目次

改訂情報

変更年月日	変更内容
2013-08-01	初版
2015-04-01	第2版 下記の内容を追加しました。 ・「*モジュール間の依存関係*」 ・「*モジュール情報の設定*」
2015-12-01	第3版 下記の内容を修正しました。 ・「*モジュール間の依存関係*」
2016-12-01	第4版 下記を修正しました ・DB2に関する情報を削除しました。

はじめに

本書の目的

本書では、ユーザモジュールを開発する際に有益な情報や注意点などについて説明します。

前提条件

e Builder のセットアップを行ってください。 詳細は、セットアップガイド を参照してください。

対象読者

以下の利用者を対象としています。

- モジュールの仕組みやモジュール開発をもっと深く知りたい方
- intra-mart e Builder for Accel Platform を利用して製品用のユーザモジュールを開発する方

テナント環境セットアップの組み込み

e Builder の モジュール・プロジェクト を利用して作成したアプリケーションにおいて、データベースのテーブル作成処理等を
テナント環境セットアップ機能で行えるようにする方法について説明します。

なお、テナント環境セットアップの詳細仕様については「テナント環境セットアップ 仕様書」を参照してください。

項目

テナント環境セットアップとは

テナント環境セットアップとは、モジュールやアプリケーションを新たに追加した場合に必要となるセットアップ処理の実行と、
その実行履歴を管理する機能です。
テナント環境セットアップは intra-mart Accel Platform のシステム管理者によって実行されます。
テナント環境セットアップを実行すると、その実行履歴が管理されるため、同じセットアップが２度実行されることはありません。

また運用後に、機能追加や仕様変更のため、データベーステーブルの追加作成等の処理が必要となった場合には、
追加でセットアップ処理を行うことができます。

テナント環境セットアップに組み込める処理は以下の通りです。

テナント環境セットアップに組み込み可能な処理一覧	
データベース系	DDL 実行
	DML 実行

テナント環境セットアップに組み込み可能な処理一覧	
テナントマスタ系	ロール インポート
	アカウント インポート
	カレンダー インポート
	メニュー インポート
	認可 インポート
	ジョブスケジューラ インポート
拡張処理	Javaプログラム実行
	サーバサイドJavaScriptプログラム実行

テナント環境セットアップの組み込みと実行の方法

下記手順の通り intra-mart Accel Platform をセットアップすることで、作成した「セットアップ設定ファイル」を
テナント環境セットアップに組み込んで実行することができます。

1. *セットアップ設定ファイルとインポートファイルの作成*
2. *ユーザモジュールの作成*
3. *ユーザモジュールを組み込んだwarファイルの出力*
4. *warファイルのデプロイと intra-mart Accel Platform の起動*
5. *テナント環境セットアップの実行*

それぞれ順に説明します。

セットアップ設定ファイルとインポートファイルの作成

はじめに e Builder の モジュール・プロジェクト 内に「セットアップ設定ファイル」を下記ルールに従って作成します。

セットアップ設定ファイルの定義ルール	
配置ディレクトリ	src/main/conf/products/import/basic/{アーティファクトID（プロジェクト名）}/
	例：src/main/conf/products/import/basic/my_project/
ファイル名	import-{アーティファクトID（プロジェクト名）}-config-1.xml
	例：import-my_project-config-1.xml

- ファイル名のサフィックスの “1” は、各モジュールにおけるテナント環境セットアップのバー

ジョン管理のための番号（ **スキーマバージョン**）です。
初回は必ず “1” とします。

- ファイルの内容については次節を参照してください。

コラム

スキーマバージョンについて、当ドキュメントでは詳細説明は省きます。

詳細仕様については「テナント環境セットアップ 仕様書」を参照してください。

注意

上述のセットアップ設定ファイルの定義ルールは、必ず守ってください。

セットアップ設定ファイルの配置ディレクトリやファイル名が綴り誤りなどにより
定義ルールに則っていない場合、テナント環境セットアップの対象となりません。
その場合の実行時には例外も発生しないため、注意してください。

セットアップ設定ファイルの定義内容

セットアップ設定ファイルの定義内容について説明します。

基本構成

セットアップ設定ファイルはXMLで記述します。その基本構造は以下の通りです。

```
<?xml version="1.0" encoding="UTF-8"?>
<import-data-config
  xmlns="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://intra_mart.co.jp/system/service/provider/importer/config/import-
data-config import-data-config.xsd ">

    <!-- データベース系 ※不要な場合、削除してください。 -->
    <database>
      : (省略)
    </database>

    <!-- テナントマスタ系 ※不要な場合、削除してください。 -->
    <tenant-master>
      : (省略)
    </tenant-master>

    <!-- 拡張処理 ※不要な場合、削除してください。 -->
    <extends-import>
      : (省略)
    </extends-import>

</import-data-config>
```

