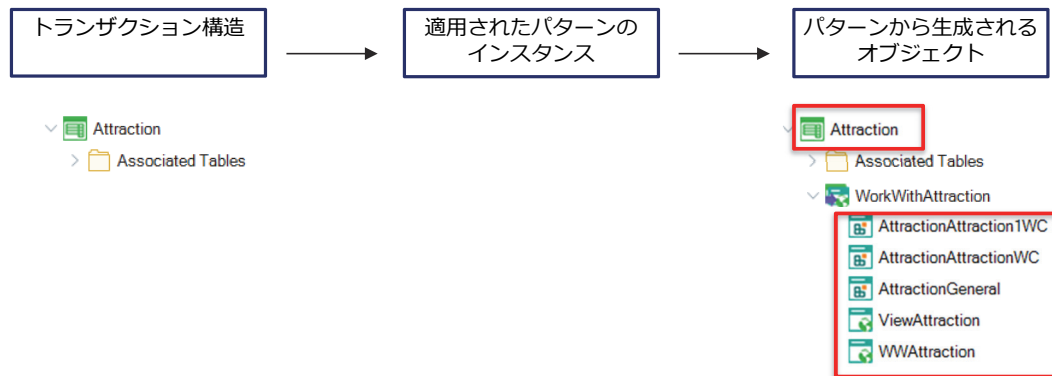
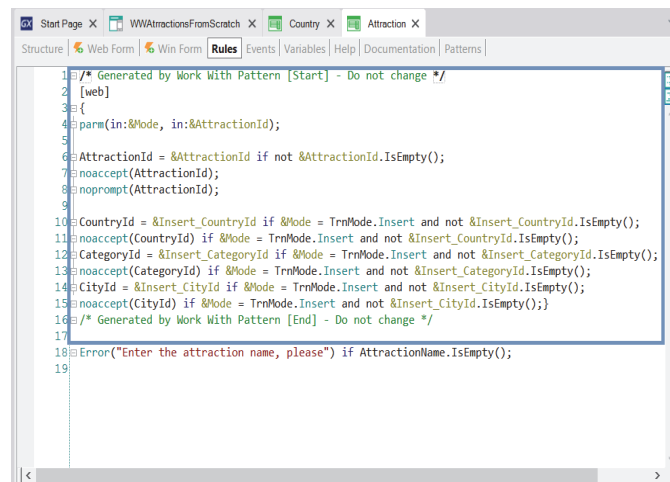


【補足】 パターンによるトランザクショ
ンへのルール追加



トランザクションにパターンを適用すると、パターンによって提供される新しい画面を実装するオブジェクトが生成されるだけでなく、新しいインタラクションのフローに従えるようにトランザクションの動作も変更されます。



```

1 /* Generated by Work With Pattern [Start] - Do not change */
2 [web]
3 {
4   parm(in:&Mode, in:&AttractionId);
5
6   AttractionId = &AttractionId if not &AttractionId.IsEmpty();
7   noaccept(AttractionId);
8   noprompt(AttractionId);
9
10  CountryId = &Insert_CountryId if &Mode = TrnMode.Insert and not &Insert_CountryId.IsEmpty();
11  noaccept(CountryId) if &Mode = TrnMode.Insert and not &Insert_CountryId.IsEmpty();
12  CategoryId = &Insert_CategoryId if &Mode = TrnMode.Insert and not &Insert_CategoryId.IsEmpty();
13  noaccept(CategoryId) if &Mode = TrnMode.Insert and not &Insert_CategoryId.IsEmpty();
14  CityId = &Insert_CityId if &Mode = TrnMode.Insert and not &Insert_CityId.IsEmpty();
15  noaccept(CityId) if &Mode = TrnMode.Insert and not &Insert_CityId.IsEmpty();
16 /* Generated by Work With Pattern [End] - Do not change */
17
18 Error("Enter the attraction name, please") if AttractionName.IsEmpty();
19

```

例として、[Rules] エlementを見てみましょう。一連のルールが追加されていて、これらのルールを変更しないようにという開発者向けのコメントが残されています。そして、前に宣言した Error ルールがあります。このルールは、パターンによって追加されたルールのブロックの外側に、そのまま残されています。

Web パターンによって追加されたコードブロック内の最初のルールは、オブジェクトがその呼び出し元とやり取りするパラメーターを定義するためのルールです。パターンを適用すると、Attraction トランザクションが開発者メニューに表示されなくなることにお気付きかもしれません。Attraction トランザクションをユーザーが直接呼び出し、観光名所を入力したり、値の一部を変更または削除したりすることはできなくなっています。これは、Attraction トランザクションは常にパラメーターを渡して呼び出すようになったためです。つまり、どのモードで開き、どの識別子の値に従ってインスタンス化するかを指定します。この点については、後ほどオブジェクト間の通信について学ぶところで詳しく説明します。

