boolean	isInRange(TimestampType startTime, TimestampType endTime, MathUtil.InRange range)	この日付時刻が、指定された値、境界値条件で範囲内にあるかどうかを判定する。
boolean	isOnAndAfter(TimestampType compareTime)	この日付時刻が、指定した日付時刻より後にあるかどうかを判定する。
boolean	isOnAndBefore(TimestampType compareTime)	この日付時刻が、指定した日付時刻より前にあるかどうかを判定する。
TimestampType	set(DateUtil.Calendar field, int amount)	この日付時刻の指定フィールドに対し、指定された値を設定する。
TimestampType	set(DateUtil.Calendar field, IntegerType amount)	この日付時刻の指定フィールドに対し、指定された値を設定する。

表4.1.1-11 DateAndTimeType

クラス	DateAndTimeType	
説明	年月日時分秒を保持する型	
戻り値	メソッド	説明
DateAndTimeType	add(DateUtil.Calendar field, int amount)	この日付時刻の日付フィールドに対し、指定された値で加減算を行う。
DateAndTimeType	add(DateUtil.Calendar field, IntegerType amount)	この日付時刻の日付フィールドに対し、指定された値で加減算を行う。
int	get(DateUtil.Calendar field)	この日付時刻の指定フィールドを取得する。
DateAndTimeType	getMonthOfFirstDay()	この日付の月初の日付を返す。
DateAndTimeType	getMonthOfLastDay()	この日付の月末の日付を返す。
DateUtil.DayOfWeek	getWeek()	この日付の曜日を返す。
boolean	isAfter(TimestampType compareTime)	この日付時刻が、指定した日付時刻より後にあるかどうかを判定する。
boolean	isBefore(TimestampType compareTime)	この日付時刻が、指定した日付時刻より前にあるかどうかを判定する。
boolean	isInRange(TimestampType startTime, TimestampType endTime, MathUtil.InRange range)	この日付時刻が、指定された値、境界値条件で範囲内にあるかどうかを判定する。
boolean	isOnAndAfter(TimestampType compareTime)	この日付時刻が、指定した日付時刻より後にあるかどうかを判定する。
boolean	isOnAndBefore(TimestampType compareTime)	この日付時刻が、指定した日付時刻より前にあるかどうかを判定する。
DateAndTimeType	set(DateUtil.Calendar field, int amount)	この日付時刻の指定フィールドに対し、指定された値を設定する。
DateAndTimeType	set(DateUtil.Calendar field, IntegerType amount)	この日付時刻の指定フィールドに対し、指定された値を設定する。

表4. 1. 1-12 ListType

クラス	ListType<E>	
説明	リストを保持する型	
戻り値	メソッド	説明
void	add(int index, E element)	リスト内の指定された位置に、指定された要素を挿入する。
E	get(int index)	リスト内の指定された位置にある要素を返す。
E	remove(int index)	リスト内の指定された位置にある要素を削除する。
E	set(int index, E element)	リスト内の指定された位置にある要素を、指定された要素に置き換える。

表4. 1. 1-13 MapType

クラス	MapType<K, V>	
説明	マップを保持する型	
戻り値	メソッド	説明
interface	Entry<K, V>	内部インタフェースクラス。
void	clear()	マップからマッピングをすべて削除する。
boolean	containsKey(Java.lang.Object key)	マップが指定のキーのマッピングを保持する場合に true を返す。つまり、このマップに、(key==null ? k==null : key.equals(k)) となるキー k が含まれている場合にだけ true を返す。
boolean	containsValue(Java.lang.Object value)	マップが 1 つまたは複数のキーと指定された値をマッピングしている場合に true を返す。つまり、マップに、(value==null ? v==null : value.equals(v)) となる値 v へのマッピングが 1 つ以上ある場合にだけ true を返す。
SetType <MapType.Entry <K, V>>	entrySet()	このマップに含まれるマップの Set ビューを返す。
boolean	equals(Java.lang.Object o)	指定されたオブジェクトがこのマップと等しいかどうかを比較する。指定されたオブジェクトがマップであり、2 つのマップが同じマッピングを表す場合に true を返す。
V	get(Java.lang.Object key)	指定されたキーがマップされている値を返す。そのキーのマッピングがこのマップに含まれていない場合は null を返す。
int	hashCode()	マップのハッシュコード値を返す。マップのハッシュコードは、マップの entrySet() ビューにある各エントリのハッシュコードの合計。
boolean	isEmpty()	マップがキーと値のマッピングを保持しない場合に true を返す。
SetType <K>	keySet()	このマップに含まれるキーの Set ビューを返す。
V	put(K key, V value)	指定された値と指定されたキーをこのマップに関連付ける。マップにすでにこのキーに対するマッピングがある場合、古い値は指定された値に置き換えられる。
void	putAll(MapType <? extends K, ? extends V> m)	指定されたマップのすべてのマッピングをこのマップにコピーする。
V	remove(Java.lang.Object key)	キーのマッピングがある場合に、そのマッピングをこのマップから削除する。
int	size()	マップ内のキー値マッピングの数を返す。マップに Integer.MAX_VALUE より多くの要素がある場合は、Integer.MAX_VALUE を返す。

クラス	MapType<K, V>	
説明	マップを保持する型	
戻り値	メソッド	説明
CollectionType <V>	values()	このマップに含まれる値の Collection ビューを返す。

表4.1.1-14 Entry

クラス	Entry<K, V>	
説明	マップのエントリ (キーと値のペア) (MapTypeの内部インタフェースクラス)	
戻り値	メソッド	説明
boolean	equals(Java.lang.Object o)	指定されたオブジェクトがエントリと等しいかどうかを比較する。
K	getKey()	エントリに対応するキーを返す。
V	getValue()	エントリに対応する値を返す。
int	hashCode()	このマップエントリのハッシュコード値を返す。
V	setValue(V value)	エントリに対応する値を、指定された値に置き換える。

表4.1.1-15 SetType

クラス	SetType<E>	
説明	セットを保持する型	
戻り値	メソッド	説明
	なし	

表4.1.1-16 CollectionType

クラス	CollectionType<E>	
説明	コレクションを保持する型	
戻り値	メソッド	説明
boolean	add(E e)	指定された要素がこのコレクションに格納されていることを保証する。この呼び出しの結果、コレクションが変更された場合は true を返す。このコレクションが要素の重複を許可せず、指定された要素がすでに含まれている場合は false を返す。
boolean	addAll(CollectionType <? extends E> c)	指定されたコレクションのすべての要素をこのコレクションに追加する。
void	clear()	このコレクションからすべての要素を削除する。
boolean	contains(Java.lang.Object o)	コレクションに指定された要素がある場合に true を返す。 コレクションに、(o==null ? e==null : o.equals(e)) となる要素 e が 1 つ以上含まれている場合にだけ true を返す。
boolean	equals(Java.lang.Object o)	指定されたオブジェクトとこのコレクションが等しいかどうかを比較する。
int	hashCode()	コレクションのハッシュコード値を返す。Object.equals メソッドをオーバーライドするクラスは、Object.hashCode メソッドもオーバーライドすること。
boolean	isEmpty()	コレクションに要素がない場合に true を返す。