コラム

セットアップ設定ファイルのフォーマットは、Webアーカイブディレクトリ内の下記フォーマットファイル(xsdファイル)で定義されています。

- WEB-INF/schema/import-data-config.xsd

コラム

セットアップ設定ファイルの完成形の例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<import-data-config
  xmlns="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://intra_mart.co.jp/system/service/provider/importer/config data-config import-data-config.xsd ">
    <database>
        <create-file>products/import/basic/my_project/my_project-ddl.sql</create-file>
        <insert-file>products/import/basic/my_project/my_project-dml.sql</insert-file>
    </database>
    <tenant-master>
        <role-file>products/import/basic/my_project/my_project-role.xml</role-file>
        <account-file>products/import/basic/my_project/my_project-account.xml</account-file>
        <calendar-file>products/import/basic/my_project/my_project-calendar.xml</calendar-file>
        <calendar-day-set-file>products/import/basic/my_project/my_project-calendar-day-set.xml</calendar-day-set-file>
        <calendar-day-file>products/import/basic/my_project/my_project-calendar-day.xml</calendar-day-file>
        <calendar-merge-file>products/import/basic/my_project/my_project-calendar-merge.xml</calendar-merge-file>
        <menu-group-category-file>products/import/basic/my_project/my_project-menu-group-category.xml</menu-group-category-file>
        <menu-group-file>products/import/basic/my_project/my_project-menu-group.xml</menu-group-file>
        <authz-resource-group-file>products/import/basic/my_project/my_project-authz-resource-group.xml</authz-resource-group-file>
        <authz-resource-file>products/import/basic/my_project/my_project-authz-resource.xml</authz-resource-file>
        <authz-subject-group-file>products/import/basic/my_project/my_project-authz-subject-group.xml</authz-subject-group-file>
        <authz-policy-file>products/import/basic/my_project/my_project-authz-policy.xml</authz-policy-file>
        <job-scheduler-file>products/import/basic/my_project/my_project-job-scheduler.xml</job-scheduler-file>
    </tenant-master>
    <extends-import>
        <extends-import-class>my_group.my_project.SampleExtendsImporter</extends-import-class>
        <extends-import-class>my_project/sample_extends_import.js</extends-import-class>
    </extends-import>
</import-data-config>
```

コラム

上述の例のほかに、既存のモジュールが定義しているセットアップ設定ファイルを参考にすると、
XMLファイルの完成状態をイメージしやすくなります。
必要に応じて参照してください。

また、後述の*データベース系*や*テナントマスタ系*、*拡張処理*についても同様です。
既存のモジュールが定義している各種ファイル群を併せて参照してください。

データベース系

データベース系では、DDL 文と DML 文を記述したファイルを、それぞれ必要に応じて(複数)定義できます。

セットアップ設定ファイルに定義を行った場合は、該当の SQL ファイルを作成してください。

データベース系	タグ名	定義内容
DDL	create-file	システムストレージからの相対パスを指定(拡張子sql)
DML	insert-file	システムストレージからの相対パスを指定(拡張子sql)

```
<?xml version="1.0" encoding="UTF-8"?>
<import-data-config
  xmlns="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config import-data-config.xsd ">

    <!-- データベース系 -->
    <database>

        <!-- DDL ※複数定義が可能です [0..*] -->
        <create-file>products/import/basic/my_project/my_project-ddl.sql</create-file>

        <!-- DML ※複数定義が可能です [0..*] -->
        <insert-file>products/import/basic/my_project/my_project-dml.sql</insert-file>

