

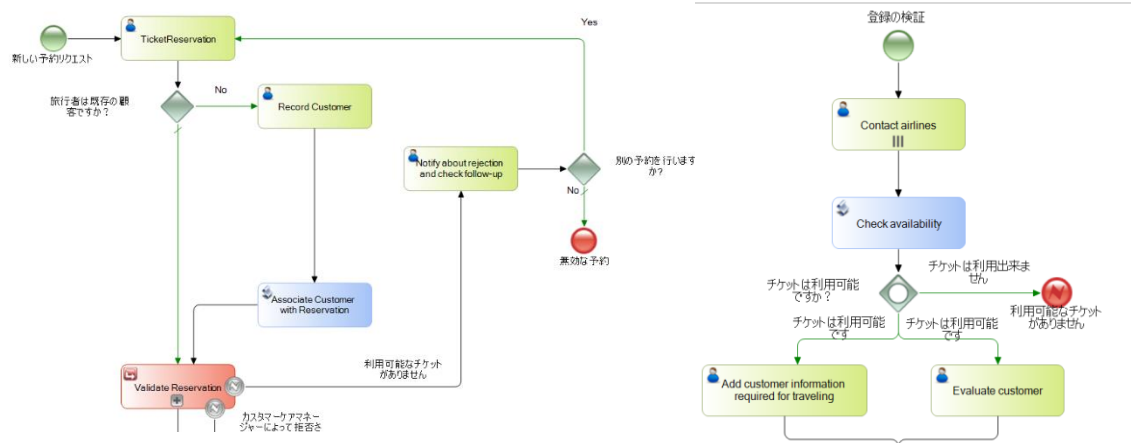
Workflow API を使用して GeneXus オブジェクトからプロセスを開始する

旅行代理店はチケット予約のプロセスを変更し、いくつかの新しい機能を追加することを決定しました。

まず、代理店の顧客がウェブサイトを通じて独自の予約を入力できるようにすることを要求しました。

顧客はサイトにログインし、予約データ（日付、出発空港、到着空港、乗客数など）を提供する必要があります。これにより、システムは予約を作成し、「FlightTicketReservation」プロセスを開始して、後で「TicketReservation」タスクが完了したことを示すことができます。

乗客がすでに旅行代理店の顧客である場合、「TicketReservation」タスクが完了すると、「ValidateReservation」サブプロセスの最初のタスクである「ContactAirlines」が実行されます。これはマルチインスタンスタスクです。



システムは、航空会社に連絡できる各人に、「ContactAirlines」タスクのインスタンスが実行可能であることを通知する必要があります。最後に、予約が承認されたことを顧客に通知するタスクタイプを置き換えて、通知が直接行われるようにします。

まず、顧客がウェブサイトですべての予約を行うところから始めます。

そのために、「TravelAgency」という WebPanel を使用します。この WebPanel には、実行ボタンに加えて、ユーザーが予約データを入力するための画面上の変数が含まれています。

TravelAgency * X

Web Layout * Rules Events Events (without WorkWithPlus code) Conditions Variables

<選択されているアクショングループはありません>

Customer name: &CustomerName

Flight ticket reservation

Please enter you reservation information

Date &ReservationDate

Passengers Qty &ReservationQty

Departure Airport &ReservationDepartureAirportId

&ReservationDepartureCityName

&ReservationDepartureCountryName

Arrival Airport &ReservationArrivalAirportId

&ReservationArrivalCityName

&ReservationArrivalCountryName

実行

このデモでは、説明を簡略化するために、CustomerId=1 が既にログインしていると仮定します。そのため、ログインコントロールは含めません。Start イベントに移動すると、これをシミュレートするコードが表示されます。

```
1 Event Start
2     &CustomerId = 1 // Hardcoded only for demo uses
3     &CustomerEmail = GetLoggedInUser (&CustomerId)
4     &CustomerName = GetCustomerName (&CustomerId)
5 Endevent
```

確認ボタンを押すと以下の Enter イベントが実行され、いくつかのタスクが実行されます。

```
7 Event Enter
8 //Create a new reservation with the entered data
9 &ReservationId = NewReservation(&ReservationDate,&ReservationQty,&CustomerId,
10                                &ReservationDepartureAirportId,&ReservationArrivalAirportId )
11
12 //Create a new FlightTicketReservation process instance
13 &WorkflowServer.Connect("WFADMINISTRATOR","WFADMINISTRATOR")
14 &WorkflowProcessDefinition = &WorkflowServer.GetProcessDefinitionByName("FlightTicketReservation")
15 &WorkflowProcessInstance = &WorkflowProcessDefinition.CreateInstance()
16 &WorkflowProcessInstance.Subject = 'FlightTicketReservation process started from GeneXus Menu'
17 &ReservationIdWorkflowApplicationData = &WorkflowProcessInstance.GetApplicationDataByName("ReservationId")
18 &ReservationIdWorkflowApplicationData.NumericValue = &ReservationId
19 &WorkflowProcessInstance.Start() // Initiate the FlightTicketReservation process instance
20
21 //Mark the TicketReservation task as completed
22 &WorkflowActivity = &WorkflowProcessDefinition.GetActivityByName('TicketReservation')
23 &WorkflowWorkitem = &WorkflowProcessInstance.GetWorkitemByActivity(&WorkflowActivity)
24 &WorkflowWorkitem.Complete()
25 Commit
26 Endevent
```

