記事
Mihoko Iijima · 2020年10月27日 4m read

【はじめてのInterSystems IRIS】Interoperability（相互運用性）：メッセージ

この記事はこちらの投稿の続きの内容です。

前回の記事
では、プロダクションとは？について確認しました。また、サンプルコードを動かしながらプロダクションに流れるメッセージの中身をトレース画面で確認しました。

今回は記事では、システム統合を行うための必要な開発内容の中から、コンポーネント間のデータ送受信に使用される **メッセージ** について、作成するときの考え方や定義方法を確認していきます。

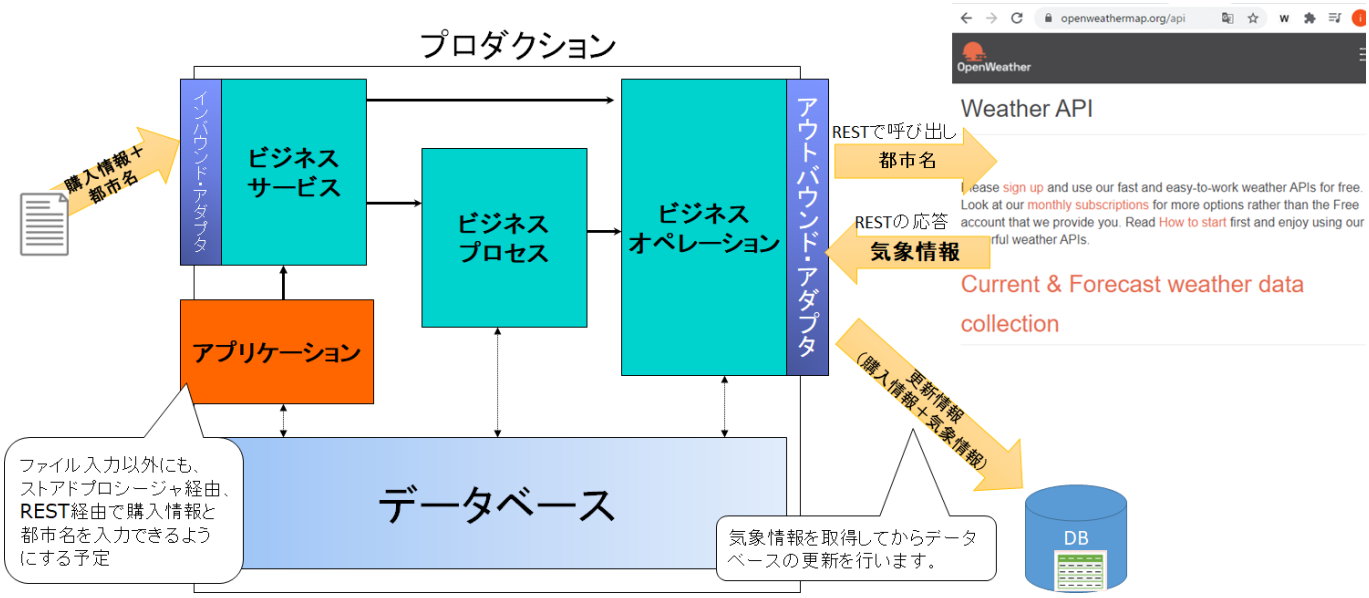
- プロダクション（前回の記事）
- **メッセージ**
- コンポーネント
  - ビジネス・サービス
  - ビジネス・プロセス
  - ビジネス・オペレーション

メッセージ を作成する前に、サンプルのテーマを再度確認しましょう。

> ショッピングサイトを運営している会社があり、季節に合わせ商品情報の表示順を変更する作業を行っています。
> ところが、季節を問わず良く売れるもの、想定しなかった時期に売れるものもあり、現在の表示順変更ルールにうまく合いません。
> そこで、季節に合わせた表示順ではなく、そのときの気温にあわせた表示順に変更できないか検討した結果、購入物品に対してそのときの気温を調査する必要が出てきました。
> 気象情報の確認には、外部の Web API が利用できるため、購入されたタイミングで気象情報を収集し、後で確認できるようにデータベースに情報を登録していく予定です。

このテーマから、以下の内容が確認できます。

外部から受信する情報は「購入した商品情報と気象情報の検索に使用する都市名」であること
IRIS から外部システムへ処理を依頼するための送信情報は「都市名」であること
処理結果である応答内容は「気象情報」であること



このテーマから、システム統合に必要なコンポーネントを実装しますが、その前に、コンポーネントを動作させるためには、中継データである**メッセージ**を送受信する必要があり、**メッセージ**を利用するためには、メッセージクラスを定義する必要があります。

メッセージクラスは、
コンポーネントを動作させるためにどのような情報（＝メッセージ）を送受信したらよいかを考えながら設計します。
今回の流れでは、以下の2種類の情報が必要となります。

