



HCL Volt MX

オブジェクトサービス

Student Guide



HCL Software Academy for HCL Digital Solutions

Creating a new generation of experts

もくじ

オブジェクトサービス	Error! Bookmark not defined.
前提条件	3
Foundry で Data Storage オブジェクトサービスを構成する	3
Foundry で Integration & Orchestration オブジェクトサービスを構成する	11
Iris からオブジェクトサービスを呼び出す	15

オブジェクトサービス

このレッスンでは、Foundry でオブジェクトサービスを構成し、Foundry で構成されたオブジェクトサービスを使用するコードを Iris で記述する方法について紹介します。

このドキュメントでは、このレッスンのハンズオン部分の詳細な手順を説明します。

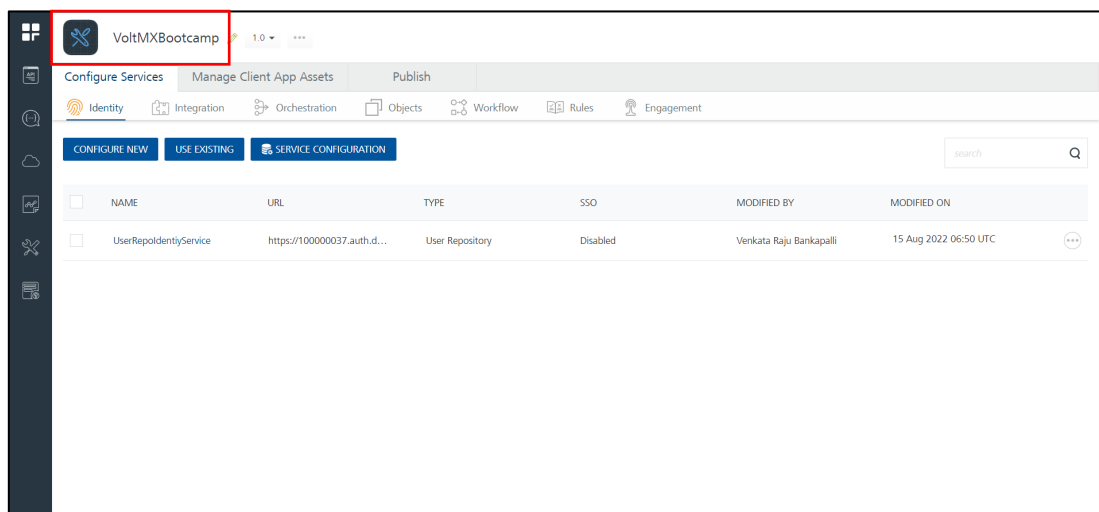
前提条件

- 前のレッスンでハンズオンの手順を完了していること。

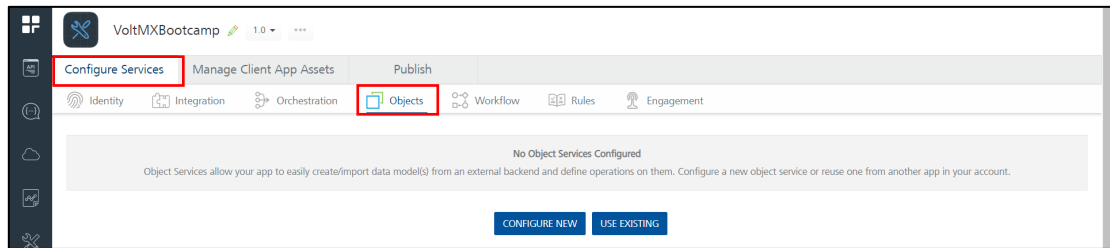
Foundry で Data Storage オブジェクトサービスを構成する

注意事項

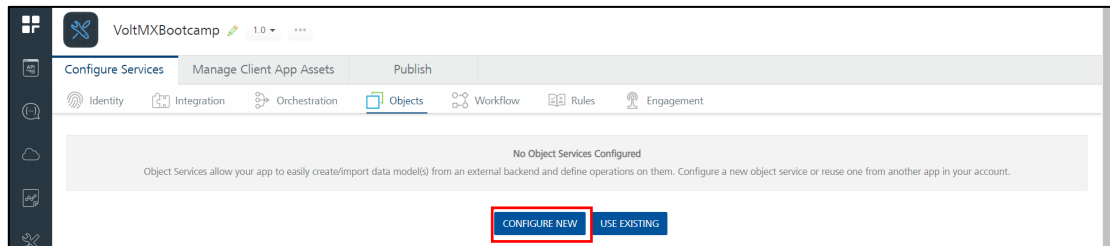
- 前のレッスンで作業していた **Foundry** プロジェクトで、以下に説明する手順を引き続き実行します。
- Foundry コンソールにログインします。
- Foundry アプリ **VoltMXBootcamp** を開きます。



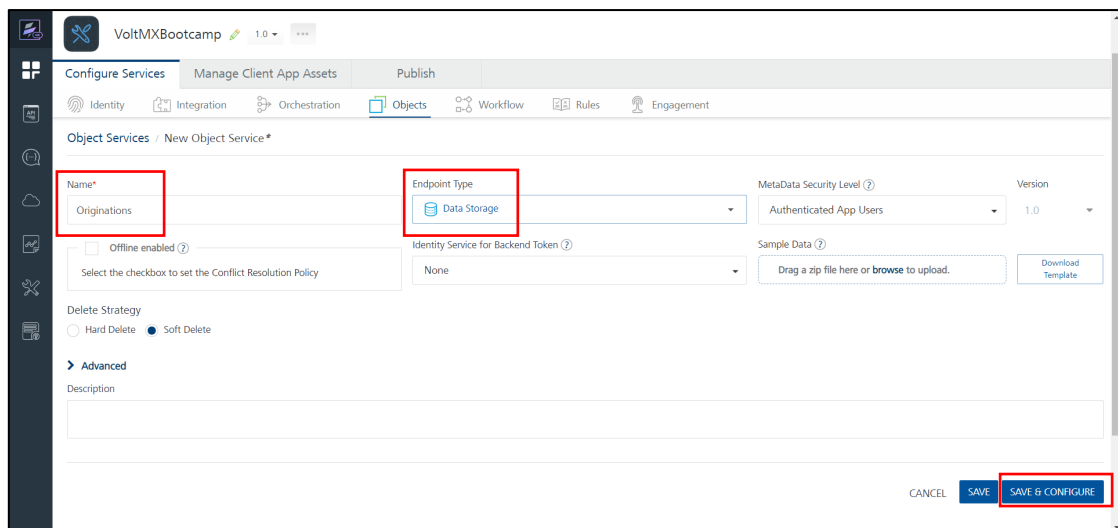
- **Configure Services > Objects** を開きます。



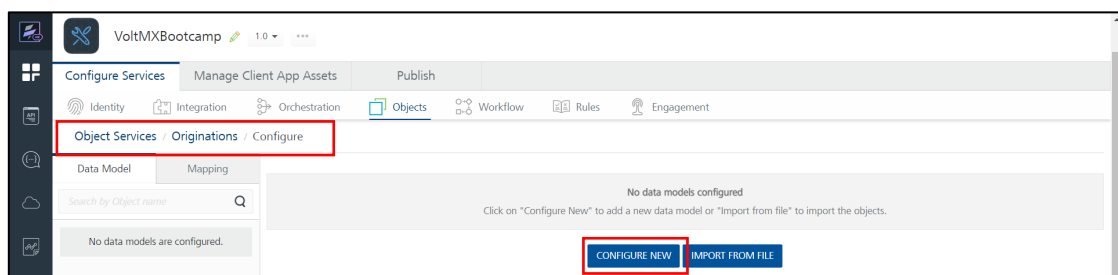
- **CONFIGURE NEW** をクリックします。



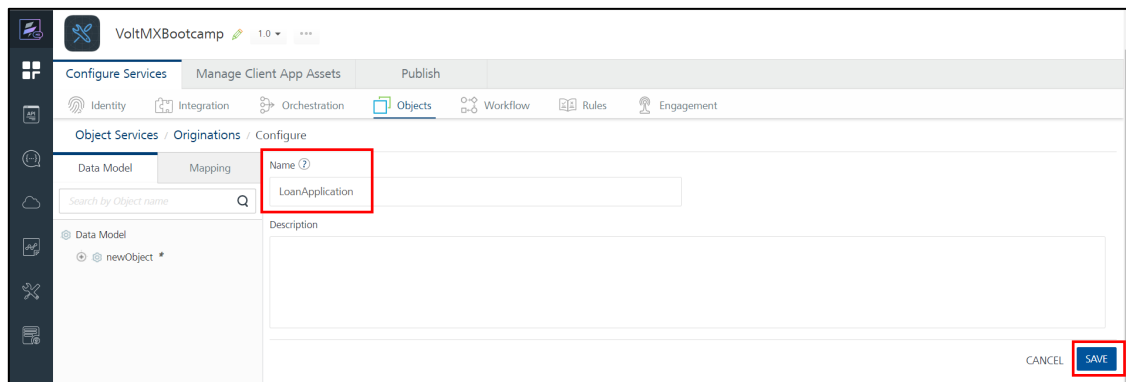
- **Name** を **Originations** に、**Endpoint Type** を **Data Storage** に設定します。 **SAVE & CONFIGURE** をクリックします。