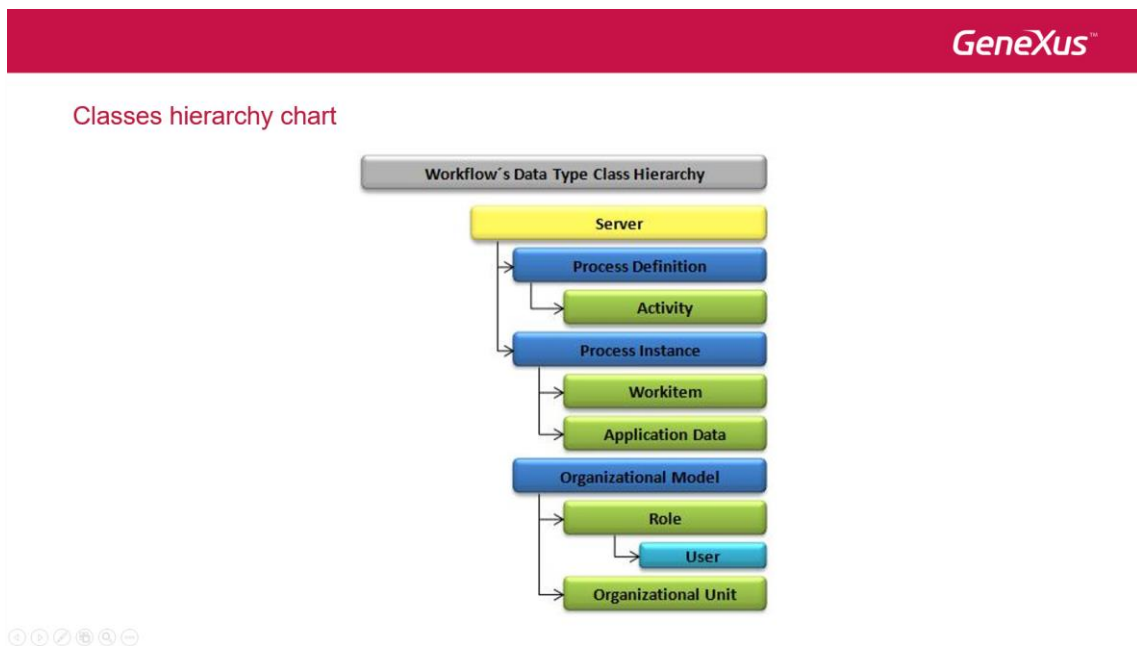
まず、プロシージャ「NewReservation」を呼び出してデータベースに予約を登録します。次に、「Workflow」データタイプを使用して、「FlightTicketReservation」プロセスが開始され、「TicketReservation」タスクが完了としてマークされます。

接頭辞が「Workflow」で始まるこれらのデータタイプは、アプリケーションがワークフローエンジンと対話できるようにするデータタイプです。

「Workflow」データタイプを使用してエンジンの API を利用し対話するには、まずそのデータタイプの変数を定義する必要があります。次に、コンテキスト情報を使用して、使用するメソッドまたはプロパティを選択できます。

```
7 Event Enter
8 //Create a new reservation with the entered data
9 &ReservationId = NewReservation(&ReservationDate,&ReservationQty,&CustomerId,
10                                &ReservationDepartureAirportId,&ReservationArrivalAirportId )
11
12 //Create a new FlightTicketReservation process instance
13 &WorkflowServer.Connect("WFADMINISTRATOR","WFADMINISTRATOR")
14 &WorkflowProcessDefinition = &WorkflowServer.GetProcessDefinitionByName("FlightTicketReservation")
15 &WorkflowProcessInstance = &WorkflowProcessDefinition.CreateInstance()
16 &WorkflowProcessInstance.Subject = 'FlightTicketReservation process started from GeneXus Menu'
17 &ReservationIdWorkflowApplicationData = &WorkflowProcessInstance.GetApplicationDataByName("ReservationId")
18 &ReservationIdWorkflowApplicationData.NumericValue = &ReservationId
19 &WorkflowProcessInstance.Start() // Initiate the FlightTicketReservation process instance
20
21 //Mark the TicketReservation task as completed
22 &WorkflowActivity = &WorkflowProcessDefinition.GetActivityByName('TicketReservation')
23 &WorkflowWorkitem = &WorkflowProcessInstance.GetWorkitemByActivity(&WorkflowActivity)
24 &WorkflowWorkitem.Complete()
25 Commit
26 Endevent
```