クラス	CollectionType<E>	
説明	コレクションを保持する型	
戻り値	メソッド	説明
Iterator <E>	iterator()	コレクションの要素の反復子を返す。要素が返される順序についての保証はしない。ただし、このコレクションが、保証を提供するクラスのインスタンスである場合は例外。
boolean	remove(Java.lang.Object o)	指定された要素のインスタンスがこのコレクションにあれば、そのインスタンスをコレクションから 1 つ削除する。
int	size()	このコレクション中の要素の数を返す。
Java.lang.Object []	toArray()	このコレクションの要素がすべて格納されている配列を返す。反復子によって要素が返される順序をコレクションが保証する場合、このメソッドは同じ順序で要素を返す必要がある。

表4.1.1-17 BaseType

クラス	BaseType	
説明	全ての型の基となる型	
戻り値	メソッド	説明
boolean	equals(obj)	このオブジェクトと指定されたオブジェクトが等しいか判定する。
StringCode	toStringCode()	このオブジェクトが保持している値の文字列を StringCode で返す。

表4.1.1-18 CursorSet

クラス	CursorSet	
説明	データベースから検索結果を一件ずつ取得するためのインタフェース。	
戻り値	メソッド	説明
void	close()	java.sql.ResultSet をクローズする。
boolean	hasNext()	次のレコードの有無を返却する。
T	next()	取得されている次のレコードを DTO として返却する。

表4.1.1-19 FinalunaFormatException

クラス	FinalunaFormatException	
説明	フォーマット・解析処理例外	
戻り値	メソッド	説明
FinalunaFormatException	FinalunaFormatException(java.lang.String errorCode, java.lang.String messageCode)	コンストラクタ。
FinalunaFormatException	FinalunaFormatException(java.lang.String errorCode, java.lang.String messageCode, Java.lang.Object... args)	コンストラクタ。
FinalunaFormatException	FinalunaFormatException(Throwable cause)	コンストラクタ。

FinalunaFormatException	FinalunaFormatException(Throwable cause, java.lang.String errorCode)	コンストラクタ。
FinalunaFormatException	FinalunaFormatException(Throwable cause, java.lang.String errorCode, java.lang.String messageCode)	コンストラクタ。

表4.1.1-20 Copiable

クラス	Copiable <T>	
説明	インスタンスのコピーを戻すインタフェース	
戻り値	メソッド	説明
T	deepCopy()	このインスタンスのコピーを戻す。

表4.1.1-21 Resid

クラス	Resid	
説明	リビジョン番号、日付情報を保持するアノテーション	
戻り値	メソッド	説明
String	value()	リビジョン番号、日付情報を返す。

表4.1.1-22 ConstantFactoryDelegate

クラス	ConstantFactoryDelegate	
説明	Finaluna型を生成する為のファクトリインタフェース	
戻り値	メソッド	説明
<REQ extends DTO, RES extends DTO> BLogicCode <REQ, RES>	createBLogicCode(java.lang.String value)	BLogicCode を生成する。
<T extends DAO> DAOCode <T>	createDAOCode(java.lang.String value)	DAOCode を生成する。
DateAndTimeType	createDateAndTimeType(java.lang.String value)	DateAndTimeType を生成する。
DateType	createDateType(java.lang.String value)	DateType を生成する。
DecimalType	createDecimalType(java.lang.String value)	DecimalType を生成する。
<T extends DTO> DTOCode <T>	createDTOCode(java.lang.String value)	DTOCode を生成する。
IntegerType	createIntegerType(java.lang.String value)	IntegerType を生成する。
<T extends DTO> QueryCode <T>	createQueryCode(java.lang.String value)	QueryCode を生成する。
StringCode	createStringCode(java.lang.String value)	StringCode を生成する。
StringType	createStringType(java.lang.String value)	StringType を生成する。
TimestampType	createTimestampType(java.lang.String value)	TimestampType を生成する。
TimeType	createTimeType(java.lang.String value)	TimeType を生成する。

表4. 1. 1-23 ConstantFactory

クラス	ConstantFactory	
説明	Finaluna型を生成する為のファクトリ	
戻り値	メソッド	説明
static <REQ extends DTO, RES extends DTO> BLogicCode <REQ, RES>	createBLogicCode(java. lang. String value)	BLogicCode を生成する。
static <T extends DAO> DAOCCode <T>	createDAOCCode(java. lang. String value)	DAOCCode を生成する。
static DateAndTimeType	createDateAndTimeType(java. lang. String value)	DateAndTimeType を生成する。
static DateTy	createDateType(java. lang. String value)	DateType を生成する。
static DecimalType	createDecimalType(java. lang. String value)	DecimalType を生成する。
static <T extends DTO> DTOCode <T>	createDTOCode(java. lang. String value)	DTOCode を生成する。
static IntegerType	createIntegerType(java. lang. String value)	IntegerType を生成する。
static <T extends DTO> QueryCode <T>	createQueryCode(java. lang. String value)	QueryCode を生成する。
static StringCode	createStringCode(java. lang. String value)	StringCode を生成する。
static StringType	createStringType(java. lang. String value)	StringType を生成する。
static TimestampType	createTimestampType(java. lang. String value)	TimestampType を生成する。
static TimeType	createTimeType(java. lang. String value)	TimeType を生成する。
static void	setTypeFactory(ConstantTypeFactory factory)	Finaluna 型生成クラスを設定する。

表4. 1. 1-24 CollectionFactoryDelegate

クラス	CollectionFactoryDelegate	
説明	Finaluna型を生成する為のファクトリインタフェース(ListType, MapType, SetTypeに用いる)	
戻り値	メソッド	説明
<E> ListType <E>	createListType()	ListType の取得。 初期容量 10 の総称型リスト型を生成する。
<E> ListType <E>	createListType(int initialCapacity)	ListType の取得。 指定した初期容量の総称型リスト型を生成する。 初期容量の値が負の場合、FinalunaSysException をスローする。
<K, V> MapType <K, V>	createMapType()	MapType の取得。 初期容量 16 の総称型マップ型を生成する。
<K, V> MapType <K, V>	createMapType(int initialCapacity)	MapType の取得。 指定した初期容量の総称型マップ型を生成する。 初期容量の値が負の場合、FinalunaSysException をスローする。
<E> SetType <E>	createSetType()	SetType の取得。 初期容量 16 の総称型のセット型を生成する。
<E> SetType <E>	createSetType(int initialCapacity)	SetType の取得。 指定した初期容量の総称型セット型を生成する。 初期容量の値が負の場合、FinalunaSysException をスローする。

表4.1.1-25 CollectionFactory

クラス	CollectionFactory	
説明	Finaluna型を生成する為のファクトリ (ListType, MapType, SetTypeに用いる)	
戻り値	メソッド	説明
static <E> ListType <E>	createListType()	ListType の取得。初期容量 10 の総称型リスト型を生成する。
static <E> ListType <E>	createListType(int initialCapacity)	指定した初期容量の総称型リスト型を生成する。 初期容量の値が負の場合、FinalunaSysException をスローする。
static <K, V> MapType <K, V>	createMapType()	MapType の取得。初期容量 16 の総称型マップ型を生成する。
static <K, V> MapType <K, V>	createMapType(int initialCapacity)	指定した初期容量の総称型マップ型を生成する。 初期容量の値が負の場合、FinalunaSysException をスローする。
static <E> SetType <E>	createSetType()	SetType の取得。初期容量 16 の総称型セット型を生成する。
static <E> SetType <E>	createSetType(int initialCapacity)	指定した初期容量の総称型セット型を生成する。 初期容量の値が負の場合、FinalunaSysException をスローする。
static void	setTypeFactory(CollectionFactoryDelegate factory)	Finaluna 型生成クラスを設定する。