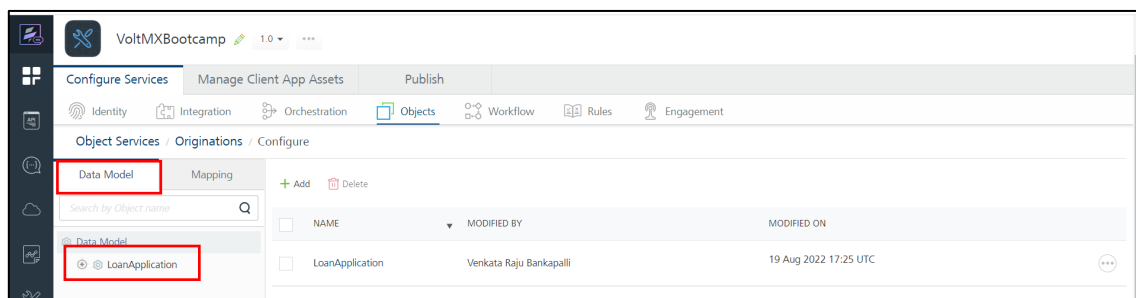
- オブジェクトサービス Originations が作成されます。
- **CONFIGURE NEW** をクリックして、このオブジェクトサービスにオブジェクトを設定します。



- オブジェクトの **Name** を **LoanApplication** とします。 **SAVE** をクリックします。



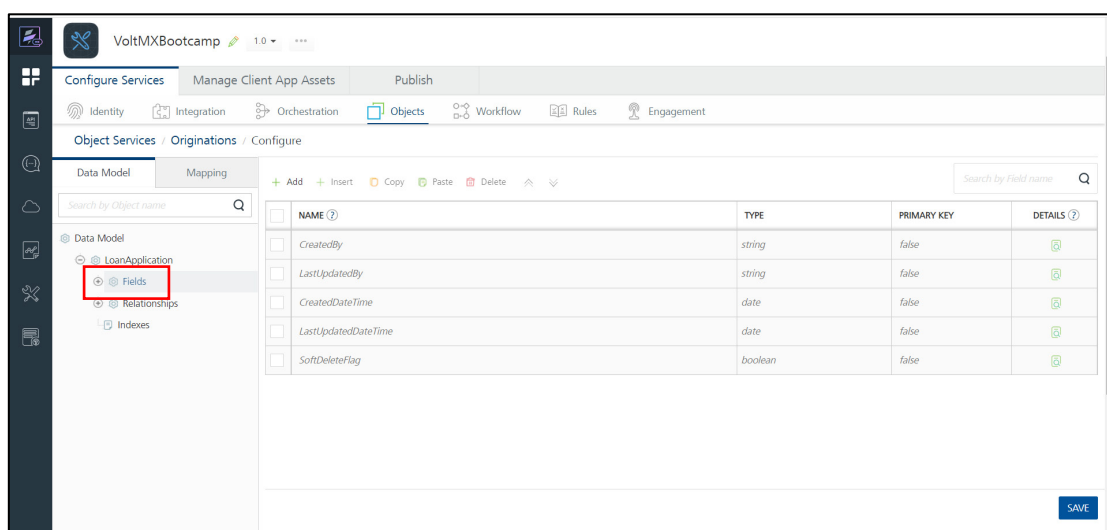
- LoanApplication** オブジェクトが作成されます。



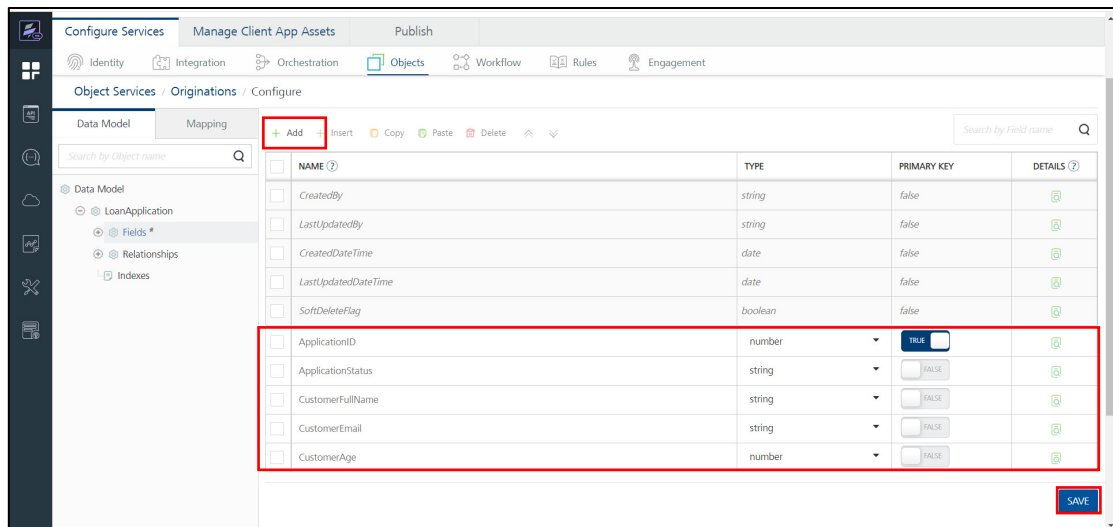
- LoanApplication** オブジェクトの内部に以下のフィールドを設定して下さい。

フィールド名	Type	PRIMARY KEY
ApplicationID	Number	Yes
ApplicationStatus	String	
CustomerFullName	String	
CustomerEmail	String	
CustomerAge	String	

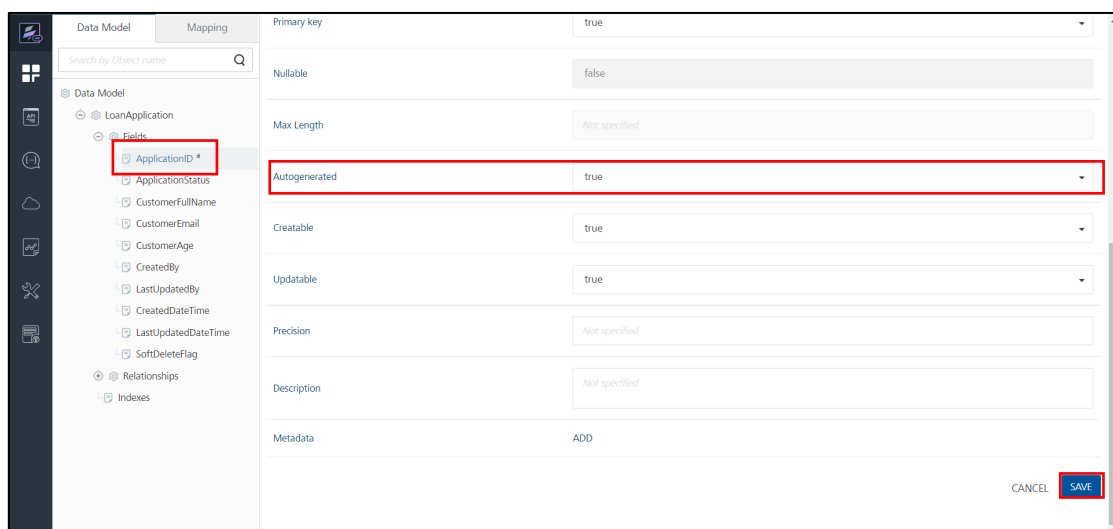
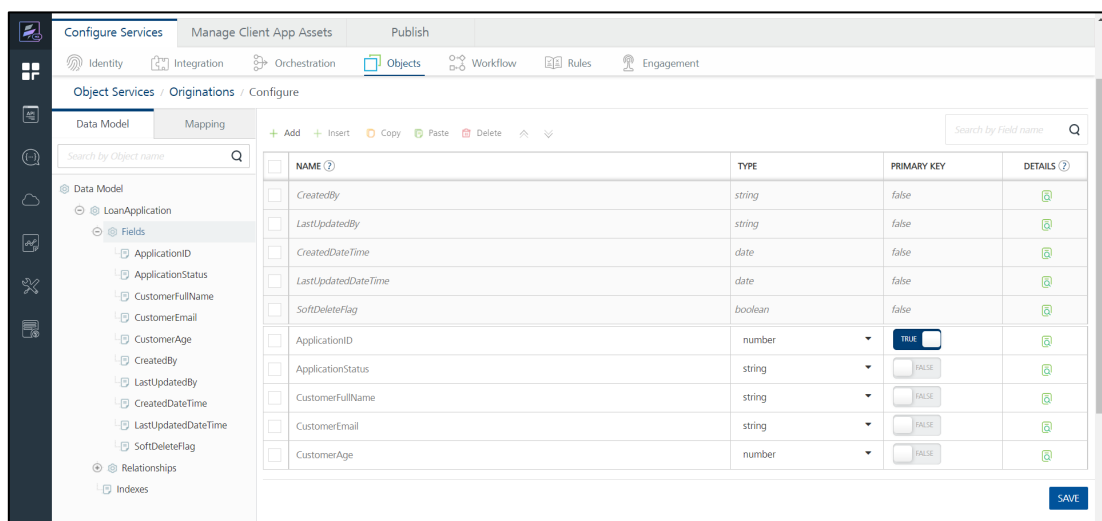
- LoanApplication** を展開し、**Fields** をクリックします。