A)　**気象情報**を取得するため外部Web API に送信する**都市名**

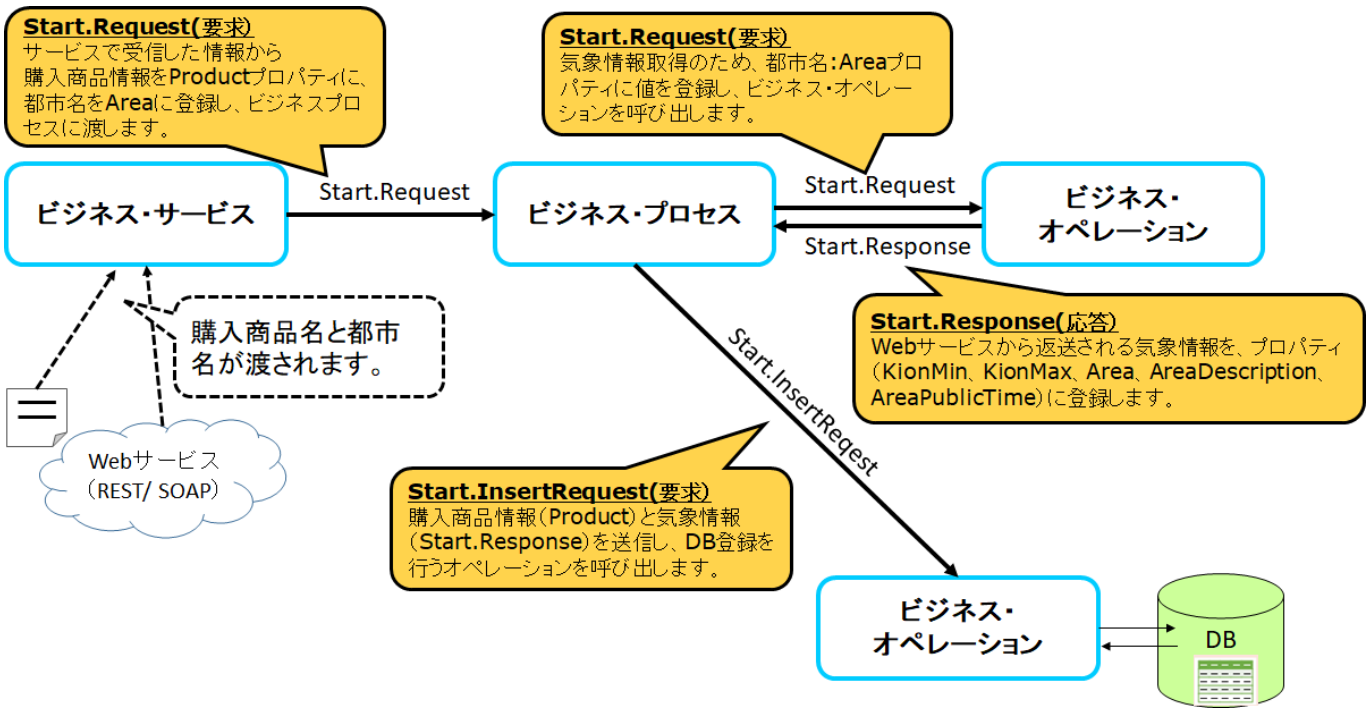
B)　DBに記録するための**気象情報**と**購入商品名**

A) の**都市名** と B) の**購入商品名** は、IRIS への入力情報に含めてもらうことで対応できます。

B) の**気象情報**は、**外部 Web API の応答情報から取得できます。**

確認できている情報から各コンポーネントを送受信する**メッセージ**としてどんな情報が必要になるかを考えた図が以下の通りです。

黄色い吹き出しの**1行目がメッセージクラス名**、2行目以降にプロパティに何を設定するか、を記載しています。



サンプルコードでは、以下3種類の**メッセージ**を用意しました。

Start.Request **（要求メッセージ）**
サービスからプロセス、プロセスから気象情報取得用オペレーションの呼び出しで、***購入商品名***と**都市名**を送信するために使用します。

Start.Response **（応答メッセージ）**
気象情報取得用オペレーションの結果（**気象情報**）をプロセスに返送するために使用します。

Start.InsertRequest **（要求メッセージ）**
プロセスからDB登録用オペレーションへ**気象情報**と**購入商品名**を送信するために使用します。

メッセージはスーパークラスの指定が決まっていて、要求メッセージは **Ens.Request** 、応答メッセージは **Ens.Response** を継承します。

以下、要求メッセージ：Start.Request の定義例です。

src > Start > Request.cls > Start.Request

```
1  Class Start.Request Extends Ens.Request
2  {
3
4  Property Product As %String;
5
6  Property Area As %String;
7
```