4.1.2. FINALUNA型ユーティリティ (jp.finaluna.api.type.util)

表4.1.2-1 BlankStringUtilDelegate

クラス	BlankStringUtilDelegate	
説明	空文字列、空白文字列を判定するためのユーティリティ	
戻り値	メソッド	説明
boolean	isBlank(CharSequenceType ct)	文字列が空白文字、または空文字、または null であるかチェックする。
boolean	isEmpty(CharSequenceType ct)	文字列が null、または空(長さが 0)であるかチェックする。
boolean	isNotBlank(CharSequenceType ct)	文字列が空白文字でない、かつ空文字でない、かつ null でないかどうかチェックする。
boolean	isNotEmpty(CharSequenceType ct)	文字列が null でなく、空(長さが 0)でないかチェックする。

表4.1.2-2 ConvertUtilDelegate

クラス	ConvertUtilDelegate	
説明	java⇄FINALUNA型相互変換ユーティリティインタフェース	
戻り値	メソッド	説明
int	intValue(DecimalType target)	DecimalType を int に変換する。
int	intValue(IntegerType target)	IntegerType を int に変換する。
int	intValueExact(DecimalType target)	DecimalType を int に変換する。
int	intValueExact(IntegerType target)	IntegerType を int に変換する。
CodeType	toCodeType(java.lang.String target)	String を CodeType に変換する。
DateAndTimeType	toDateAndTimeType(java.lang.String target)	String を DateAndTimeType に変換する。
DateType	toDateType(java.lang.String target)	String を DateType に変換する。
DecimalType	toDecimalType(int target)	int を DecimalType に変換する。
DecimalType	toDecimalType(java.lang.String target)	String を DecimalType に変換する。
DecimalType	toDecimalTypeExact(java.lang.String target)	String を DecimalType に変換する。
IntegerType	toIntegerType(int target)	int を IntegerType に変換する。
IntegerType	toIntegerType(java.lang.String target)	String を IntegerType に変換する。
IntegerType	toIntegerTypeExact(java.lang.String target)	String を IntegerType に変換する。
KeyCodeType	toKeyCodeType(java.lang.String target)	String を KeyCodeType に変換する。
java.lang.String	toString(CodeType target)	CodeType を String に変換する。
java.lang.String	toString(DecimalType target)	DecimalType を String に変換する。
java.lang.String	toString(IntegerType target)	IntegerType を String に変換する。
java.lang.String	toString(KeyCodeType target)	KeyCodeType を String に変換する。

java.lang.String	toString(StringCode target)	StringCode を String に変換する。
java.lang.String	toString(StringType target)	StringType を String に変換する。
StringCode	toStringCode(java.lang.String target)	String を StringCode に変換する。
StringType	toStringType(java.lang.String target)	String を StringType に変換する。
TimestampType	toTimestampType(java.lang.String target)	String を TimestampType に変換する。
TimeType	toTimeType(java.lang.String target)	String を TimeType に変換する。

表4.1.2-3 DateFormatUtil

クラス	DateFormatUtil	
説明	日付文字列の出力、解析を行う。	
戻り値	メソッド	説明
StringCode	format(DateAndTimeType dateTime, StringCode pattern)	指定されたパターンでDateAndTimeTypeを変換した日時文字列をStringCodeで取得する。
StringCode	Format(DateType date, StringCode pattern)	指定されたパターンでDateTypeを変換した日付文字列をStringCodeで取得する。
StringCode	Format(TimestampType timestamp, StringCode pattern)	指定されたパターンでTimestampTypeを変換したタイムスタンプ文字列をStringCodeで取得する。
StringCode	Format(TimeType time, StringCode pattern)	指定されたパターンでTimeTypeを変換した時刻文字列をStringCodeで取得する。
boolean	isDateString(StringCode dateString, StringCode pattern)	指定された日付文字列が、カレンダーに存在する日付かを判定する。
DateAndTimeType	toDateAndTimeType(StringCode dateAndTimeString, StringCode pattern)	指定されたパターンで日時文字列を変換し、DateAndTimeTypeで取得する。
DateType	toDateType(StringCode dateString, StringCode pattern)	指定されたパターンで日付文字列を変換し、DateTypeで取得する。
TimestampType	toTimestampType(StringCode timestampString, StringCode pattern)	指定されたパターンでタイムスタンプ文字列を変換し、TimestampTypeで取得する。
timeType	toTimeType(StrinCode timeString, StringCode pattern)	指定されたパターンで時刻文字列を変換し、TimeTypeで取得する。

表4.1.2-4 DateUtil

クラス	DateUtil	
説明	日付ユーティリティ	
戻り値	メソッド	説明
static class	DateUtil.Calendar	日付フィールド(列挙型)。
static class	DateUtil.DayBoundary	日付計算 端入コード(列挙型)。
static class	DateUtil.DayCountFraction	日付計算方法(列挙型)。
static class	DateUtil.DayOfWeek	曜日(列挙型)。
static class	DateUtil.DayOfWeekInMonth	月内曜日序数(列挙型)。
DateAndTimeType	getDateTime(DateType date, TimeType time)	日付時刻型取得 日付型と時刻型を用いて日付時刻型を作成する。
IntegerType	getDays(DateType start, DateType end, DateUtil.DayBoundary boundary)	期間計算(日数) 指定した日付間の実日数を返す。
IntegerType	getDays(DateType start, DateType end,	期間計算(日数) 指定した日付間の日数を返す。

	DateUtil.DayBoundary boundary, DateUtil.DayCountFraction fraction)	
DateType	getNthDayByWeekday(DateType baseDate, DateUtil.DayOfWeekInMonth inMonth, DateUtil.DayOfWeek week)	第 n、Y 曜日取得 基準日の月の第 n、Y 曜日に対応する日付を取得する。
DecimalType	getYears(DateType start, DateType end, DateUtil.DayBoundary boundary, DateUtil.DayCountFraction fraction)	期間計算(年数) 指定した日付の期間(年数)を返す。