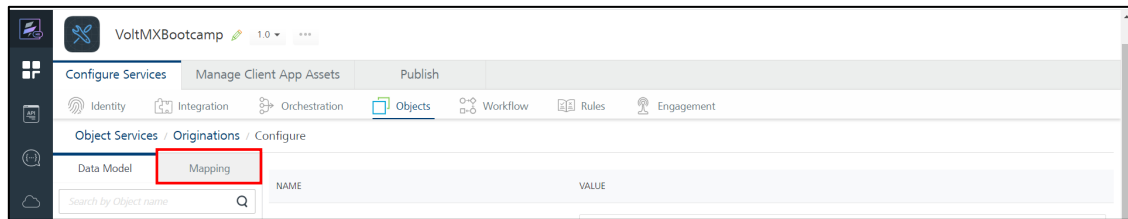
- **Add** をクリックし、上記のフィールドを追加していきます。すべてのフィールドを追加した後、**SAVE** をクリックします。



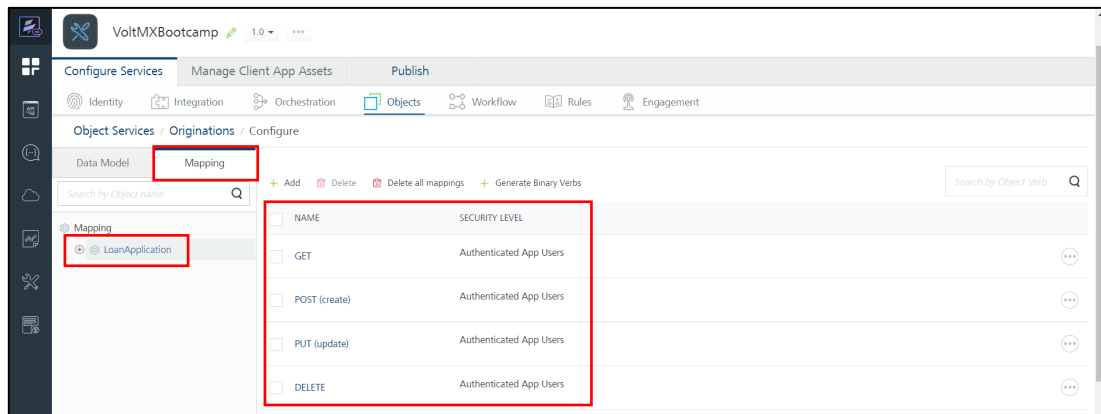
- **Field** を展開し、**ApplicationID** フィールドの **Auto-generated** プロパティを **true** に設定します。**SAVE** をクリックします。



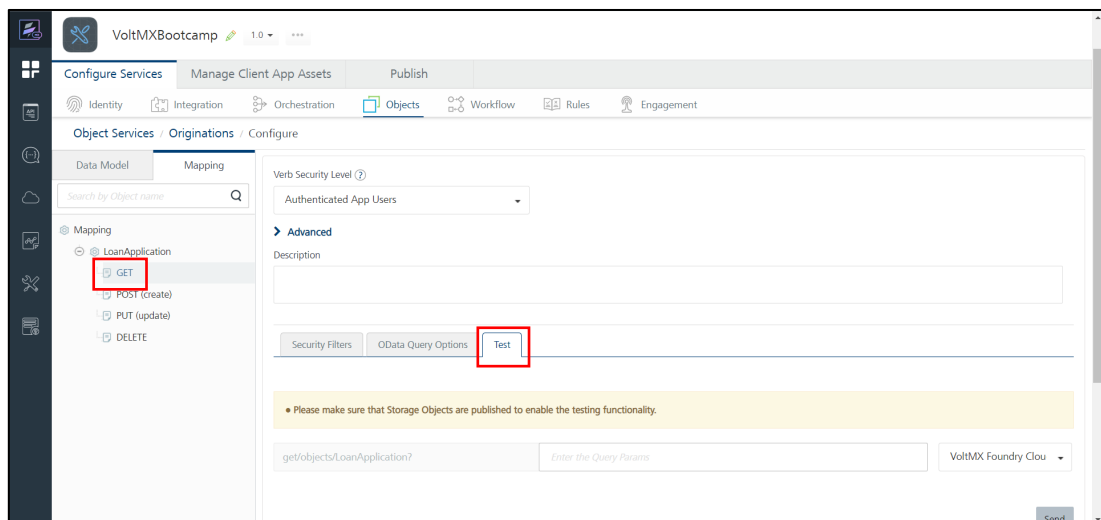
- **Mapping** タブをクリックします。



- **LoanApplication** オブジェクトに生成された CRUD オペレーションが表示されます。

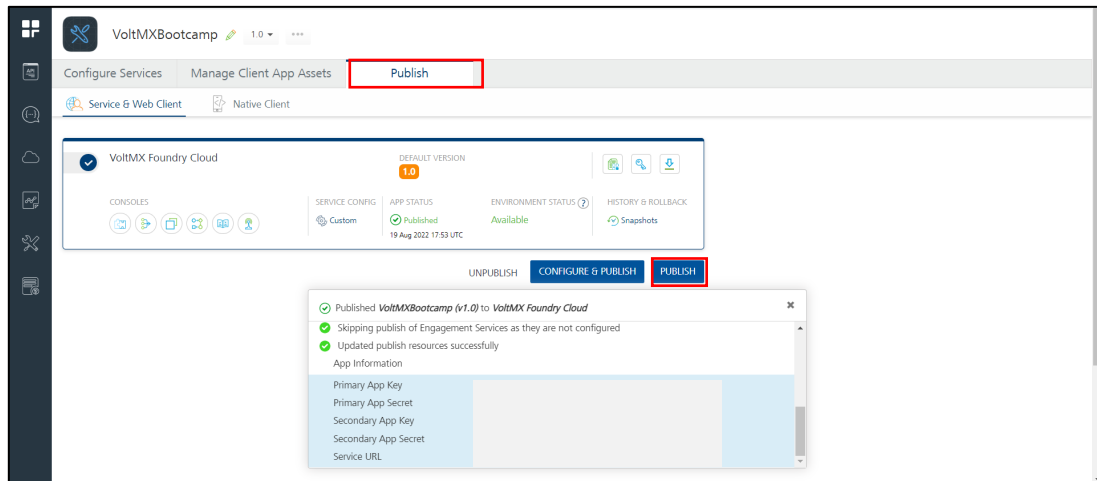


- **GET** オペレーションをクリックし、**TEST** タブをクリックします。

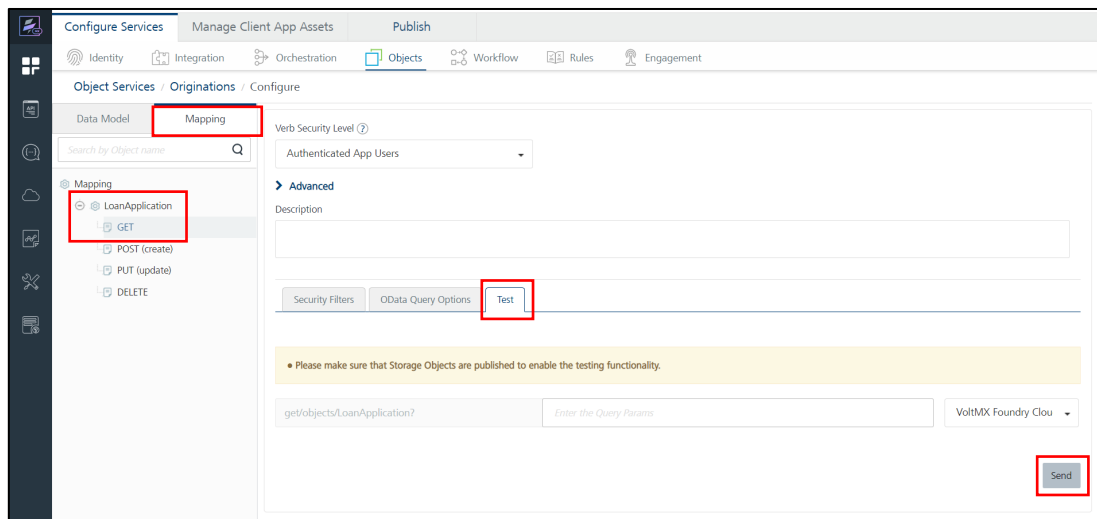


- 「テスト機能を有効にするために、ストレージオブジェクトがパブリッシュされていることを確認してください」というメッセージ (**Please make sure that Storage Objects are published to enable the testing functionality.**) が表示されます。