「Workflow」データタイプはクラスの階層に分類されます。



最上位クラスはサーバークラスで、このクラスには他の 3 つのクラスが依存しています。

「ProcessDefinition」クラスではプロセスダイアグラムのコンポーネントにアクセスできます。

「ProcessInstance」では実行中のプロセスのインスタンスにアクセスできます。

「OrganizationalModel」クラスでは、役割やユーザーなど、会社の組織構造に関する情報にアクセスできます。

「Server」クラスは型階層のエントリポイントであり、そのメソッドにより任意の「Workflow」データタイプにアクセスできます。

頻繁に使用されるデータタイプは以下です。

GeneXus™

Most used Workflow Data Types

- WorkflowProcessDefinition
- WorkflowProcessInstance
- WorkflowWorkItem
- WorkflowContext
- WorkflowApplicationData

「WorkflowProcessDefinition」を使用すると、名前、バージョン、そこに含まれるタスク（アクティビティと呼びます）などのダイアグラムのいくつかのプロパティにアクセスできます。また、その定義に基づいて、「CreateInstance」メソッドを使用してプロセスの実行インスタンスを作成することもできます。

GeneXus™

Most used Workflow Data Types

- WorkflowProcessDefinition
 - ✓ Diagram name
 - ✓ Diagram version
 - ✓ Activities (tasks)
 - ✓ Create Instance
- WorkflowProcessInstance
- WorkflowWorkItem
- WorkflowContext
- WorkflowApplicationData

「WorkflowProcessInstance」を使用すると、インスタンスのベースとなるプロセスの定義、インスタンスが処理する問題、インスタンスの一部である作業項目のコレクションを見つけることができます。

「GetApplicationByName」メソッドを通じて、名前を使用して関連データを取得できます。

GeneXus™

Most used Workflow Data Types

- WorkflowProcessDefinition
- WorkflowProcessInstance
 - ✓ Process Definition
 - ✓ Subject
 - ✓ Workitems
 - ✓ [GetApplicationByName](#)
- WorkflowWorkItem
- WorkflowContext
- WorkflowApplicationData

「WorkflowWorkItem」クラスを使用すると、プロセスインスタンス内のアクティビティのコンテキストで参加者が実行する必要がある作業を知ることができます。

「ProcessInstance」は、作業項目が属するプロセスインスタンスに関する情報を提供し、「Activity」プロパティは作業項目から生成したアクティビティを返します。

「Assign」メソッドを使用すると、特定のユーザーに作業項目を割り当てることができ、「Complete」メソッドを使用すると、作業項目の実行を終了することができます。