表4.1.2-5 FinalunaTypeConvertUtilDelegate

クラス	FinalunaTypeConvertUtilDelegate	
説明	FINALUNA型間の相互変換を行うための型変換ユーティリティインタフェース。	
戻り値	メソッド	説明
DateAndTimeType	toDateAndTimeType(DateType target)	DateType を DateAndTimeType に変換する。
DateAndTimeType	toDateAndTimeType(StringCode target)	StringCode を DateAndTimeType に変換する。
DateAndTimeType	toDateAndTimeType(TimestampType target)	TimestampType を DateAndTimeType に変換する。
DateType	toDateType(DateAndTimeType target)	DateAndTimeType を DateType に変換する。
DateType	toDateType(StringCode target)	StringCode を DateType に変換する。
DateType	toDateType(TimestampType target)	TimestampType を DateType に変換する。
DecimalType	toDecimalType(IntegerType target)	IntegerType を DecimalType に変換する。
DecimalType	toDecimalType(StringCode target)	StringCode を DecimalType に変換する。
DecimalType	toDecimalTypeExact(StringCode target)	StringCode を DecimalType に変換する。
IntegerType	toIntegerType(StringCode target)	StringCode を IntegerType に変換する。
IntegerType	toIntegerTypeExact(StringCode target)	StringCode を IntegerType に変換する。
TimestampType	toTimestampType(DateAndTimeType target)	DateAndTimeType を TimestampType に変換する。
TimestampType	toTimestampType(DateType target)	DateType を TimestampType に変換する。
TimestampType	toTimestampType(StringCode target)	StringCode を TimestampType に変換する。
TimeType	toTimeType(DateAndTimeType target)	DateAndTimeType を TimeType に変換する。
TimeType	toTimeType(StringCode target)	StringCode を TimeType に変換する。
TimeType	toTimeType(TimestampType target)	TimestampType を TimeType に変換する。

表4.1.2-6 MathUtil

クラス	MathUtil	
説明	数値系のユーティリティ	
戻り値	メソッド	説明

static class	MathUtil.Boundary	境界値区分(列挙型)。
static class	MathUtil.InRange	範囲区分(列挙型)。
static class	MathUtil.Round	丸めモード区分(列挙型)。
DecimalType	averageDecimal(ListType<DecimalType> values, int scale)	DecimalTypeの平均を算出する。
DecimalType	averageDecimal(ListType<DecimalType> values, int scale, MathUtil.Pound round)	DecimalTypeの平均を算出する。
DecimalType	averageInteger(ListType<IntegerType> values, int scale)	IntegerTypeの平均を算出する。
DecimalType	averageInteger(ListType<IntegerType> values, int scale, MathUtil.Round round)	IntegerTypeの平均を算出する。
DecimalType	sumDecimal(ListType<DecimalType> values)	DecimalTypeを合計する。
IntegerType	sumInteger(ListType<IntegerType> values)	IntegerTypeを合計する。

表4.1.2-7 MessageUtil

クラス	MessageUtil	
説明	メッセージ系のユーティリティ	
戻り値	メソッド	説明
StringCode	getMessage(StringCode messageCode, java.lang.Object... args)	メッセージの取得を行う。

表4.1.2-8 NumberFormatUtil

クラス	NumberFormatUtil	
説明	数値を文字列に変換し整形する。	
戻り値	メソッド	説明
StringCode	format(DecimalType number, StringCode pattern)	受け取った DecimalType を数値フォーマットで指定したフォーマットの文字列表現で返す。
StringCode	format(DecimalType number, StringCode pattern, MathUtil.Round roundingMode)	受け取った DecimalType を数値フォーマットで指定したフォーマットの文字列表現で返す。
StringCode	format(IntegerType integer, StringCode pattern)	受け取った IntegerType を数値フォーマットで指定したフォーマットの文字列表現で返す。
DecimalType	parseDecimalType(StringCode numberString, StringCode pattern)	受け取った IntegerType を数値フォーマットで指定したフォーマットの文字列表現で返す。
IntegerType	parseIntegerType(StringCode integerString, StringCode pattern)	受け取ったStringCodeをIntegerTypeにparseして返す。
StringCode	toPlainStringCode(DecimalType number)	指数フィールドなしで、受け取ったDecimalTypeの文字列表現を返す。

表4.1.2-9 NumerationSystemUtil

クラス	NumerationSystemUtil	
説明	命数法を用いた数値表現の文字列を、命数法を用いない数値表現の文字列に置き換える。	
戻り値	メソッド	説明
StringCode	replace(StringCode target)	文字列内の命数法を用いた数値表現を、命数法を用いない数値表現の文字列に置き換えて返す。
StringCode	toPlainNumberString(StringCode target)	命数法を用いた数値表現の文字列を、命数法を用いない数値表現の文字列に置き換えて返す。

表4.1.2-10 OutPrinterUtil

クラス	OutPrinterUtil	
説明	帳票出力ユーティリティ	
戻り値	メソッド	説明
void	toPrinter(DTO dto)	プリンター印刷をする。

表4.1.2-11 PropertyUtil

クラス	PropertyUtil	
説明	プロパティの操作を行うユーティリティ	
戻り値	メソッド	説明
StringCode	getProperty(StringCode key)	指定されたキーのプロパティを取得する。
StringCode	getProperty(StringCode key, StringCode defaultValue)	指定されたキーのプロパティを取得する。存在しない場合はデフォルト値を返す。

表4.1.2-12 TimestampUtil

クラス	TimestampUtil	
説明	現在タイムスタンプ取得ユーティリティ	
戻り値	メソッド	説明
TimestampType	getCurrentTimestamp()	現在タイムスタンプを取得する。

表4.1.2-13 BlankStringUtil

クラス	BlankStringUtil	
説明	空文字列、空白文字列を判定するためのユーティリティクラス	
戻り値	メソッド	説明
boolean	isBlank(CharSequenceType ct)	文字列が空白文字、または空文字、または null であるかチェックする。
boolean	isEmpty(CharSequenceType ct)	文字列が null、または空(長さが 0)であるかチェックする。
boolean	isNotBlank(CharSequenceType ct)	文字列が空白文字でない、かつ空文字でない、かつ null でないかどうかチェックする。
boolean	isNotEmpty(CharSequenceType ct)	文字列が null でなく、空(長さが 0)でないかチェックする。
void	setStringUtilDelegate(BlankStringUtilDelegate)	空文字列、空白文字列を行うユーティリティクラスを設定する。

	stringUtilDelegate)	
--	---------------------	--

表4. 1.2-14 ConvertUtil

クラス	ConvertUtil	
説明	java⇔FINALUNA型相互変換ユーティリティクラス	
戻り値	メソッド	説明
int	intValue(DecimalType target)	DecimalType を int に変換する。
int	intValue(IntegerType target)	IntegerType を int に変換する
int	intValue Exact(DecimalType target)	DecimalType を int に変換する。
int	intValue Exact(IntegerType target)	IntegerType を int に変換する。
void	setConvertUtil(ConvertUtilDelegate convertUtilDelegate)	java⇔FINALUNA 型相互変換クラスを設定する。
CodeType	toCodeType(java.lang.String target)	String を CodeType に変換する。
DateAndTimeType	toDateAndTimeType(java.lang.String target)	String を DateAndTimeType に変換する。
DateType	toDateType(java.lang.String target)	String を DateType に変換する。
DecimalType	toDecimalType(int target)	int を DecimalType に変換する。
DecimalType	toDecimalType(java.lang.String target)	String を DecimalType に変換する。
DecimalType	toDecimalTypeExact(java.lang.String target)	String を DecimalType に変換する。
IntegerType	toIntegerType(int target)	int を IntegerType に変換する。
IntegerType	toIntegerType(java.lang.String target)	String を IntegerType に変換する。
IntegerType	toIntegerTypeExact(java.lang.String target)	String を IntegerType に変換する。
KeyCodeType	toKeyCodeType(java.lang.String target)	String を KeyCodeType に変換する。
java.lang.String	toString(CodeType target)	CodeType を String に変換する。
java.lang.String	toString(DecimalType target)	DecimalType を String に変換する。
java.lang.String	toString(IntegerType target)	IntegerType を String に変換する。
java.lang.String	toString(KeyCodeType target)	KeyCodeType を String に変換する。
java.lang.String	toString(StringCode target)	StringCode を String に変換する。
java.lang.String	toString(StringType target)	StringType を String に変換する。
StringCode	toStringCode(java.lang.String target)	String を StringCode に変換する。
StringType	toStringType(java.lang.String target)	String を StringType に変換する。
TimestampType	toTimestampType(java.lang.String target)	String を TimestampType に変換する。
TimeType	toTimeType(java.lang.String target)	String を TimeType に変換する。