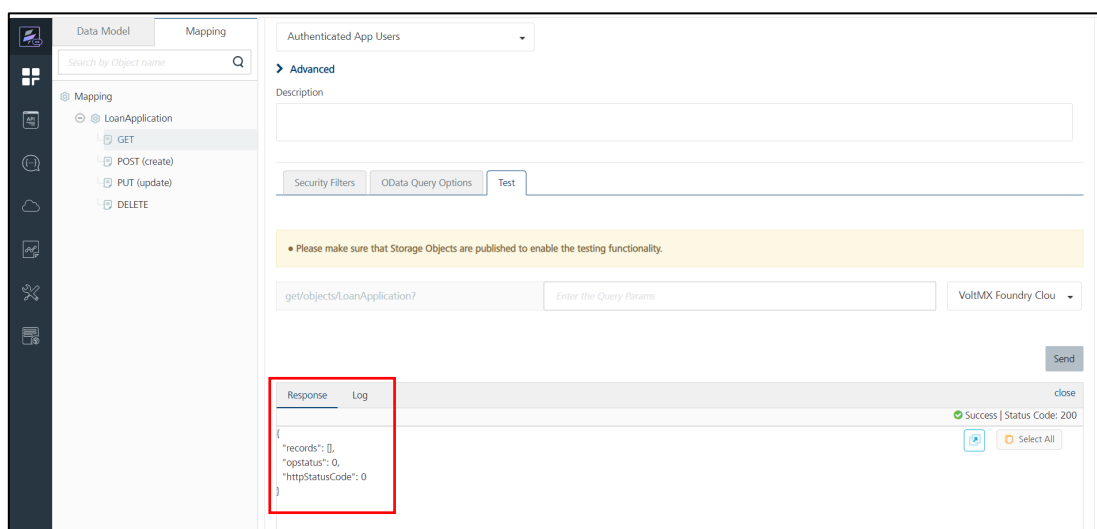
- **Publish** タブに移動し、アプリを公開します。



- **LoanApplication** オブジェクト、**Mapping**、**GET** オペレーション、**Test** タブへ戻ります。

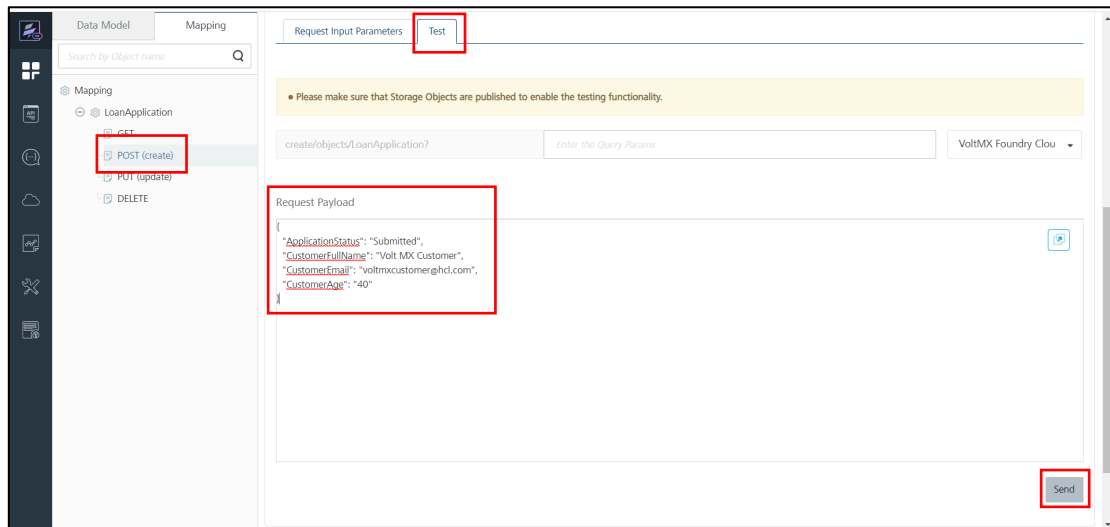


- **SEND** をクリックします。 **LoanApplication** オブジェクトにレコードが存在しないため、**LoanApplication** オブジェクトからレコードがフェッチされないことが分かります。

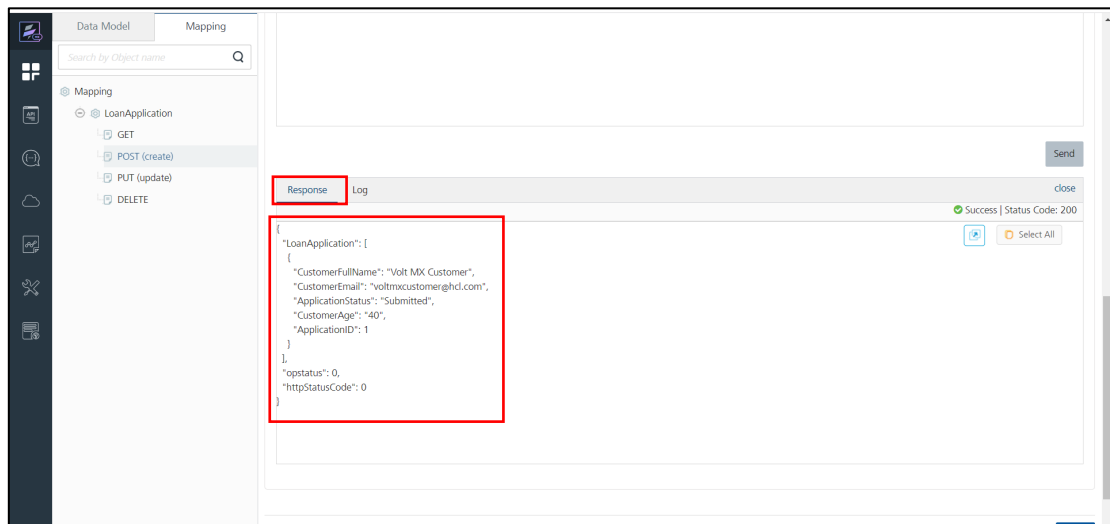


- **LoanApplication** オブジェクト、**Mapping**、**POST (create)**、**Test** タブへ戻ります。
- 以下のペイロードを入力し、**SEND** をクリックします。

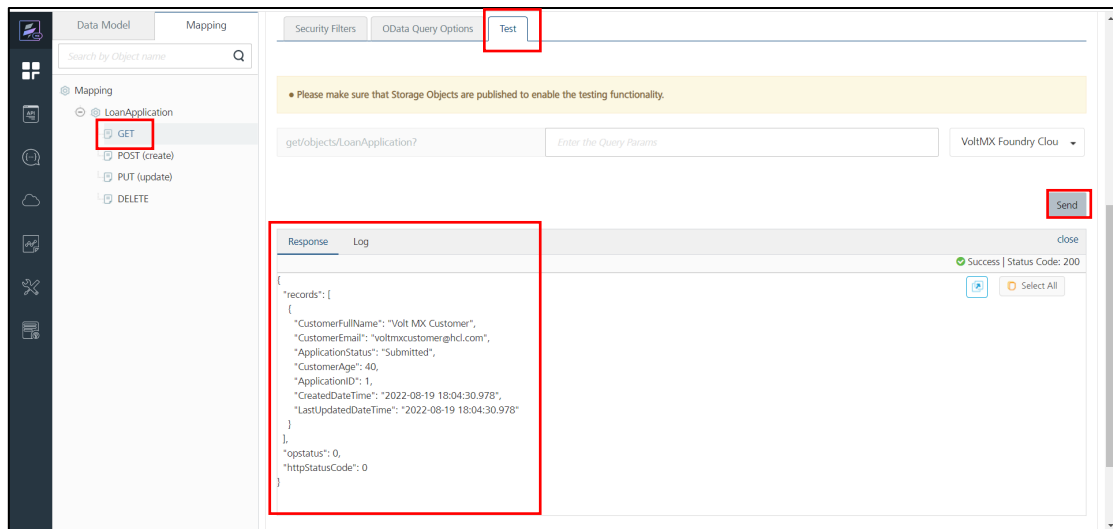
```
{  
  "ApplicationStatus": "Submitted",  
  "CustomerFullName": "Volt MX Customer",  
  "CustomerEmail": "voltmxcustomer@hcl.com",  
  "CustomerAge": "40"  
}
```



- **LoanApplication** オブジェクトにレコードがひとつ作成されたことがわかります。



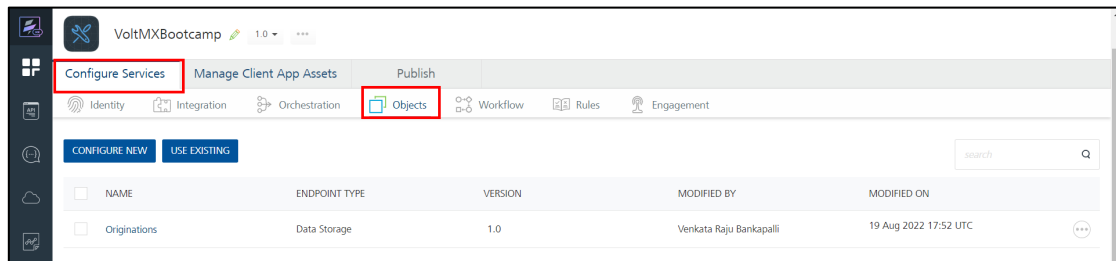
- **LoanApplication** オブジェクト、**Mapping**、**GET** オペレーション、**Test** タブに戻り、**Send** をクリックして GET オペレーションをテストしてください。ひとつのレコードが取得されたことが確認できます。



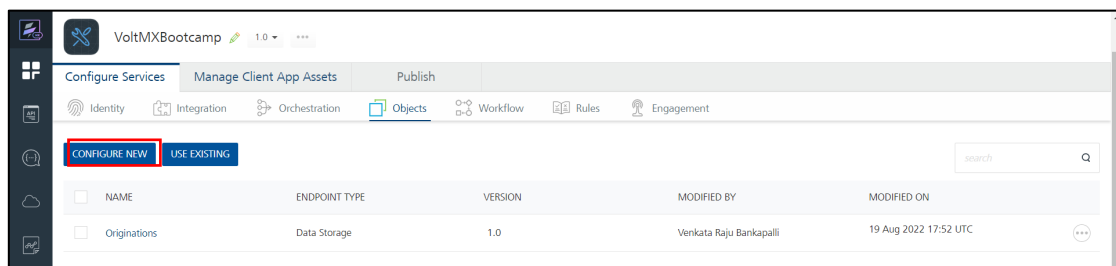
これで1つのオブジェクトでデータストレージオブジェクトサービスを構成することに成功しました。