    </database>

</import-data-config>
```

コラム

組み込み可能な処理としては、以下のような SQL を想定しています。

- DDL
  - CREATE TABLE
  - CREATE INDEX
  - CREATE VIEW
- DML
  - INSERT
  - UPDATE
  - DELETE

コラム

intra-mart Accel Platform がサポートしているデータベースごとに、実行する DDL ファイルおよび DML ファイルを切り換えることができます。
たとえば、セットアップ設定ファイルの DDL ファイル名に my_project-ddl.sql というファイルを指定している場合に、
Oracle 専用の DDL 文を実行したい場合は、 **my_project-ddl_oracle.sql** というファイルを作成します。
このようにすると、テナントデータベースが Oracle の場合、my_project-ddl_oracle.sql が実行されます。

データベース	サフィックス
Oracle	oracle
Postgre SQL	postgre
SQL Server	sqlserver

※ Postgre SQL に対応するサフィックスは、 **“postgre”** です。”postgres” ではないので注意してください。

コラム

DML によって複数ロケール分の情報を登録する場合、次のように定義することを推奨します。

ロケールに関連しない情報の場合は、ファイル名にロケール ID サフィックスを **つけません** 。
ロケールに関連する情報の場合は、ファイル名にロケール ID サフィックスを **つけます** 。

定義情報	ファイルパス例
ロケールに関連しない情報	products/import/basic/my_project/my_project-dml.sql
ja ロケールの情報を定義した情報	products/import/basic/my_project/my_project-dml_ja.sql
en ロケールの情報を定義した情報	products/import/basic/my_project/my_project-dml_en.sql
zh_CN ロケールの情報を定義した情報	products/import/basic/my_project/my_project-dml_zh_CN.sql

このように定義することで、新たな言語情報を追加する際に、翻訳対象の特定と、翻訳元ロケール情報の特定が容易になります。

ただし、上記のとおりに定義しなければならないというわけではありません。
ロケール ID サフィックスなしのファイルに全ロケール分の情報を定義した場合でも、テナント環境セットアップは正常に動作します。

目的に合わせて定義の方法を検討してください。

テナントマスタ系

テナントマスタ系では、下記を記述した定義ファイルを、それぞれ必要に応じて(複数)定義できます。

セットアップ設定ファイルに定義を行った場合は、該当のXMLファイルを作成してください。

テナントマスタ系	タグ名	定義内容
ロール	role-file	システムストレージからの相対パスを指定(拡張子xml)
アカウント	account-file	システムストレージからの相対パスを指定(拡張子xml)
カレンダー：カレンダー	calendar-file	システムストレージからの相対パスを指定(拡張子xml)
カレンダー：日付情報セット	calendar-day-set-file	システムストレージからの相対パスを指定(拡張子xml)

テナントマスタ系	タグ名	定義内容
カレンダー：日付情報	calendar-day-file	システムストレージからの相対パスを指定(拡張子xml)
カレンダー：カレンダーマージ	calendar-merge-file	システムストレージからの相対パスを指定(拡張子xml)
メニュー：メニューカテゴリ	menu-group-category-file	システムストレージからの相対パスを指定(拡張子xml)
メニュー：メニューグループ	menu-group-file	システムストレージからの相対パスを指定(拡張子xml)
認可：リソースグループ	authz-resource-group-file	システムストレージからの相対パスを指定(拡張子xml)
認可：リソース	authz-resource-file	システムストレージからの相対パスを指定(拡張子xml)
認可：サブジェクト	authz-subject-group-file	システムストレージからの相対パスを指定(拡張子xml)
認可：ポリシー	authz-policy-file	システムストレージからの相対パスを指定(拡張子xml)
ジョブスケジューラ	job-scheduler-file	システムストレージからの相対パスを指定(拡張子xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<import-data-config
  xmlns="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config import-data-config.xsd ">

    <!-- テナントマスタ系 -->
    <tenant-master>

        <!-- ロール ※複数定義が可能です [0..*] -->
        <role-file>products/import/basic/my_project/my_project-role.xml</role-file>
        <role-file>products/import/basic/my_project/my_project-role_en.xml</role-file>
        <role-file>products/import/basic/my_project/my_project-role_ja.xml</role-file>
        <role-file>products/import/basic/my_project/my_project-role_zh_CN.xml</role-file>

        <!-- アカウント ※複数定義が可能です。 [0..*] -->
        <account-file>products/import/basic/my_project/my_project-account.xml</account-file>

        <!-- カレンダー -->

        <!-- カレンダー:カレンダー ※複数定義が可能です。 [0..*] -->
        <calendar-file>products/import/basic/my_project/my_project-calendar.xml</calendar-file>
```

```
    <calendar-file>products/import/basic/my_project/my_project-calendar_ja.xml</calendar-file>
    <calendar-file>products/import/basic/my_project/my_project-calendar_en.xml</calendar-file>
    <calendar-file>products/import/basic/my_project/my_project-calendar_zh_CN.xml</calendar-file>

    <!-- カレンダー:日付情報セット ※複数定義が可能です。 [0..*] -->
    <calendar-day-set-file>products/import/basic/my_project/my_project-calendar-day-set.xml</calendar-day-set-file>
    <calendar-day-set-file>products/import/basic/my_project/my_project-calendar-day-set_ja.xml</calendar-day-set-file>
    <calendar-day-set-file>products/import/basic/my_project/my_project-calendar-day-set_en.xml</calendar-day-set-file>
    <calendar-day-set-file>products/import/basic/my_project/my_project-calendar-day-set_zh_CN.xml</calendar-day-set-file>

    <!-- カレンダー:日付情報 ※複数定義が可能です。 [0..*] -->
    <calendar-day-file>products/import/basic/my_project/my_project-calendar-day.xml</calendar-day-file>
    <calendar-day-file>products/import/basic/my_project/my_project-calendar-day_ja.xml</calendar-day-file>
    <calendar-day-file>products/import/basic/my_project/my_project-calendar-day_en.xml</calendar-day-file>
    <calendar-day-file>products/import/basic/my_project/my_project-calendar-day_zh_CN.xml</calendar-day-file>

    <!-- カレンダー:カレンダーマージ ※複数定義が可能です。 [0..*] -->
    <calendar-merge-file>products/import/basic/my_project/my_project-calendar-merge.xml</calendar-merge-file>

    <!-- メニュー -->

    <!-- メニュー:メニューカテゴリ ※複数定義が可能です。 [0..*] -->
    <menu-group-category-file>products/import/basic/my_project/my_project-menu-group-category.xml</menu-group-category-file>
    <menu-group-category-file>products/import/basic/my_project/my_project-menu-group-category_ja.xml</menu-group-category-file>
    <menu-group-category-file>products/import/basic/my_project/my_project-menu-group-category_en.xml</menu-group-category-file>
    <menu-group-category-file>products/import/basic/my_project/my_project-menu-group-category_zh_CN.xml</menu-group-category-file>

    <!-- メニュー:メニューグループ ※複数定義が可能です。 [0..*] -->
    <menu-group-file>products/import/basic/my_project/my_project-menu-group.xml</menu-group-file>
    <menu-group-file>products/import/basic/my_project/my_project-menu-group_ja.xml</menu-group-file>
    <menu-group-file>products/import/basic/my_project/my_project-menu-group_en.xml</menu-group-file>
    <menu-group-file>products/import/basic/my_project/my_project-menu-group_zh_CN.xml</menu-group-file>

    <!-- 認可 -->
```

```
    <!-- 認可 -->

    <!-- 認可:リソースグループ ※複数定義が可能です。 [0..*] -->
    <authz-resource-group-file>products/import/basic/my_project/my_project-authz-resource-group.xml</authz-resource-group-file>
    <authz-resource-group-file>products/import/basic/my_project/my_project-authz-resource-group_ja.xml</authz-resource-group-file>
    <authz-resource-group-file>products/import/basic/my_project/my_project-authz-resource-group_en.xml</authz-resource-group-file>
    <authz-resource-group-file>products/import/basic/my_project/my_project-authz-resource-group_zh_CN.xml</authz-resource-group-file>

    <!-- 認可:リソース ※複数定義が可能です。 [0..*] -->
    <authz-resource-file>products/import/basic/my_project/my_project-authz-resource.xml</authz-resource-file>
    <authz-resource-file>products/import/basic/my_project/my_project-authz-resource_ja.xml</authz-resource-file>
    <authz-resource-file>products/import/basic/my_project/my_project-authz-resource_en.xml</authz-resource-file>
    <authz-resource-file>products/import/basic/my_project/my_project-authz-resource_zh_CN.xml</authz-resource-file>

    <!-- 認可:サブジェクトグループ ※複数定義が可能です。 [0..*] -->
    <authz-subject-group-file>products/import/basic/my_project/my_project-authz-subject-group.xml</authz-subject-group-file>
    <authz-subject-group-file>products/import/basic/my_project/my_project-authz-subject-group_ja.xml</authz-subject-group-file>
    <authz-subject-group-file>products/import/basic/my_project/my_project-authz-subject-group_en.xml</authz-subject-group-file>
    <authz-subject-group-file>products/import/basic/my_project/my_project-authz-subject-group_zh_CN.xml</authz-subject-group-file>

    <!-- 認可:ポリシー ※複数定義が可能です。 [0..*] -->
    <authz-policy-file>products/import/basic/my_project/my_project-authz-policy.xml</authz-policy-file>

    <!-- ジョブスケジューラ ※複数定義が可能です。 [0..*] -->
    <job-scheduler-file>products/import/basic/my_project/my_project-job-scheduler.xml</job-scheduler-file>
    <job-scheduler-file>products/import/basic/my_project/my_project-job-scheduler_ja.xml</job-scheduler-file>
    <job-scheduler-file>products/import/basic/my_project/my_project-job-scheduler_en.xml</job-scheduler-file>
    <job-scheduler-file>products/import/basic/my_project/my_project-job-scheduler_zh_CN.xml</job-scheduler-file>

  </tenant-master>

</import-data-config>
```