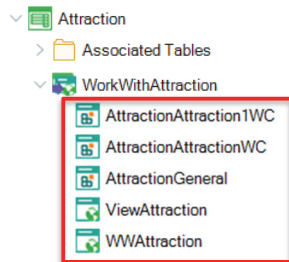
同様に…

```

1  Event Start
2  /* Generated by Work With Pattern [Start] - Do not change */
3  [web]
4  {
5  If not IsAuthorized(&PgmlName)
6  NotAuthorized(&PgmlName)
7  Endif
8
9  &TrnContext.FromXml(&WebSession.Get(!"TrnContext"))
10 &Insert_CountryId.SetEmpty()
11 &Insert_CategoryId.SetEmpty()
12 &Insert_CityId.SetEmpty()
13
14 If (&TrnContext.TransactionName = &PgmlName and &Mode = TrnMode.Insert)
15 For &TrnContextAtt in &TrnContext.Attributes
16 Do Case
17 // When inserting with instantiated CountryId
18 Case &TrnContextAtt.AttributeName = !"CountryId"
19 &Insert_CountryId.FromString(&TrnContextAtt.AttributeValue)
20 // When inserting with instantiated CategoryId
21 Case &TrnContextAtt.AttributeName = !"CategoryId"
22 &Insert_CategoryId.FromString(&TrnContextAtt.AttributeValue)
23 // When inserting with instantiated CityId
24 Case &TrnContextAtt.AttributeName = !"CityId"
25 &Insert_CityId.FromString(&TrnContextAtt.AttributeValue)
26 Endcase
27 Endfor
28 Endif
29 }
30 /* Generated by Work With Pattern [End] - Do not change */
31 EndEvent

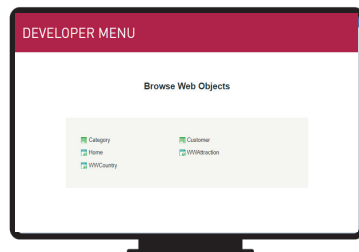
```

…これらのトランザクションではまだ触れていませんでしたが、[Events] エLEMENT を見ると、パターンによって大量のコードが追加されています。



Web パネル/Web コンポーネント

Work With エレメントの情報を整理して表示する画面に対応するオブジェクトは、Web パネルと呼ばれます (また、Web パネルの一部に特化して対応したものは Web コンポーネントと呼ばれます)。UI の重要な部分であるアプリケーションの画面を独自に開発する方法を知る必要があるため、このオブジェクトについては後ほど説明します。GeneXus によって実装されるものではありませんが、自分でも方法を学んでおく必要があります。



バックオフィス

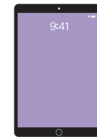
Web



スマートフォン



タブレット



まずトランザクション、そしてそれらのトランザクションに適用される Work With パターンによって作成される画面について見てきたことで、バックオフィスアプリケーションがどのようにできあがっていくかを確認しました。バックオフィスアプリケーションは、エンドユーザーにリリースする別のアプリのデータを処理する専用のアプリケーションです。この別のアプリは顧客向けアプリケーションと呼ばれることを思い出してください。

多くの場合、バックオフィスアプリケーションは単なる Web アプリケーションですが、そうでなくてもかまいません。スマートフォンやタブレットなどのデバイスで実行することもできます。別の Work With パターンを使って、同じような方法ですることができます。

Work With パターンの Attraction トランザクションへの適用

Attraction トランザクションの動作が変更される

新しいオブジェクト (Web パネル) が作成され、Work With の画面を実装する

開発者メニューが変更される

F5 キーを押す

データベースの再編成は不要

すべての新規オブジェクトおよび変更されたオブジェクトは、対応する言語でプログラミング、コンパイルされ、すべてがサーバーにアップロードされる

変更された開発者メニューを GeneXus が実行する

Attraction トランザクションに Work With パターンを適用したとき、何が起こったかを確認しましょう。

- Attraction トランザクションの動作に関連するいくつかのパーツが変更されました。
- Work With の画面を実装する Web パネルなど、新しいオブジェクトが複数作成されました。
- 開発者メニューが変更され、Attraction トランザクションを呼び出すのではなく、関連する Work With コンポーネントを呼び出すようになりました。そのためにいくつかのオブジェクトが変更されたり作成されたりしましたが、それらについてはここでは触れません。

次に、F5 キーを押したとき、何が起こったかを確認しましょう。

- GeneXus は、データに構造的な変化がなく、データベースの再編成を行う必要がないことを理解しました。
- また、新規のオブジェクトまたは変更されたオブジェクトそれぞれを対象の言語でプログラミングし、コンパイルしてから、すべてをサーバーにアップロードしました。
- 最後に、変更された開発者メニューを GeneXus が実行しました。

【補足】 ページソートと データベースソート

プログラムのフロントエンド部分 (ユーザーインターフェースを制御し、実行されたアクションに反応し、サーバーと通信するオブジェクトの一部) がクライアントのブラウザにロードされます。プログラムのこの部分は、グリッド (すべての観光名所) に表示する情報をサーバーに要求します。プログラムのバックエンド部分は、データベースに情報を要求し、それをブラウザに返します。ブラウザはロードしたページを表示します。

グリッドのタイトルを操作したときのソートは、クライアントで処理するページソートで
インスタンスにより生成したソートは、プログラムのバックエンド部分に情報を要求し、それをブラウザーに返します。



動画

<https://www.genexus.com/community-and-support-jp/training?ja>

ドキュメント

<http://wiki.genexus.jp/>

認定資格

training.genexus.com/certifications