以下、応答メッセージの定義例です。

src > Start > Response.cls > Start.Response

```
1   Class Start.Response Extends Ens.Response
2   {
3
4   Property KionMax As %Numeric;
5
6   Property KionMin As %Numeric;
7
8   Property Area As %String;
9
10  Property AreaDescription As %String(MAXLEN = 1000);
11
12  Property AreaPublicTime As %String;
13
14  Storage Default
```

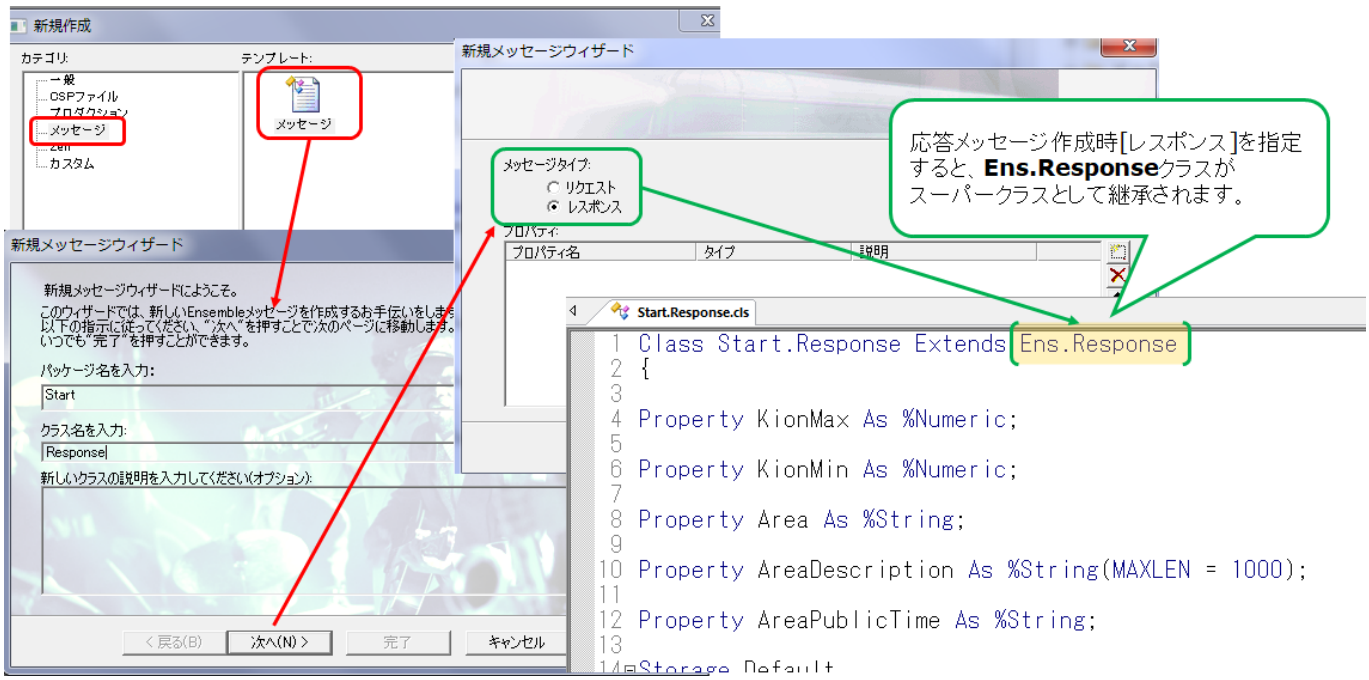
DB登録依頼で送付する要求メッセージの Start.InsertRequest は以下の通りです。
（天気情報を取得後に返送される Start.Response の情報をそのまま
WeatherInfoプロパティに設定する予定で定義しています）

src > Start > InsertRequest.cls > Start.InsertRequest

```
1   Class Start.InsertRequest Extends Ens.Request
2   {
3
4   Property Product As %String;
5
6   Property WeatherInfo As Start.Response;
7
```

スタジオで作成する場合、メッセージ作成用ウィザードを利用することもできます。

ご参考）スタジオでレスポンスクラスを作成する手順



ここまでのポイント

メッセージクラスは、コンポーネントを動作させるために「どのような情報（＝**メッセージ**）を送受信したらよいか」を考えながら設計します。

各コンポーネントを動かすための情報であるメッセージクラスが実装できたら、次はコンポーネント用クラスを作成します。

#初心者 #相互運用性 #InterSystems IRIS #InterSystems IRIS for Health

ソースURL:
https://jp.community.intersystems.com/post/%E3%80%90%E3%81%AF%E3%81%98%E3%82%81%E3%81%A6%E3%81%AEintersystems-iris%E3%80%91interoperability%EF%BC%88%E7%9B%B8%E4%BA%92%E9%81%8B%E7%94%A8%E6%80%A7%EF%BC%89%EF%BC%9A%E3%83%A1%E3%83%83%E3%82%BB%E3%83%BC%E3%82%B8