コラム

各テナントマスタ情報のインポートファイルについては、次を参照してください。

- テナント管理機能の「インポート・エクスポート仕様書」
- 製品モジュールがテナント環境セットアップで利用するインポートファイル
- 画面上でマスタ情報を作成した後にエクスポートジョブを実行することにより出力されるファイル

各テナントマスタ情報インポートファイルのフォーマットファイル(xsd)は以下の通りです。

テナントマスタ系	タグ名	定義内容
ロール	role-file	WEB-INF/schema/role-data.xsd
アカウント	account-file	WEB-INF/schema/account-data.xsd
カレンダー：カレンダー	calendar-file	WEB-INF/schema/calendar.xsd
カレンダー：日付情報セット	calendar-day-set-file	WEB-INF/schema/calendar-day-set.xsd
カレンダー：日付情報	calendar-day-file	WEB-INF/schema/calendar-day.xsd
カレンダー：カレンダーマージ	calendar-merge-file	WEB-INF/schema/calendar-merge.xsd
メニュー：メニューカテゴリ	menu-group-category-file	WEB-INF/schema/menu-group-category-data.xsd
メニュー：メニューグループ	menu-group-file	WEB-INF/schema/menu-group-data.xsd
認可：リソースグループ	authz-resource-group-file	WEB-INF/schema/authz-resource-group.xsd
認可：リソース	authz-resource-file	WEB-INF/schema/authz-resource.xsd
認可：サブジェクト	authz-subject-group-file	WEB-INF/schema/authz-subject-group.xsd
認可：ポリシー	authz-policy-file	WEB-INF/schema/authz-policy.xsd
ジョブスケジューラ	job-scheduler-file	WEB-INF/schema/im-job-scheduler-data.xsd

コラム

テナントマスタ情報を複数ロケール分インポートする場合、次のように定義することを推奨します。

ロケールに関連しない情報の場合は、ファイル名にロケール ID サフィックスを **つけません** 。
ロケールに関連する情報の場合は、ファイル名にロケール ID サフィックスを **つけます** 。

定義情報	ファイルパス例
ロケールに関連しない情報	products/import/basic/my_project/my_project-role.xml
ja ロケールの情報を定義した情報	products/import/basic/my_project/my_project-role_ja.xml
en ロケールの情報を定義した情報	products/import/basic/my_project/my_project-role_en.xml
zh_CN ロケールの情報を定義した情報	products/import/basic/my_project/my_project-role_zh_CN.xml

このように定義することで、新たな言語情報を追加する際に翻訳対象の特定と、翻訳元の情報の特定が容易になります。

ただし、上記のとおりに定義しなければならないというわけではありません。
ロケール ID サフィックスなしのファイルに全ロケール分の情報を定義した場合でも、テナント環境セットアップは正常に動作します。

目的に合わせて定義の方法を検討してください。

コラム

ファイル群はモジュールやスキーマバージョンを意識した構成で管理することを推奨します。

拡張処理

拡張処理では、Java クラスまたはサーバサイド JavaScript を、それぞれ必要に応じて(複数)定義できます。

拡張処理として実行する処理を、 **拡張インポート** と呼びます。

セットアップ設定ファイルに定義を行った場合は、拡張インポート実装を作成してください。

拡張処理	タグ名	定義内容
プログラム	extends-import-class	Java クラス名 または サーバサイド JavaScript のファイルパス

```
<?xml version="1.0" encoding="UTF-8"?>
<import-data-config
  xmlns="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://intra_mart.co.jp/system/service/provider/importer/config/import-data-config import-data-config.xsd ">

    <!-- 拡張処理 -->
    <extends-import>

        <!-- Javaクラスを指定する場合 ※複数定義が可能です。 [0..*] -->
        <!--  インタフェース jp.co.intra_mart.foundation.security.ExtendsImport を実装したJavaクラスの完全修飾クラス名を指定します。 -->
        <extends-import-class>my_group.my_project.SampleExtendsImporter</extends-import-class>

        <!-- サーバサイドJavaScriptを指定する場合 ※複数定義が可能です。 [0..*] -->
        <!--  WEB-INF/jssp/platform/srcなど、サーバサイドJavaScriptのルートからの相対パスを、拡張子".js"をつけて指定します。 -->
        <extends-import-class>my_project/sample_extends_import.js</extends-import-class>

    </extends-import>

</import-data-config>
```