Foundry で Integration & Orchestration オブジェクトサービスを構成する

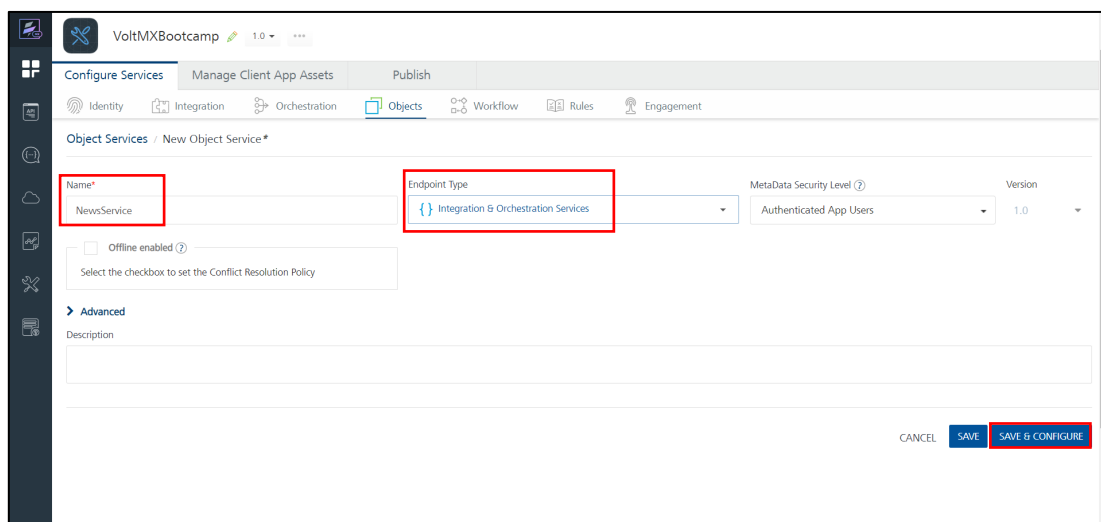
- **Configure Services > Objects** を開きます。



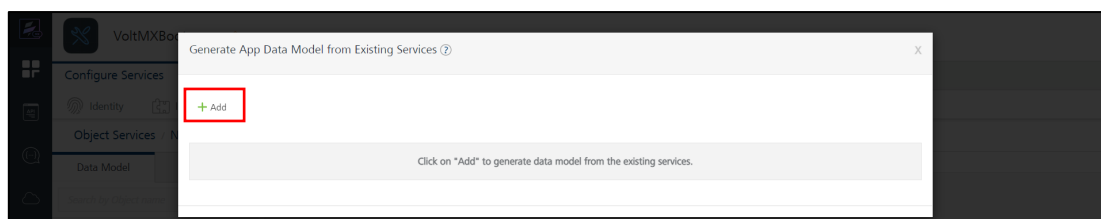
- **CONFIGURE NEW** をクリックします。



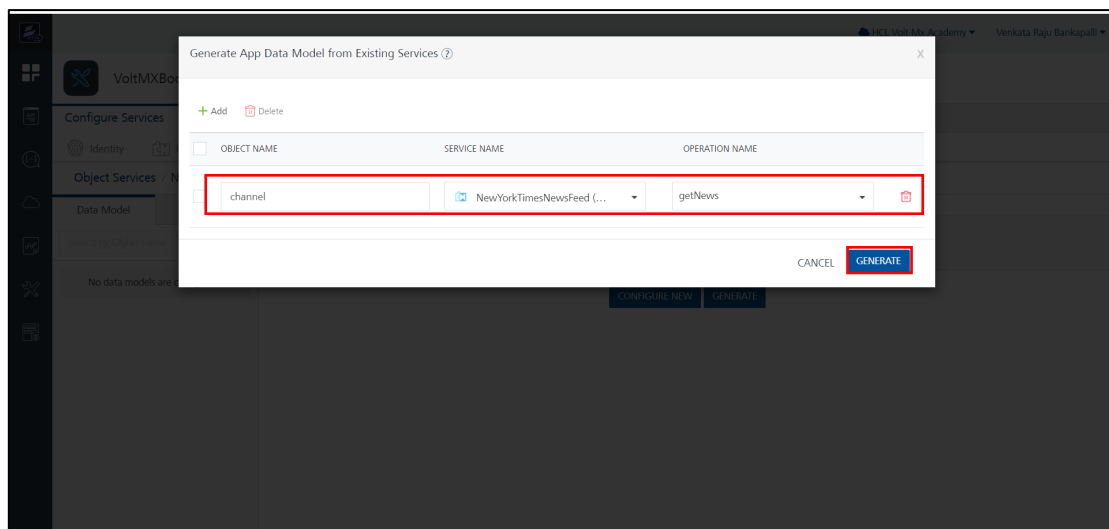
- **Name** を **NewsService**、**Endpoint Type** を **Integration & Orchestration** にします。 **SAVE & CONFIGURE** をクリックします。



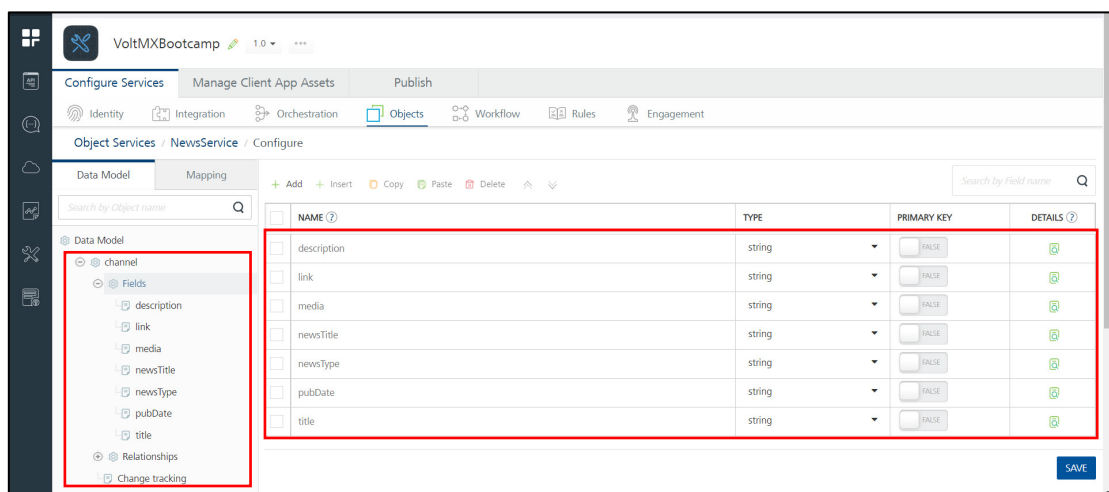
- **NewsService** オブジェクトサービスが作成されます。 **Add** をクリックします。



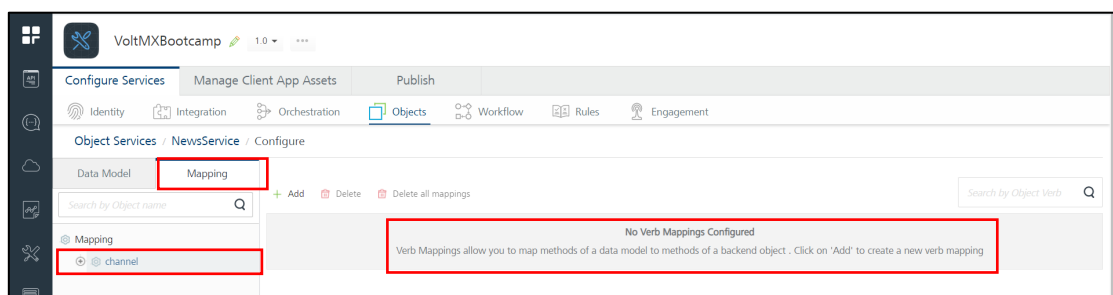
- Object Name を **channel** とし、Service Name を **NewYorkTimesNewsFeed**、Operation Name を **getNews** を選択します。**GENERATE** をクリックします。



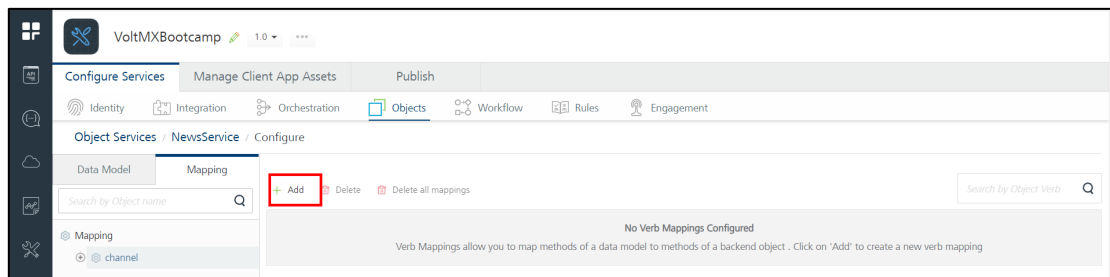
- channel** オブジェクトが作成されることがわかります。このオブジェクトは、**getNews** の統合サービソペレーションで、リクエスト入力と出力パラメータから生成されたフィールドを持っています。