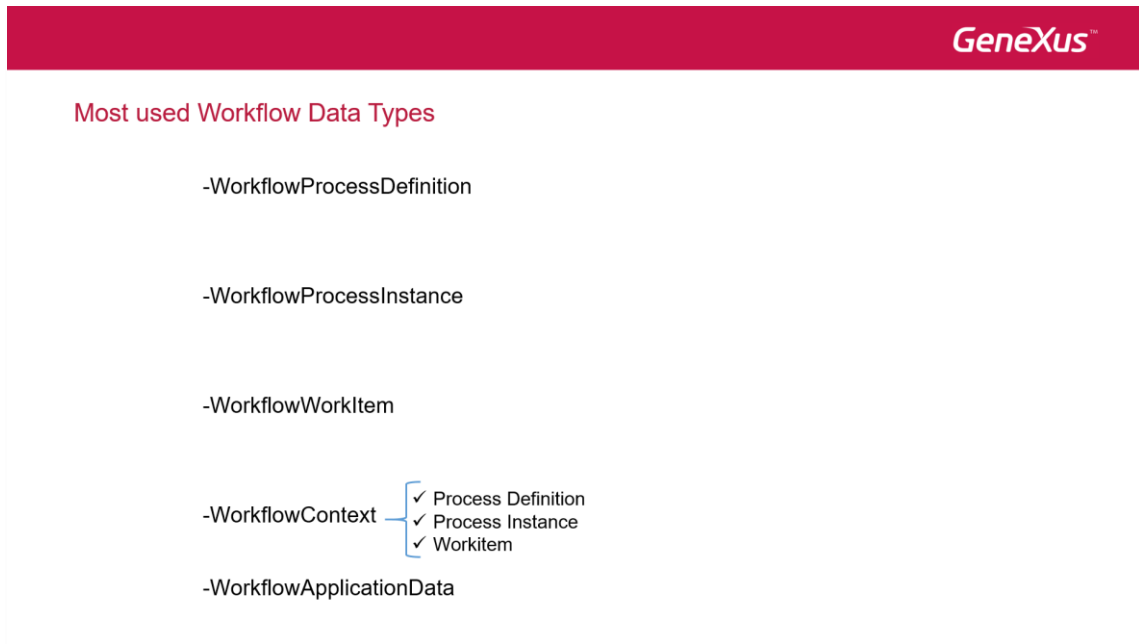
GeneXus™

Most used Workflow Data Types

- WorkflowProcessDefinition
- WorkflowProcessInstance
- WorkflowWorkItem
 - ✓ Process instance
 - ✓ Activity
 - ✓ **Assign**
 - ✓ **Complete**
- WorkflowContext
- WorkflowApplicationData

「WorkflowContext」データタイプは、アクティビティに関連付けられたアプリケーションの実行コンテキストに関する情報を提供します。このコンテキストは、アクティビティに関連付けられたアプリケーション（つまり、タスク）がプロセスダイアグラムを含む同じ KB の一部である GeneXus オブジェクトである場合に自動的にインスタンス化されます。

このコンテキストの自動インスタンス化により、プロセス定義、プロセスのインスタンス、およびアクティビティに関連付けられた作業項目の値を知ることができます。



最後に、「WorkflowApplicationData」データタイプは、「GetApplicationDataByName」メソッドを通じて取得した関連データを保存する場合など、関連データを扱うときに使用するデータ型です。

GeneXus™

Most used Workflow Data Types

- WorkflowProcessDefinition
- WorkflowProcessInstance
- WorkflowWorkItem
- WorkflowContext
- WorkflowApplicationData — ✓ Relevant datas

ここで、「FlightTicketReservation」プロセスを呼び出す WebPanel のイベントに戻ります。ここでは、「WorkflowServer」タイプの定義済み変数「&WorkflowServer」があります。実際には、識別しやすいように、常に「Workflow」データタイプに一致する変数名を使用します。

「WorkflowServer」データタイプで実行する最初の操作は、管理者パスワードを使用してワークフローエンジンに接続することです。

```
//Create a new FlightTicketReservation process instance
&WorkflowServer.Connect("WFADMINISTRATOR", "WFADMINISTRATOR")
&WorkflowProcessDefinition = &WorkflowServer.GetProcessDefinitionByName("FlightTicketReservation")
&WorkflowProcessInstance = &WorkflowProcessDefinition.CreateInstance()
&WorkflowProcessInstance.Subject = 'FlightTicketReservation process started from GeneXus Menu'
&ReservationIdWorkflowApplicationData = &WorkflowProcessInstance.GetApplicationDataByName("ReservationId")
&ReservationIdWorkflowApplicationData.NumericValue = &ReservationId
&WorkflowProcessInstance.Start() // Initiate the FlightTicketReservation process instance
```

次に、「FlightTicketReservation」プロセスの名前に基づいて定義を取得し、それを「WorkflowProcessDefinition」型の変数に保存します。

定義ができたなら、「CreateInstance」メソッドを使用してプロセスにインスタンスを作成します。次に、入力トレイでプロセスを簡単に認識できるように Subject を変更します。

その後、以前に作成した予約識別子を使用して関連データ「 ReservationId」 をロードし、Start()メソッドでインスタンスを開始します。

以下のコード行は、「TicketReservation」タスクを完了としてマークするために使用されます。

```
//Mark the TicketReservation task as completed
&WorkflowActivity = &WorkflowProcessDefinition.GetActivityByName('TicketReservation')
&WorkflowWorkitem = &WorkflowProcessInstance.GetWorkitemByActivity(&WorkflowActivity)
&WorkflowWorkitem.Complete()
Commit
```

まず、プロセスの定義から「TicketReservation」アクティビティを取得し、そのアクティビティを使用して、プロセスで実行中のタスクに対応する作業項目を取得します。次に、作業項目（タスク）を完了としてマークします。

Complete() メソッドの後にコミットがあることに注目してください。「Workflow」データタイプを使用して行われた変更は、アプリケーションのロジック作業単位に含まれます。

ただし、ワークフロー操作はコミットを実行しないため、アプリケーションで UTL を正しく定義し、最後にコミットを実行する必要があります。この場合、WebPanel でワークフロー操作を実行したため、完了したらコミットを追加する必要があります。

ここで確認したこれらの「Workflow」データタイプは、使用可能なすべてのデータのサブグループであり、ワークフローエンジンの API を介してコードで多くのタスクを実行できます。