- Java クラスを指定する場合
  - インタフェース jp.co.intra_mart.foundation.security.ExtendsImport の実装クラスを、完全修飾クラス名で指定してください。
  - 拡張インポートでは doImport メソッドが実行されます。
  - インタフェース jp.co.intra_mart.foundation.security.ExtendsImport について、詳細は API リストを参照してください。
- サーバサイド JavaScript を指定する場合
  - doImport メソッドを実装したサーバサイド JavaScript を、サーバサイド JavaScript のルートからの相対パスで拡張子”.js”をつけて指定してください。
  - 拡張インポートでは doImport メソッドが実行されます。
  - doImport メソッドの第一引数にはテナント ID を渡します。
  - doImport メソッドは処理結果を Boolean 型で返却する必要があります。

コラム

インタフェース jp.co.intra_mart.foundation.security.ExtendsImport は、次の jar に入っています。

- im_import_export_base-x.x.x-main.jar （ x.x.x はバージョン番号です）

上記の jar は、モジュール・プロジェクトに Accel Platform Library の設定が行われることで参照可能となります。

e Builder 2013 Spring 以降のバージョンでモジュール・プロジェクトを作成した場合は、自動で設定されるため、特別な対応は不要です。

e Builder 2013 Spring より前のバージョンでモジュール・プロジェクトを作成した場合は、個別で Accel Platform Library の設定を行う必要があります。
「intra-mart e Builder for Accel Platform アプリケーション開発ガイド」を参照してください。

コラム

拡張インポートでは、フレームワークによるトランザクション制御は行われません。
必要に応じて独自にトランザクション制御を行ってください。

参考：テナント環境セットアップのトランザクション単位

テナント環境セットアップは、各種モジュールで定義・配置されたセットアップ設定ファイルの情報をもとに
モジュール単位で以下の順番で実行されます。

1. データベース系：DDL
2. **トランザクション開始**
    1. データベース系：DML
    2. テナントマスタ系
3. **トランザクション終了**
4. 拡張処理

コラム

DDLと拡張処理は、トランザクション制御されません。
特に拡張処理に関しては、必要に応じて独自にトランザクション制御を行ってください。

参考：テナント環境セットアップの実行順序

テナント環境セットアップの実行順序は、原則としてモジュールの依存関係順で実行されます。

対象のユーザモジュールが依存するすべてのモジュールのテナント環境セットアップが完了した後に、対象のユーザモジュールのテナント環境セットアップが実行されます。

テナント環境セットアップの実行順序の詳細は「テナント環境セットアップ 仕様書」を参照してください。

ユーザモジュールの作成

セットアップ設定ファイルとそこで定義したインポートファイルを含むユーザモジュールを作成します。

ユーザモジュールの作成方法については「intra-mart e Builder for Accel Platform アプリケーション開発ガイド」を参照してください。

ユーザモジュールを組み込んだwarファイルの出力

作成したユーザモジュールを組み込んだ war ファイルを出力します。

ユーザモジュールの組み込み方法については「intra-mart e Builder for Accel Platform アプリケーション開発ガイド」を参照してください。
war ファイルの出力方法については「セットアップガイド」を参照してください。

warファイルのデプロイと intra-mart Accel Platform の起動

Web アプリケーションサーバを起動し、war ファイルをデプロイします。

手順については「セットアップガイド」を参照してください。

テナント環境セットアップの実行

システム管理者でログインし、テナント環境セットアップ画面を開きます。
テナント環境セットアップのボタンが有効になり、ボタンを押下すると、テナント環境セットアップが行えます。