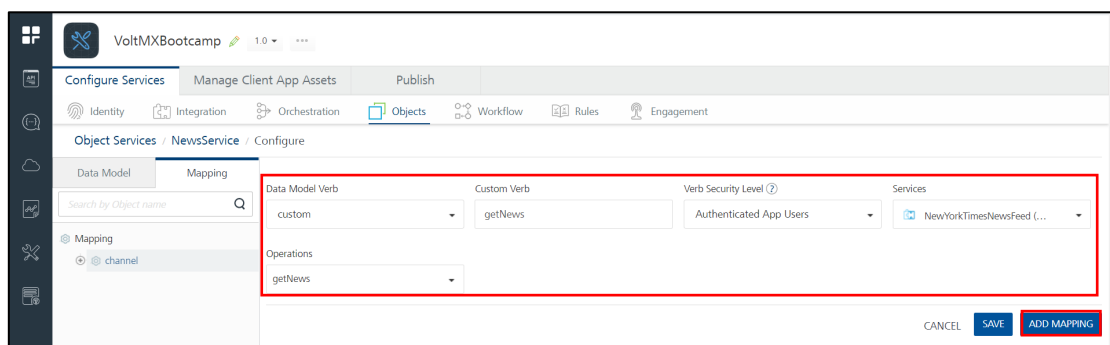
- Mapping** をクリックします。**Integration & Orchestration** タイプのオブジェクトサービスのオブジェクトに対して生成されたマッピングは表示されません。



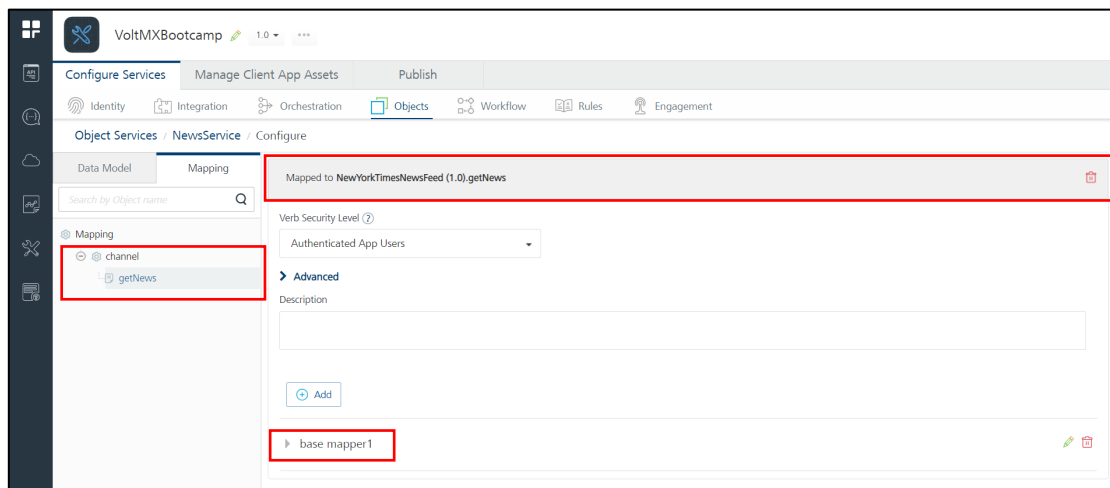
- **Add** をクリックして、新しいマッピングを追加します。



- **Data Model Verb** の値を **custom**、**Custom Verb** の値を **getNews**、**Services** の値を **NewYorkTimesNewsFeed**、**Operation** の値を **getNews** として選択します。 **ADD MAPPING** をクリックします。

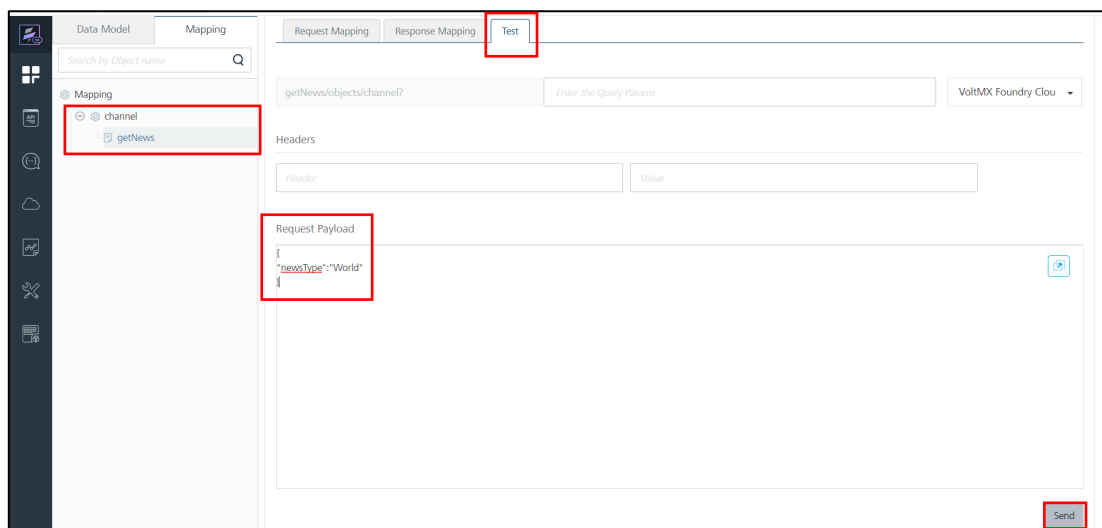


- チャンネルオブジェクトの下に **getNews** の Verb が作成され、**NewYorkTimesNewsFeed** 統合サービスの **getNews** 操作にマップされていることがわかります。

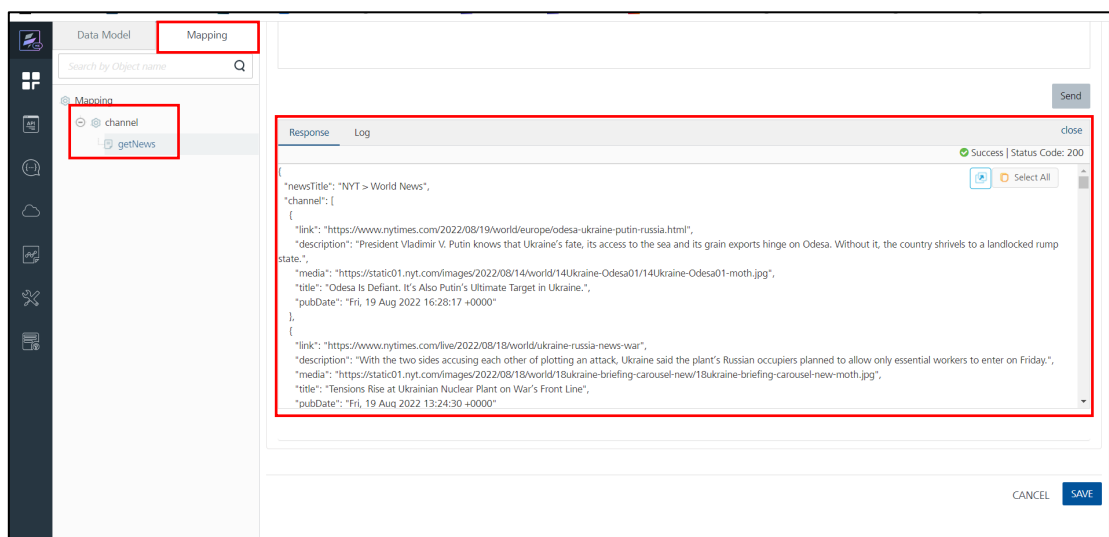


- **base mapper1** を展開し、**Test** タブに移動し、**SEND** をクリックして、以下をリクエストペイロードとして設定します。

```
{
  "newsType": "World"
}
```



- レスポンスを確認します。



- Foundry アプリケーションを公開します。

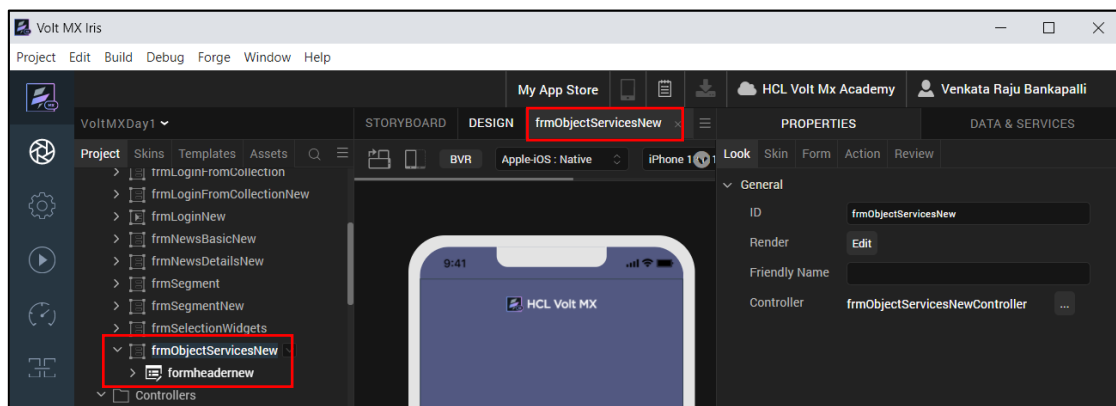
1つのオブジェクトで **Integration & Orchestration** タイプのオブジェクトサービスを構成することに成功しました。