表4. 1.2-15 FinalunaTypeConvertUtil

クラス	FinalunaTypeConvertUtil	
説明	FINALUNA型間の相互変換を行うための型変換ユーティリティクラス	
戻り値	メソッド	説明

void	setFinalunaTypeConvertUtil (FinalunaTypeConvertUtilDelegate convertUtilDelegate)	変換処理実装インスタンスを設定する。
DateAndTimeType	toDateAndTimeType (DateType target)	DateType を DateAndTimeType に変換する。
DateAndTimeType	toDateAndTimeType (StringCode target)	StringCode を DateAndTimeType に変換する。
DateAndTimeType	toDateAndTimeType (TimestampType target)	TimestampType を DateAndTimeType に変換する。
DateType	toDateType (DateAndTimeType target)	DateAndTimeType を DateType に変換する。
DateType	toDateType (StringCode target)	StringCode を DateType に変換する。
DateType	toDateType (TimeStampType target)	TimestampType を DateType に変換する。
DecimalType	toDecimalType (IntegerType target)	IntegerType を DecimalType に変換する。
DecimalType	toDecimalType (StringCode target)	StringCode を DecimalType に変換する。
DecimalType	toDecimalTypeExact (StringCode target)	StringCode を DecimalType に変換する。
IntegerType	toIntegerType (StringCode target)	StringCode を IntegerType に変換する。
IntegerType	toIntegerTypeExact (StringCode target)	StringCode を IntegerType に変換する。
TimestampType	toTimestampType (DateAndTimeType target)	DateAndTimeType を TimestampType に変換する。
TimestampType	toTimestampType (DateType target)	DateType を TimestampType に変換する。
TimestampType	toTimestampType (StringCode target)	StringCode を TimestampType に変換する。
TimeType	toTimeType (DateAndTimeType target)	DateAndTimeType を TimeType に変換する。
TimeType	toTimeType (StringCode target)	StringCode を TimeType に変換する。
TimeType	toTimeType (TimestampType target)	TimestampType を TimeType に変換する。

4.1.3. FINALUNA型入力チェック (jp.finaluna.api.type.util.validator)

表4.1.3-1 DateValidator

クラス	DateValidator	
説明	DateType型の入力チェックを行う機能	
戻り値	メソッド	説明
boolean	isAfter(DateType value, DateType afterDate)	日付が指定した日付より後にあるかチェックする。
boolean	isBefore(DateType value, DateType beforeDate)	日付が指定した日付より前にあるかチェックする。
boolean	isInRange(DateType value, DateType min, DateType max, MathUtil.InRange range)	DateType 型の範囲チェックを行う。
boolean	isOnAndAfter(DateType value, DateType afterDate)	日付が指定した日付より後にあるかチェックする。
boolean	isOnAndBefore(DateType value, DateType beforeDate)	日付が指定した日付より前にあるかチェックする。

表4.1.3-2 NumericValidator

クラス	NumericValidator	
説明	IntegerType型及びDecimalType型の入力チェックを行う機能	
戻り値	メソッド	説明
boolean	isInRange(DecimalType value, DecimalType min, DecimalType max, MathUtil.InRange range)	DecimalType 型の範囲チェックを行う。
boolean	isInRange(IntegerType value, IntegerType min, IntegerType max, MathUtil.InRange range)	IntegerType 型の範囲チェックを行う。
boolean	maxLength(DecimalType value, int integerMax, int decimalMax, MathUtil.Boundary boundary)	DecimalType 型の最大桁数チェックを行う。
boolean	maxLength(DecimalType value, int integerMax, MathUtil.Boundary boundary)	DecimalType 型の最大桁数チェックを行う。
boolean	maxLength(IntegerType value, int max, MathUtil.Boundary boundary)	IntegerType 型の最大桁数チェックを行う。
boolean	minLength(DecimalType value, int integerMin, int decimalMin, MathUtil.Boundary boundary)	DecimalType 型の最小桁数チェックを行う。
boolean	minLength(IntegerType value, int min, MathUtil.Boundary boundary)	IntegerType 型の最小桁数チェックを行う。

表4.1.3-3 StringValidator

クラス	StringValidator	
説明	StringType型及びStringCode型の入力チェックを行う機能	
戻り値	メソッド	説明
boolean	isBlankOrNull (StringCode value)	空白・NULLチェックを行う。
boolean	isBlankOrNull (StringType value)	空白・NULLチェックを行う。
boolean	isDecimal (StringCode value)	小数チェックを行う。
boolean	isDecimal (StringType value)	小数チェックを行う。
boolean	isInteger (StringCode value)	数値チェックを行う。
boolean	isInteger (StringType value)	数値チェックを行う。
boolean	matchRegexp (StringCode value, StringCode regexp)	正規表現チェックを行う。
boolean	matchRegexp (StringType value, StringType regexp)	正規表現チェックを行う。
boolean	maxLength (StringCode value, int max, MathUtil.Boundary boundary)	StringCode 型の最大桁数チェックを行う。
boolean	maxLength (StringType value, int max, MathUtil.Boundary boundary)	StringType 型の最大桁数チェックを行う。
boolean	minLength (StringCode value, int min, MathUtil.Boundary boundary)	StringCode 型の最小桁数チェックを行う。
boolean	minLength (StringType value, int min, MathUtil.Boundary boundary)	StringType 型の最小桁数チェックを行う。

表4.1.3-4 TimestampValidator

クラス	TimestampValidator	
説明	TimestampType型の入力チェックを行う機能	
戻り値	メソッド	説明
boolean	isAfter (TimestampType value, TimestampType afterTime)	日付時刻が指定した日付時刻より後にあるかチェックする。
boolean	isBefore (TimestampType value, TimestampType beforeTime)	日付時刻が指定した日付時刻より前にあるかチェックする。
boolean	isInRange (TimestampType value, TimestampType min, TimestampType max, MathUtil.InRange range)	TimestampType 型の範囲チェックを行う。
boolean	isOnAndAfter (TimestampType value, TimestampType afterTime)	日付時刻が指定した日付時刻より後にあるかチェックする。
boolean	isOnAndBefore (TimestampType value, TimestampType beforeTime)	日付時刻が指定した日付時刻より前にあるかチェックする。

表4.1.3-5 TimeValidator

クラス	TimeValidator	
説明	TimeType型の入力チェックを行う機能	
戻り値	メソッド	説明
boolean	isAfter(TimeType value, TimeType afterTime)	時刻が指定した時刻より後にあるかチェックする。
boolean	isBefore(TimeType value, TimeType beforeTime)	時刻が指定した時刻より前にあるかチェックする。
boolean	isInRange(TimeType value, TimeType min, TimeType max, MathUtil.InRange range)	TimeType 型の範囲チェックを行う。
boolean	isOnAndAfter(TimeType value, TimeType afterTime)	時刻が指定した時刻より後にあるかチェックする。
boolean	isOnAndBefore(TimeType value, TimeType beforeTime)	時刻が指定した時刻より前にあるかチェックする。