手順については「システム管理者操作ガイド」を参照してください。

モジュール情報の設定

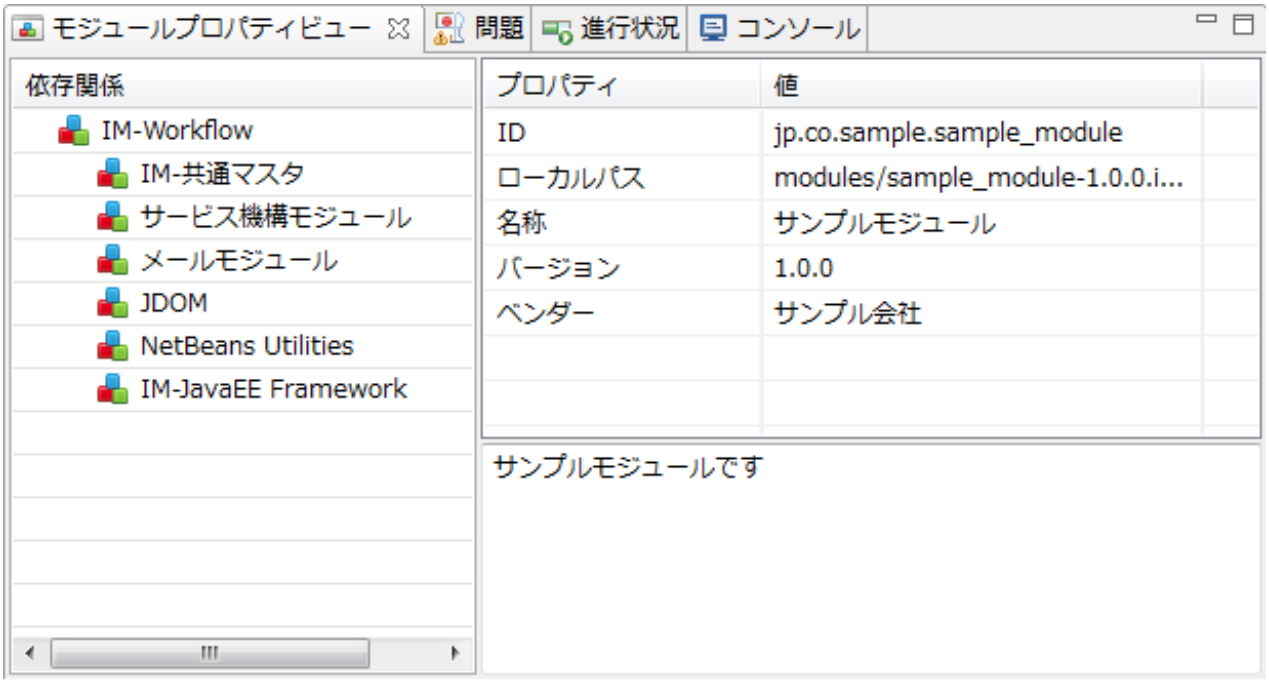
概要

本項ではIM-Juggling等で表示される、ユーザモジュー情報の定義方法について説明します。
本項でのモジュール情報とは、IM-Jugglingの「モジュールプロパティビュー」にて表示される情報を指します。

モジュール情報の定義

モジュール情報は、プロジェクト直下にあるmodule.xmlにて<module>内にて定義します。
定義したモジュールの情報はIM-Jugglingの「モジュールプロパティビュー」にて確認することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<module conf:schemaLocation="urn:intramart:jackling:toolkit:configurations
configurations.xsd" xmlns="urn:intramart:jackling:module"
xmlns:conf="urn:intramart:jackling:toolkit:configurations"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemeLocation="urn:intramart:jackling:module module.xsd">
  <id>jp.co.sample.sample_module</id>
  <version>1.0.0</version>
  <type>module</type>
  <name>サンプルモジュール</name>
  <vendor>サンプル会社</vendor>
  <description>サンプルモジュールです</description>
  <dependencies>
    <dependency>
      <module-id>jp.co.intra_mart.im_workflow</module-id>
      <verified-version min="8.0.5" max="8.0.7">8.0.6</verified-version>
    </dependency>
  </dependencies>
</module>
```



<module>内に設定する各要素は以下の値を設定します。

要素名	モジュールプロパティビューのプロパティ	説明
id	ID	モジュールID
version	バージョン	モジュールのバージョン
type	-	モジュールタイプ。 ユーザモジュールは、この値を変更しないでください。
name	名称	モジュールの名称。
vender	ベンダー	モジュールの提供ベンダー
description	プロパティテーブルの下の説明欄	モジュールの説明文
dependencies	依存関係のリスト	モジュールで定義されている依存関係のリスト。 詳細は「*モジュール間の依存関係*」を参照してください。

コラム

IM-Jugglingを別のロケールで動作させた場合の多言語表示をプロパティファイルを用いて行うことができます。
プロパティファイルはモジュール直下にあるmessage.propertiesファイルとなります。
このプロパティは標準では日本語、中国語、英語、およびデフォルトの4種類を用意してあります。
またプロパティファイルを利用する対象となるのはname、vender、descriptionの3属性となります。

表示するロケールに置換するためには、置換文字列として ‘${‘ + プロパティ名 + ‘}’ の形で指定します。
書き方を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<module conf:schemaLocation="urn:intramart:jackling:toolkit:configurations
configurations.xsd" xmlns="urn:intramart:jackling:module"
xmlns:conf="urn:intramart:jackling:toolkit:configurations"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemeLocation="urn:intramart:jackling:module module.xsd">
  <id>jp.co.sample.sample_module</id>
  <version>1.0.0</version>
  <type>module</type>
  <name>${module.name}</name>
  <vendor>${module.vendor}</vendor>
  <description>${module.description}</description>
  <dependencies>
    <dependency>
      <module-id>jp.co.intra_mart.im_workflow</module-id>
      <verified-version min="8.0.5" max="8.0.7">8.0.6</verified-version>
    </dependency>
  </dependencies>
</module>
```