Iris からオブジェクトサービス呼び出す

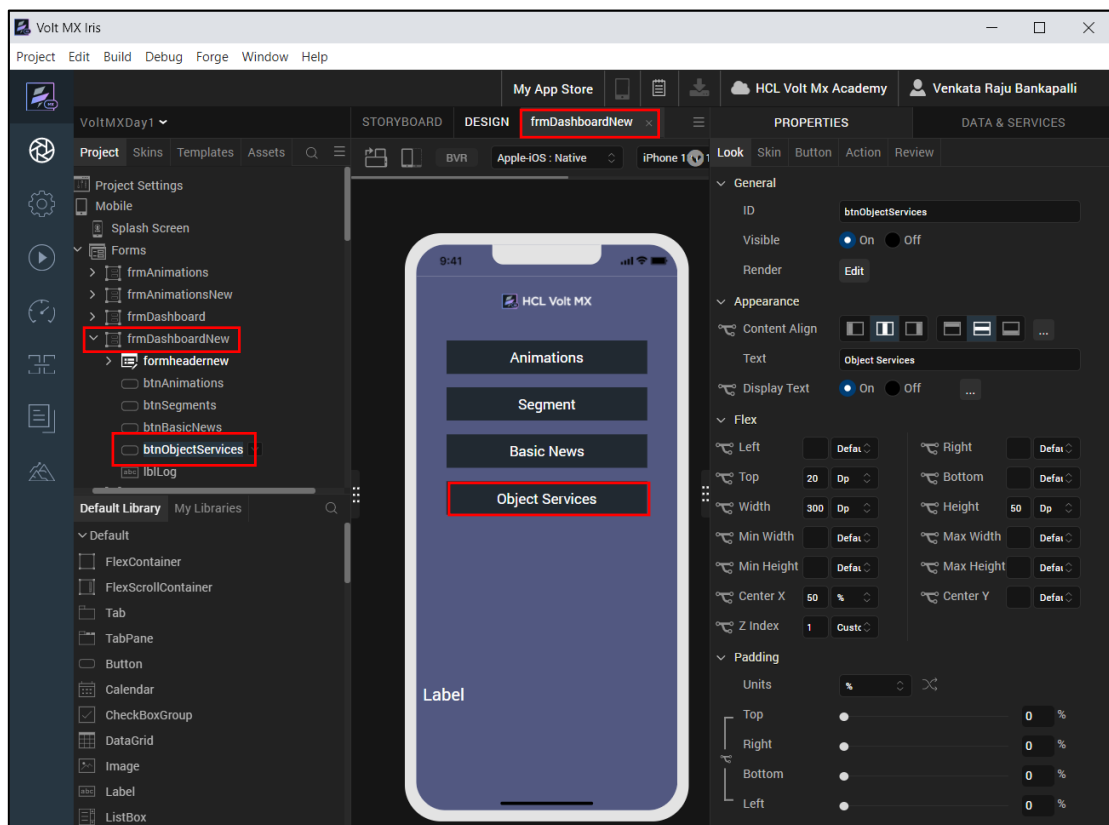
- Iris を開きます。

注意事項

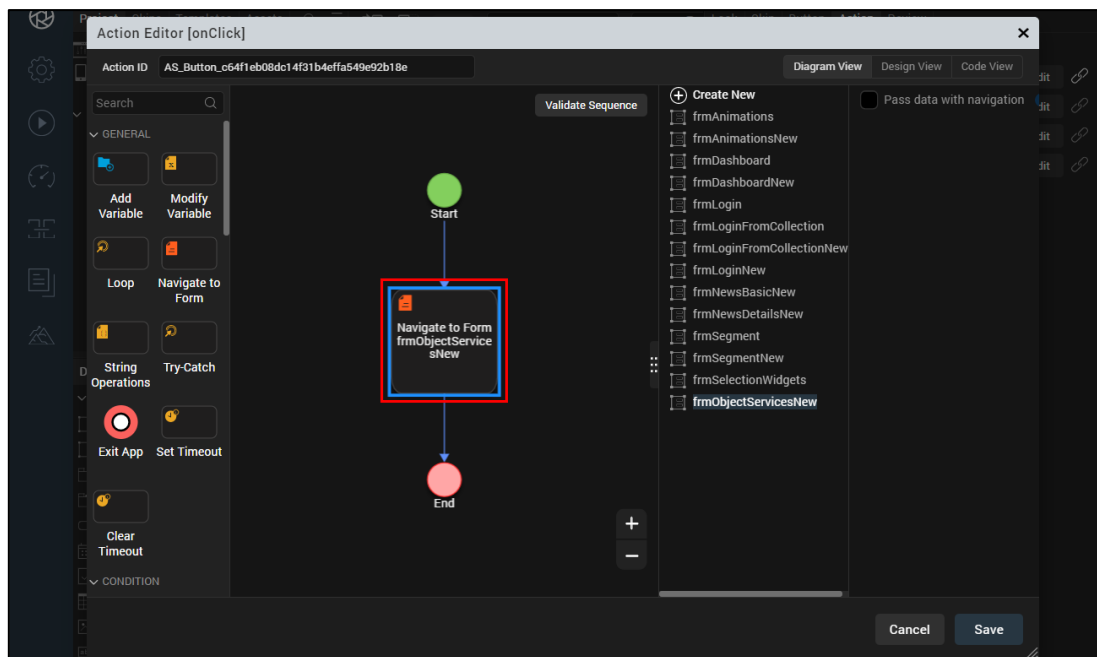
- このドキュメントで説明されている手順を、前のレッスンで作業していた Iris プロジェクトで引き続き実行してください。
- 前のレッスンで作業しているプロジェクトを開きます。
- Mobile チャネルに新しいフォームを作成し、名前を **frmObjectServicesNew** に変更します。
- **frmObjectServicesNew** フォームに **formheadernew** コンポーネントを追加します。



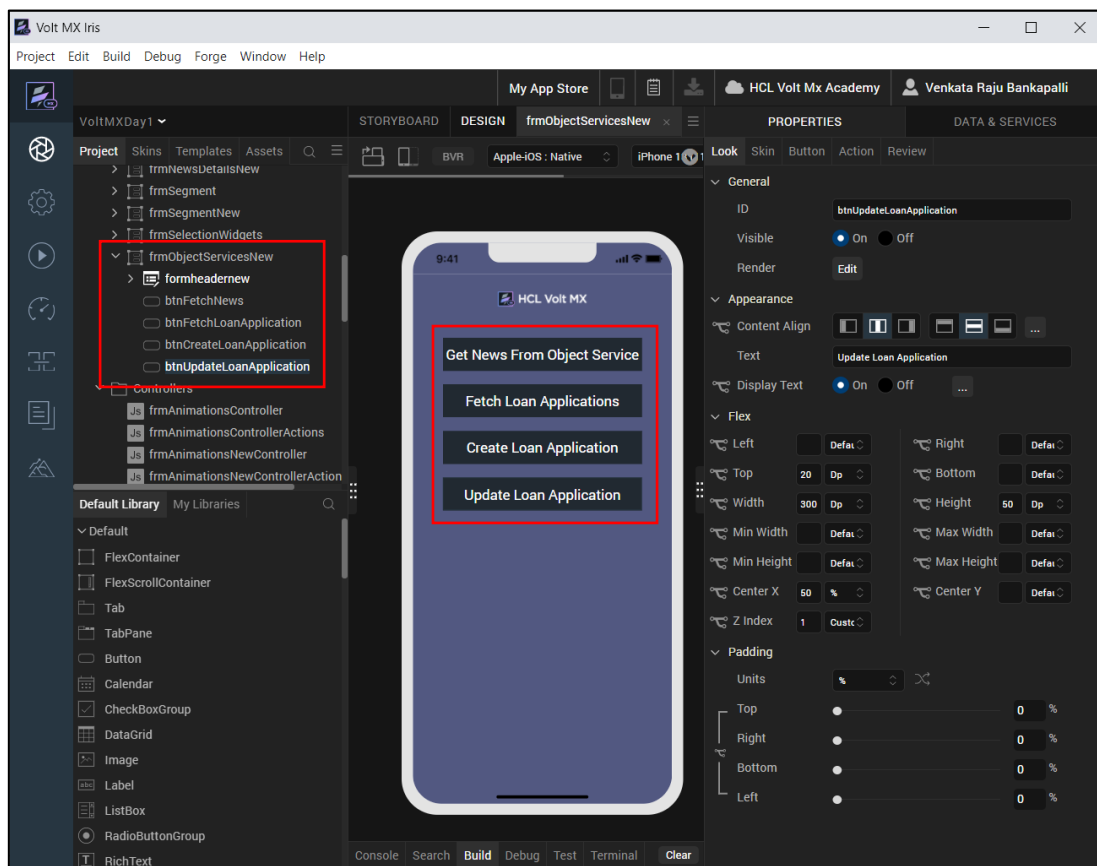
- frmDashboardNew フォームに btnObjectServices という名前で新しいボタンを追加します



- このボタンに対して、**Properties > Action > onClick > Edit** で onClick イベントを設定して、新しいフォーム **frmObjectServicesNew** に移動するようにします。