表4.1.3-6 DateAndTimeValidator

クラス	DateAndTimeValidator	
説明	DateAndTimeType型の入力チェックを行う機能	
戻り値	メソッド	説明
boolean	isAfter(DateAndTimeType value, DateAndTimeType afterTime)	日付時刻が指定した日付時刻より後にあるかチェックする。
boolean	isBefore(DateAndTimeType value, DateAndTimeType beforeTime)	日付時刻が指定した日付時刻より前にあるかチェックする。
boolean	isInRange(DateAndTimeType value, DateAndTimeType min, DateAndTimeType max, MathUtil.InRange range)	DateAndTimeType 型の範囲チェックを行う。
boolean	isOnAndAfter(DateAndTimeType value, DateAndTimeType afterTime)	日付時刻が指定した日付時刻より後にあるかチェックする。
boolean	isOnAndBefore(DateAndTimeType value, DateAndTimeType beforeTime)	日付時刻が指定した日付時刻より前にあるかチェックする。

4.1.4. 電文I/O (jp.finaluna.api.dto)

表4.1.4-1 DTO

クラス	DTO	
説明	業務ロジックの入出力値を保持するデータ転送オブジェクトのマーカージャインタフェース	
戻り値	メソッド	説明
	なし	

表4.1.4-2 RequestDTO

クラス	RequestDTO	
説明	入力情報を格納するデータ転送オブジェクト	
戻り値	メソッド	説明
Java.lang.Object	get(java.lang.String name)	指定した項目名の Finaluna 型抽象クラスを取得する。

表4.1.4-3 ResponseDTO

クラス	ResponseDTO	
説明	出力情報を格納するデータ転送オブジェクト	
戻り値	メソッド	説明
	なし	

表4.1.4-4 ParamaterDTO

クラス	ParameterDTO	
説明	入出力情報を格納するデータ転送オブジェクト。	
戻り値	メソッド	説明
void	clear()	データ転送オブジェクト内に保持している項目オブジェクトを全て削除する。
java.lang.Object []	getValues()	データ転送オブジェクト内に保持している全項目オブジェクトを配列として戻す。
void	remove(java.lang.String name)	データ転送オブジェクトから指定した識別名を持つ項目オブジェクトを削除する。
void	set(java.lang.String name, java.lang.Object value)	データ転送オブジェクトに項目オブジェクトを追加する。
void	setAll(DTO dto)	他のデータ転送オブジェクトの値を全て追加する。

表4.1.4-5 ResponseStatusHolder

クラス	ResponseStatusHolder	
説明	処理結果ステータスを保持するインタフェース	
戻り値	メソッド	説明
ResponseStatusCode	getResponseStatusCode	業務ロジックの処理結果ステータスを保持する。
void	setResponseStatusCode(ResponseStatusCode responseStatus)	業務ロジックの処理結果ステータスを設定する。

表4.1.4-6 MessageHolder

クラス	MessageHolder	
説明	メッセージを保持するインタフェース	
戻り値	メソッド	説明
void	addErrorMessage(String field, Message message)	指定フィールドにエラーメッセージを追加する。
void	addMessage(String field, Message message)	指定フィールドにメッセージを追加する。
Messages	getErrorMessages()	複数のエラーメッセージを取得する。
Messages	getMessages()	複数のメッセージを取得する。

表4.1.4-7 ResponseStatusCode

クラス	ResponseStatusCode	
説明	応答ステータスを一意に識別するコード	
戻り値	メソッド	説明
	なし	

4.1.5. メッセージ (jp.finaluna.api.dto.message)

表4.1.5-1 Message

クラス	Message	
説明	メッセージを保持するインタフェース	
戻り値	メソッド	説明
	なし	

表4.1.5-2 Messages

クラス	Messages	
説明	メッセージを複数持つメッセージクラスのインタフェース	
戻り値	メソッド	説明
void	addMessage(String field, Message message)	指定された field に、message を追加する。

4.1.6. DB、ファイル I/O 、共有変数(jp.finaluna.api.dao)

表4.1.6-1 DAO

クラス	DAO	
説明	外部データへアクセスするためのデータアクセスオブジェクトのマーカージャインタフェース	
戻り値	メソッド	説明
	なし	

表4.1.6-2 QueryDAO

クラス	QueryDAO	
説明	業務ロジックより外部リソースを取得するための検索用データアクセスオブジェクト	
戻り値	メソッド	説明
< P extends DTO , R extends DTO > ListType <R>	getInputResource(P dto, QueryCode <R> id)	検索処理を実行する。
< R extends DTO > ListType <R>	getInputResource(QueryCode <R> id)	検索処理を実行する。

表4.1.6-3 UpdateDAO

クラス	UpdateDAO	
説明	業務ロジックより外部リソースを更新するためのデータアクセスオブジェクト	
戻り値	メソッド	説明
int	update(QueryCode id)	更新処理を実行する(1件の実行)。
< P extends DTO > int	update(P dto, QueryCode <P>id)	更新処理を実行する(複数件の実行)。戻り値は総更新件数を返却する。
< PA extends ListType<P>, p extends DTO> int	update(PA dtoList, QueryCode<P> id)	更新処理を実行する(1件の実行)。

表4.1.6-4 PageQueryDAO

クラス	PageQueryDAO	
説明	業務ロジックより外部リソースを取得するための一覧検索用データアクセスオブジェクト	
戻り値	メソッド	説明
< P extends DTO , R extends DTO > ListType<R>	getInputResource(P dto, QueryCode <R> id, int start, int count)	指定ページ検索処理を実行する。

表4.1.6-5 CursorDAO

クラス	CursorDAO	
説明	業務ロジックより外部リソースをカーソルで取得するための検索用データアクセスオブジェクト	
戻り値	メソッド	説明
< P extends DTO , R extends DTO > CursorSet<R>	getInputResource(P dto, QueryCode <R> id)	データベースの検索処理を行う。
< R extends DTO > CursorSet<R>	getInputResource(QueryCode<R> id)	データベースの検索処理を行う。
void	close	カーソル用 DAO が保持しているデータベース関連リソースをクローズする。

表4.1.6-6 StoredProcedureDAO

クラス	StoredProcedureDAO	
説明	業務ロジックよりストアードプロシージャにより外部リソースを取得、更新するためのデータアクセスオブジェクト	
戻り値	メソッド	説明
< P extends DTO > P	execute(P dto, QueryCode <P> id)	ストアードプロシージャを呼び出して戻り値を DTO に変換して返却する。

表4.1.6-7 EntityQueryDAO

クラス	EntityQueryDAO	
説明	外部データを主キー参照する為のデータアクセスオブジェクト	
戻り値	メソッド	説明
VALUE	get(PK pk)	主キーを条件にして、1レコードを格納した DTO を取得する。