モジュール間の依存関係

概要

本章ではモジュール間の依存関係の概念について説明します。

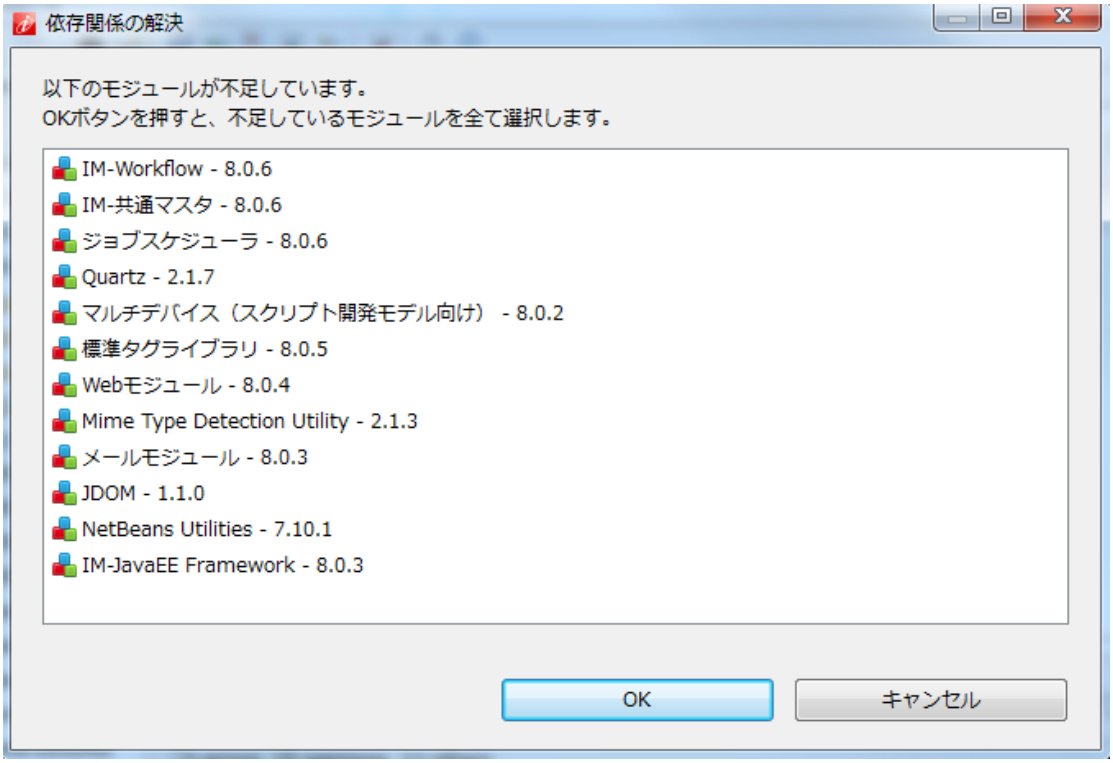
依存関係とは

依存関係の定義はモジュール間の関係性を示します。
依存関係はモジュールごとに定義を行います。
定義されたモジュールは、定義したモジュールを含むwarを作成する際に必須のモジュールとなります。

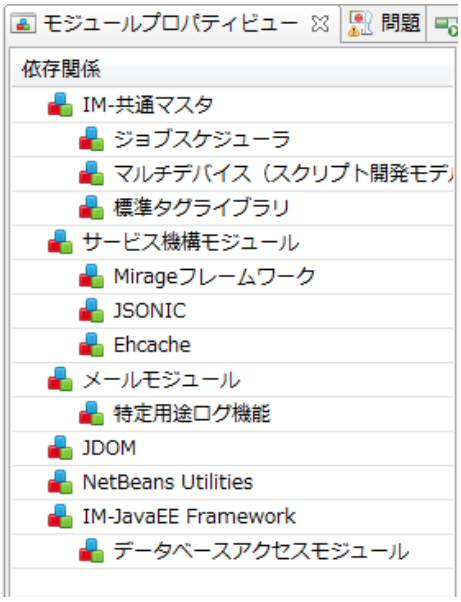
ex. サンプルモジュールのmodule.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<module conf:schemaLocation="urn:intramart:jackling:toolkit:configurations
configurations.xsd" xmlns="urn:intramart:jackling:module"
xmlns:conf="urn:intramart:jackling:toolkit:configurations"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemeLocation="urn:intramart:jackling:module module.xsd">
  <id>jp.co.sample.sample_module</id>
  <version>1.0.0</version>
  <type>module</type>
  <name>サンプルモジュール</name>
  <vendor>サンプル会社</vendor>
  <description>サンプルモジュールです</description>
  <dependencies>
    <dependency>
      <module-id>jp.co.intra_mart.im_workflow</module-id>
      <verified-version min="8.0.5" max="8.0.7">8.0.5</verified-version>
    </dependency>
  </dependencies>
</module>
```

例として上記のmodule.xmlで設定されている依存関係の例を見てみます。
これは「サンプルモジュール」と名付けられたモジュールのmodule.xmlです。
サンプルモジュールは依存関係として、IM-Workflow モジュール(jp.co.intra_mart.im_workflow)を定義しています。
これはサンプルモジュールを同梱した環境をwarで作る際に、必ずIM-Workflowモジュールもwarで含まれるように
設定しないといけないことになります。
実際にIM-Jugglingでwarを作成する際に依存関係に含まれているモジュールが含まれないと
依存関係の対象モジュール、およびそのモジュールに依存関係で紐づくモジュールが足りないことをエラーで表示します。



選択するモジュールが定義している依存モジュール、および依存関係で紐づくモジュールの一覧は「モジュールプロパティビュー」にて確認することができます。



依存関係の設定の必要性

開発者は自身が作成したモジュールがどのモジュールに依存しているかを定義すべきです。
もし、intra-mart Accel Platformのバージョンアップに伴って対応を行う時に、依存関係を記述しておくことによって、調査すべき変更内容と自らのモジュールに対する影響の調査範囲を絞り込むことができます。
また、定義しておくことによって、ユーザモジュールを含んだ環境を作成する際に必要なモジュールがぬけていたといったような設定漏れを防ぐことができます。

依存関係の定義方法

モジュールプロジェクトでは、プロジェクト直下にあるmodule.xmlで依存関係を示します。
module.xml内の<dependencies>の子要素である<dependency>に依存関係あるモジュールの情報を記述します。