- 新しいフォーム **frmObjectServicesNew** に **btnFetchNews** , **btnFetchLoanApplication** , **btnCreateLoanApplication** , **btnUpdateLoanApplication** の 4 つのボタンを追加します。



- フォーム **frmObjectServicesNew** のコントローラに移動して、以下のコード・スニペットを追加します。

```
/**
 * Integration や Orchestraion などのオブジェクトサービスからニュースを取得する機能です。
 */
fetchNewsFromObject : function () {
    voltmx.print ("Entering into fetchNesFromObject");

    var objSvc = voltmx.sdk.getCurrentInstance().getSystemService("NewsService", {"access":"online"});
    var dataObject = new voltmx.sdk.dto.DataObject("channel");
    dataObject.addField("newsType","World");
    var options = {"dataObject":dataObject};
    voltmx.application.showLoadingScreen(null, "Fetching News ...",
    constants.LOADING_SCREEN_POSITION_ONLY_CENTER, true, true, null);
    objSvc.customVerb("getNews",
        options,
        function(res){voltmx.application.dismissLoadingScreen(); alert("record::" + JSON.stringify(res));},
        function(err){voltmx.application.dismissLoadingScreen(); alert("Failed to fetch:: " + JSON.stringify(err));}
    );

    voltmx.print ("Exiting out of fetchNesFromObject");
},

/**
 * This funtion is responsible to fetch all the loan applications from Data Storage type of object service
 */
fetchLoanApplications : function () {
    voltmx.print ("Entering into fetchLoanApplications");

    var objSvc = voltmx.sdk.getCurrentInstance().getSystemService("Originations", {"access":"online"});
    var dataObject = new voltmx.sdk.dto.DataObject("LoanApplication");
    var options = {"dataObject":dataObject};
    voltmx.application.showLoadingScreen(null, "Fetching LoanApplications ...",
    constants.LOADING_SCREEN_POSITION_ONLY_CENTER, true, true, null);
    objSvc.fetch(options,
        function(res){voltmx.application.dismissLoadingScreen(); alert("record::" + JSON.stringify(res.records));},
        function(err){voltmx.application.dismissLoadingScreen(); alert("Failed to fetch:: " + JSON.stringify(err));}
    );

    voltmx.print ("Exiting out of fetchLoanApplications");
},

/**
 * This funtion is responsible to create a loan application in the Data Storage type of object service
 */
createLoanApplication : function () {
    voltmx.print ("Entering into createLoanApplication");

    var objSvc = voltmx.sdk.getCurrentInstance().getSystemService("Originations", {"access":"online"});
    var dataObject = new voltmx.sdk.dto.DataObject("LoanApplication");
    dataObject.addField("CustomerFullName","Volt MX Customer 1");
    dataObject.addField("CustomerEmail","voltmxcustomer1@hcl.com");
    dataObject.addField("ApplicationStatus","Submitted");
    dataObject.addField("CustomerAge",40);
    var options = {"dataObject":dataObject};
    voltmx.application.showLoadingScreen(null, "Creating LoanApplication ...",
    constants.LOADING_SCREEN_POSITION_ONLY_CENTER, true, true, null);
```

```

    objSvc.create(options,
        function(res){voltmx.application.dismissLoadingScreen(); alert("Record created:: " +
JSON.stringify(res));},
        function(err){voltmx.application.dismissLoadingScreen(); alert("Error in record creation:: " +
JSON.stringify(err));}
    );

    voltmx.print ("Exiting out of createLoanApplication");
},

/**
 * This funtion is responsible to update an existing loan application in the Data Storage type of object service
 */
updateLoanApplication : function () {
    voltmx.print ("Entering into updateLoanApplication");

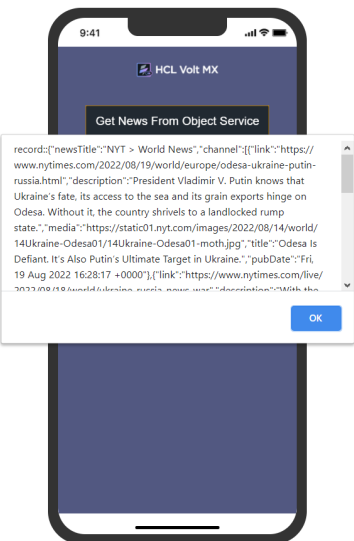
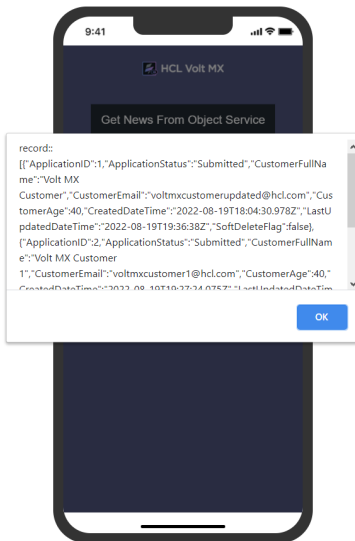
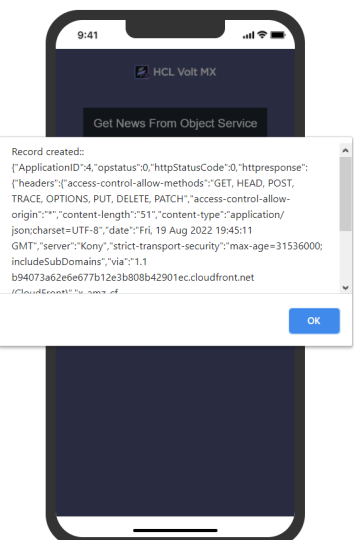
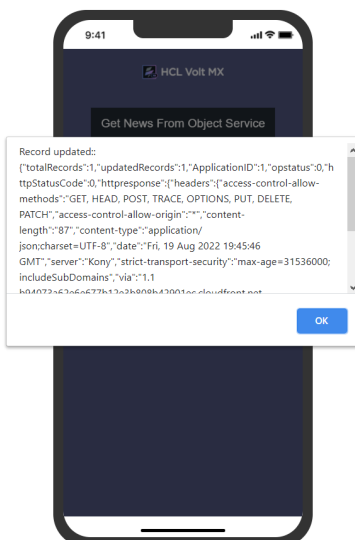
    var objSvc = voltmx.sdk.getCurrentInstance().getSystemService("Originations", {"access":"online"});
    var dataObject = new voltmx.sdk.dto.DataObject("LoanApplication");
    dataObject.addField("CustomerEmail", "voltmxcustomerupdated@hcl.com");
    dataObject.addField("ApplicationID", 1);
    var options = {"dataObject":dataObject};
    voltmx.application.showLoadingScreen(null, "Updating LoanApplcation Record ...",
constants.LOADING_SCREEN_POSITION_ONLY_CENTER, true, true, null);
    objSvc.update(options,
        function(res){voltmx.application.dismissLoadingScreen(); alert("Record updated:: " +
JSON.stringify(res));},
        function(err){voltmx.application.dismissLoadingScreen(); alert("Error in record update:: " +
JSON.stringify(err));}
    );

    voltmx.print ("Exiting out of updateLoanApplication");
}

```

- アクションエディタで **Invoke Function** アクションを使い、**btnFetchNews** ボタンの **onClick** イベントで **fetchNewsFromObject** 関数を呼び出します。
- アクションエディターの **Invoke Function** アクションを使って、**btnFetchLoanApplication** ボタンの **onClick** イベントで **fetchLoanApplications** ファンクションを呼び出します。
- アクションエディタの **Invoke Function** アクションを使用して、**btnCreateLoanApplication** ボタンの **onClick** イベントで **createLoanApplication** 関数を呼び出します。
- アクションエディターの **Invoke Function** アクションを使い、**btnUpdateLoanApplication** ボタンの **onClick** イベントで **updateLoanApplication** 関数を呼び出します。

- **Build > Run Live Preview** メニューでライブプレビュービルドを生成し、テストします。

Integration & Orchestration タイプの オブジェクトサービスを使用したニュースの取得	Data Storage タイプのオブジェクトでのフェッチ操作
 <p>The screenshot shows a mobile app interface with a button labeled "Get News From Object Service". A modal window displays a JSON record containing news details from NYTimes, including title, channel, description, media, and publication date.</p>	 <p>The screenshot shows the same app interface. The modal window displays a JSON record with application metadata, including ApplicationID, ApplicationStatus, CustomerFullName, CustomerEmail, and timestamps for creation and update.</p>
オブジェクトのデータストレージタイプで作成操作	オブジェクトの Data Storage タイプの Update 操作
 <p>The screenshot shows the app interface. The modal window displays a JSON record created in Data Storage, including ApplicationID, opstatus, httpStatusCode, httpresponse, headers, content-length, content-type, application/jsoncharset, date, GMT, server, strict-transport-security, max-age, includeSubDomains, and via.</p>	 <p>The screenshot shows the app interface. The modal window displays a JSON record updated in Data Storage, including totalRecords, updatedRecords, ApplicationID, opstatus, httpStatusCode, httpresponse, headers, content-length, content-type, application/jsoncharset, date, GMT, server, strict-transport-security, max-age, includeSubDomains, and via.</p>

注：ライブプレビューの設定で以前に選択されたチャンネル/プラットフォーム/アプリケーションの種類はそのまま残ります。そして、ライブプレビュービルドは、これらの選択されたチャンネル/プラットフォーム/アプリケーションの種類で生成されます。

おめでとうございます。あなたはこのレッスンのハンズオンを完了しました。