表4.1.6-8 EntityUpdateDAO

クラス	EntityUpdateDAO	
説明	外部データに主キー更新する為のデータアクセスオブジェクト。	
戻り値	メソッド	説明
void	delete(VALUE value)	1レコードを削除する。
void	forceDelete(PK pk)	1レコードを強制削除する。
void	forceUpdate(VALUE value)	1レコードを強制更新する。
VALUE	getForUpdate(VALUE value)	1レコードを更新用取得する。
void	insert(VALUE value)	1レコードを挿入する。
void	update(VALUE from, VALUE to)	1レコードを更新する。

表4.1.6-9 SqlPreCheck

クラス	SqlPreCheck	
説明	SQL実行前チェック	
戻り値	メソッド	説明
void	check(ProcessContext context, DTO dto)	SQL実行前チェック。

表4.1.6-10 QueryCode

クラス	QueryCode	
説明	クエリコード情報を保持する型のコード型	
戻り値	メソッド	説明
	なし	

表4.1.6-11 FinalunaOptimisticLockFailureException

クラス	FinalunaOptimisticLockFailureException	
説明	参照した時点から更新時まで、他者によってレコードが更新されていた場合にスローされる例外。	
戻り値	メソッド	説明
	なし	

4.1.7. 業務処理 (jp.finaluna.api.blogic)

表4.1.7-1 Logic

クラス	Logic< CONTEXT extends Context, REQ extends DTO, RES extends DTO>	
説明	業務ロジックが実装するインタフェース	
戻り値	メソッド	説明
void	execute(CONTEXT context, REQ requestDto, RES responseDto)	業務ロジックを実行する。

表4.1.7-2 Context

クラス	Context	
説明	業務ロジックに必要な情報を保持するコンテキストインタフェース	
戻り値	メソッド	説明
< D extends ResponseDTO > D	createResponseDTO(DTOCode < D > key)	ResponseDTOの取得。
< D extends ParameterDTO > D	createParameterDTO(DTOCode < D > key)	ParameterDTOの取得。
< Q extends QueryDAO > Q	createQueryDAO(DAOCCode < Q > key)	QueryDAOの取得。
< U extends UpdateDAO > U	createUpdateDAO(DAOCCode < U > key)	UpdateDAOの取得。
< P extends PageQueryDAO > P	createPageQueryDAO(DAOCCode < P > key)	PageQueryDAOの取得。
< C extends CursorDAO > C	createCursorDAO(DAOCCode < C > key)	CursorDAOの取得。
Message	createMessage(String messageKey, Object messageArgs)	メッセージの取得。
ResponseStatusCode	createResponseStatusCode(String key)	レスポンスステータスの取得。
< REQ extends DTO, RES extends DTO > void	execute(REQ request, RES response, BLogicCode < REQ, RES > key)	業務ロジックを呼出す。
StringCode	getParameter(StringCode key)	業務ロジックへのパラメータを取得。
ListType < StringCode >	getParameterValues(StringCode key)	業務ロジックへのパラメータを取得。
< D extends DTO > D	createDTO(DTOCode < D > key)	DTOの取得。
< PK extends BaseType, VALUE extends DTO, D extends EntityQueryDAO < PK, VALUE > > D	createEntityQueryDAO(DAOCCode < D > key)	EntityQueryDAOの取得。
< PK extends BaseType, VALUE extends DTO, D extends EntityUpdateDAO < PK, VALUE > > D	createEntityUpdateDAO(DAOCCode < D > key)	EntityUpdateDAOの取得。
< S extends StoredProcedureDAO > S	createStoredProcedureDAO(DAOCCode < S > key)	StoredProcedureDAOの取得。

表4. 1. 7-3 ProcessContext

クラス	ProcessContext	
説明	業務ロジックに必要な情報を保持するコンテキストインタフェース	
戻り値	メソッド	説明
DateAndTimeType	createDateAndTimeType()	DateAndTimeType の取得。
DateAndTimeType	createDateAndTimeType(String value)	DateAndTimeType の取得。
DateType	createDateType()	DateType の取得。
DateType	createDateType(String value)	DateType の取得。
DecimalType	createDecimalType()	DecimalType の取得。
DecimalType	createDecimalType(double value)	DecimalType の取得。
DecimalType	createDecimalType(int value)	DecimalType の取得。
DecimalType	createDecimalType(long value)	DecimalType の取得。
DecimalType	createDecimalType(String value)	DecimalType の取得。
IntegerType	createIntegerType()	IntegerType の取得。
IntegerType	createIntegerType(double value)	IntegerType の取得。
IntegerType	createIntegerType(int value)	IntegerType の取得。
IntegerType	createIntegerType(long value)	IntegerType の取得。
IntegerType	createIntegerType(String value)	IntegerType の取得。
< E > ListType < E >	createListType()	ListType の取得。
< E > ListType < E >	createListType(int initialCapacity)	ListType の取得。
< E > SetType < E >	createSetType()	SetType の取得。
< E > SetType < E >	createSetType(int initialCapacity)	SetType の取得。
StringCode	createStringCode()	StringCode の取得。
StringCode	createStringCode(char value)	StringCode の取得。
StringCode	createStringCode(int value)	StringCode の取得。
StringCode	createStringCode(long value)	StringCode の取得。
StringCode	createStringCode(String value)	StringCode の取得。
StringType	createStringType()	StringType の取得。
StringType	createStringType(boolean value)	StringType の取得。
StringType	createStringType(byte value)	StringType の取得。
StringType	createStringType(char value)	StringType の取得。
StringType	createStringType(double value)	StringType の取得。
StringType	createStringType(int value)	StringType の取得。
StringType	createStringType(long value)	StringType の取得。
StringType	createStringType(String value)	StringType の取得。
TimestampType	createTimestampType()	TimestampType の取得。
TimestampType	createTimestampType(String value)	TimestampType の取得。

クラス	ProcessContext	
説明	業務ロジックに必要な情報を保持するコンテキストインタフェース	
戻り値	メソッド	説明
TimeType	createTimeTyp()	TimeType の取得。
TimeType	createTimeType(String value)	TimeType の取得。
< K, V > MapType < K, V >	createMapType()	MapType の取得。
< K, V > MapType < K, V >	createMapType(int initialCapacity)	MapType の取得。

表4.1.7-4 BLogicCode

クラス	BLogicCode < REQ extends DTO, RES extends DTO >	
説明	業務ロジックを一意に識別するコード型	
戻り値	メソッド	説明
	なし	

表4.1.7-5 DAOCODE

クラス	DAOCODE < T extends DAO >	
説明	データアクセスオブジェクト(DAO)を一意に識別するコード型	
戻り値	メソッド	説明
	なし	

表4.1.7-6 DTOCODE

クラス	DTOCODE < T extends DTO >	
説明	データ転送オブジェクト(DTO)を一意に識別するコード型	
戻り値	メソッド	説明
	なし	

表4.1.7-7 StartupLogic

クラス	StartupLogic < CONTEXT extends Context >	
説明	業務起動時処理インタフェース	
戻り値	メソッド	説明
void	start(CONTEXT context)	業務起動時処理を行う。

表4.1.7-8 StopLogic

クラス	StopLogic < CONTEXT extends Context >	
説明	業務起動時処理インタフェース	
戻り値	メソッド	説明
void	stop(CONTEXT context)	業務停止時処理を行う。