```
<?xml version="1.0" encoding="UTF-8"?>
<module conf:schemaLocation="urn:intramart:jackling:toolkit:configurations
configurations.xsd" xmlns="urn:intramart:jackling:module"
xmlns:conf="urn:intramart:jackling:toolkit:configurations"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemeLocation="urn:intramart:jackling:module module.xsd">
  <id>jp.co.sample.sample_module</id>
  <version>1.0.0</version>
  <type>module</type>
  <name>サンプルモジュール</name>
  <vendor>サンプル会社</vendor>
  <description>サンプルモジュールです</description>
  <dependencies>
    <dependency>
      <module-id>jp.co.intra_mart.im_workflow</module-id>
      <verified-version min="8.0.5" max="8.0.7">8.0.5</verified-version>
    </dependency>
  </dependencies>
</module>
```

module.xmlにおける<dependency>の各要素の定義は以下のようになります。

要素名	説明
module-id	依存関係に設定するモジュールのIDです。 モジュールのIDは以下で確認することができます。 ・「モジュールプロパティビュー」に表示されるプロパティ「ID」の値 ・IM-Jugglingの「 *モジュール構成Excel出力* 」で出力したExcelファイル

要素名	説明
verified-version	依存関係に設定するモジュールのバージョンです。 値にバージョンを指定することにより、依存関係に設定したモジュールのバージョンを指定できます。 範囲指定の属性として「min」属性と「max」属性を指定できます。 minは依存するモジュールの最小バージョンを、maxは依存するモジュールの最大バージョンを指定できます。 min、maxを指定することによって、そのモジュールに依存するバージョンの範囲を指定することができます。 min、maxをいずれも指定しなかった場合は、値に指定したバージョン以上となります。 なお、minのみ設定した場合は設定したバージョンより大きいバージョンのいずれか、maxのみ指定した場合は設定したバージョンより小さいバージョンのいずれかという範囲になります。 また、minにバージョンを指定する場合は、上記の例のように、この要素の値にはminに指定したバージョンと同じものを指定してください。

コラム

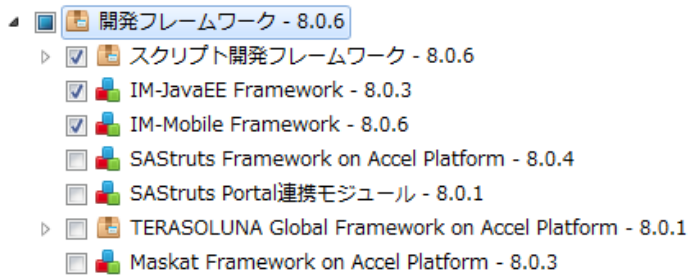
「モジュール構成Excel出力」は以下のように行います。

1. juggling.imをエディタで開きます。
2. エディタ右上のアイコンでビルドウィザードを起動します。
3. Jugglingビルドウィザードで、「モジュール構成Excel出力」を選択して、「次へ」をクリックします。
4. 出力場所などを設定して、「次へ」をクリックします。
5. 内容を確認して、「終了」をクリックします。
6. モジュール構成情報のExcelファイルが出力されます。

依存関係に設定するモジュールの選別

本項では、開発者が作成するモジュールに対して、何を基準にして依存関係のモジュールを設定すべきかについて例をあげます。

- **利用している開発フレームワークモジュール**
  開発者がユーザモジュールを開発する際に利用する開発フレームワークのモジュールです。
  開発フレームワークモジュールとは、 IM-Jugglingにおいて、juggling.im ファイルを「IM-Juggling Editor」で開いたときにベースモジュールの開発フレームワーク内にて選択できるモジュール群です。



開発者は開発フレームワークの依存関係を指定してあげることで、開発フレームワークモジュール、もしくは開発フレームワークが依存するサードパーティライブラリの漏れを防ぐことができます。

- **利用しているAPI、もしくは機能を含んでいるモジュール**
  例えば開発者が、IMBoxのAPIを利用して、特定のアクションが発生した際にIMBoxに投稿するような機能をユーザモジュールに含む場合、依存関係にIMBoxのモジュールを含める必要があります。
  さらにこのIMBox投稿機能をジョブスケジューラにて定期実行を行いたいとした場合、IMBoxのモジュールだけでなくジョブスケジューラのモジュールを依存関係に含める必要があります。
- **カスタマイズするソースを含んでいるモジュール**
  intra-mart Accel Platformにて提供されているソースのカスタマイズを行いたい場合、カスタマイズ元のソースを提供しているモジュールを依存関係に含みます。
  カスタマイズしたいソースがどのモジュールに含まれているかはIM-Jugglingの「*モジュール構成Excel出力*」で出力したExcelファイルにて確認することができます。
  もし、Excel上に複数存在していた場合、シート番号がより大きいモジュールを依存関係として設定してください。
  カスタマイズしているモジュールの依存関係を正しく設定してあげることにより、展開時にカスタマイズしたソースが反映されるようになります。

> **注意**
>
> 同一のパッケージ名とクラス名でのカスタマイズしたJavaソースの作成は非推奨です。
> 上記のようなクラスを作成した場合、 jar に含んだ場合は正常にカスタマイズソースが呼び出されない可能性があります。
> そのため、カスタマイズしたJavaソースに対して、お客様独自で配置したソースかどうかが分かるよう、明確な違いが分かるパッケージ名、およびクラス名でのカスタマイズを行ってください。