4.1.8. 出口処理 (jp.finaluna.api.exit)

表4.1.8-1 LogicPostExit

クラス	LogicPostExit	
説明	ロジック後出口インタフェース	
戻り値	メソッド	説明
Void	executePostExit(CONTEXT context, REQ requestDto, RES responseDto)	ロジック後出口を実行する

表4.1.8-1 LogicPreExit

クラス	LogicPreExit	
説明	ロジック前出口インタフェース	
戻り値	メソッド	説明
void	executePreExit(CONTEXT context, REQ requestDto, RES responseDto)	ロジック前出口を実行する

4.1.9. 例外処理 (jp.finaluna.api.exception)

表4.1.9-1 FinalunaAppException

クラス	FinalunaAppException	
説明	業務ロジックがスローする例外(業務例外)	
戻り値	メソッド	説明
java.lang.Object []	getArgs ()	埋め込み文字列を取得する。
java.lang.String	getErrorCode ()	エラーコードを取得する。
java.lang.String	getMessage ()	メッセージを取得する。
java.lang.String	getMessageCode ()	メッセージコードを取得する。
void	setMessage (String message)	メッセージを設定する。

表4.1.9-2 FinalunaSysException

クラス	FinalunaSysException	
説明	システム例外を表す非検査例外	
戻り値	メソッド	説明
java.lang.Object []	getArgs ()	埋め込み文字列を取得する。
java.lang.String	getErrorCode ()	エラーコードを取得する。
java.lang.String	getMessage ()	メッセージを取得する。
java.lang.String	getMessageCode ()	メッセージコードを取得する
void	setMessage (String message)	メッセージを設定する。

表4.1.9-3 FinalunaUnsupportedOperationException

クラス	FinalunaUnsupportedOperationException	
説明	サポートしないメソッドが呼び出された場合の例外。	
戻り値	メソッド	説明
	なし	

表4.1.9-4 ErrorMessageHolder

クラス	ErrorMessageHolder	
説明	エラーメッセージを保持するためのインタフェース	
戻り値	メソッド	説明
java.lang.Object[]	getArgs()	エラーメッセージに埋め込むパラメータ文字列を取得する。
java.lang.String	getErrorCode()	エラーコードを取得する。
java.lang.String	getMessageCode()	メッセージコードを取得する。
void	setMessage(String message)	エラーコードに対応するメッセージ文字列を設定する。

4.1.10. ロガー (jp.finaluna.api.log)

表4.1.10-1 LogFactory

クラス	LogFactory	
説明	ロガーを生成する為のファクトリ	
戻り値	メソッド	説明
static void	setLogFactory(LogFactoryDelegate factory)	ロガー生成クラスの設定。
static void	getLogger(Class<?> clazz)	ロガーインスタンス取得。
static void	init()	初期化处理。
static void	destroy()	終了処理。

表4.1.10-2 LogFactoryDelegate

クラス	LogFactoryDelegate	
説明	ロガーを生成する為のファクトリインタフェース	
戻り値	メソッド	説明
void	getLogger(Class<?> clazz)	ロガーインスタンス取得。
void	init()	初期化处理。
void	destroy()	終了処理。

表4.1.10-3 Logger

クラス	Logger	
説明	FINALUNAでログ出力機能を使用するためのインタフェース	
戻り値	メソッド	説明
void	error(String msgCode, Throwable th, Object... args)	ログ出力処理 (error)。
void	error(String msgCode, Object... args)	ログ出力処理 (error)。
void	warn(String msgCode, Throwable th, Object... args)	ログ出力処理 (warn)。
void	warn(String msgCode, Object... args)	ログ出力処理 (warn)。
void	info(String msgCode, Throwable th, Object... args)	ログ出力処理 (info)。
void	info(String msgCode, Object... args)	ログ出力処理 (info)。
void	debug(String msgCode, Throwable th, Object... args)	ログ出力処理 (debug)。
void	debug(String msgCode, Object... args)	ログ出力処理 (debug)。
void	trace(String msgCode, Throwable th, Object... args)	ログ出力処理 (trace)。
void	trace(String msgCode, Object... args)	ログ出力処理 (trace)。

void	isInfoEnabled()	ログ出力レベル判定処理 (info)。
void	isDebugEnabled()	ログ出力レベル判定処理 (debug)。
void	isTraceEnabled()	ログ出力レベル判定処理 (trace)。

4.2. 用語集

A-Z	
DAO	外部データとの接続インタフェースとなるデータアクセスオブジェクト(DataAccessObject)の略称。
DTO	業務処理ロジックの入出力データインタフェースであり、内部に FINALUNA 型のデータを保持する。DataTransferObject の略称。
FINALUNA API	FINALUNA フレームワークを適用した業務処理ロジックで使用する API。業務処理ロジック開発に必要な機能を網羅した API であり、恒久的に不変である。
FINALUNA API EX	テストケース作成時に使用する FINALUNA API の拡張 API。FINALUNA API 同様、恒久的に不変である。
FINALUNA フレームワーク	FINALUNA API の実装と、FINALUNA が提供する処理方式(オンライン、メッセージ交換、バッチ)の実装の総称。FINALUNA API (EX) に対し、Java 技術の変化等により、改変される。
FINALUNA 型	FINALUNA フレームワークを適用した業務処理ロジックで使用する型の総称。Java の標準 API をラップしており、それぞれの型は業務処理ロジック作成に必要な機能を持つ。
オンライン処理	FINALUNA フレームワークが提供する処理方式の 1 つ。Web・オンライン方式。
POJO	Plain Old Java Object の略。シンプルで、依存性をなくしたオブジェクト。実行環境やフレームワークのことは一切知らないオブジェクト。
Queue	メッセージ交換処理にて使用される入力/出力のデータ構造。
TERASOLUNA	オンライン連携フレームワーク。

UT用ドライバ	テストケース作成時に使用するユーティリティAPI。
---------	---------------------------

あーん

外部データアクセス	DAOと同意。
共通機能	FINALUNAフレームワークの各処理で共通的な機能を切り出したもの。
共有変数	異なる筐体、JVM間でデータの共用、高速のデータ参照ができる機能。データ参照はDAOで行う。値の参照のみ可能。
多重化	バッチ処理で、1処理を分割して同時並行で実行すること。
通常版バッチ	バッチ処理クライアントの一つで、一業務分の処理を実行したら処理を終了する。
ディレードバッチ (同期版)	バッチ処理クライアントからの一つで、ディレード管理テーブルからリクエスト情報を取得し、スレッド処理にて複数のバッチ処理を同時に起動する。未実行のレコードがなくなったら処理を終了する。
ディレードバッチ (非同期版)	バッチ処理クライアントからの一つで、ディレード管理テーブルからリクエスト情報を取得し、非同期でバッチ処理を起動する。外部から終了されるまで処理を実行し続ける。
テストケース	単体試験を実行するためのテスト用コード。Javaで記述する。
電文	DTOと同意。
バッチ処理	FINALUNAフレームワークが提供する処理方式の1つ。一括処理方式。
リクエストDTO	DTOのうち、特に業務ロジックの入力データを示すもの。

例外	処理中に発生した問題を通知するもの。
レスポンス DTO	DTOのうち、特に業務ロジックの出力データを示すもの。