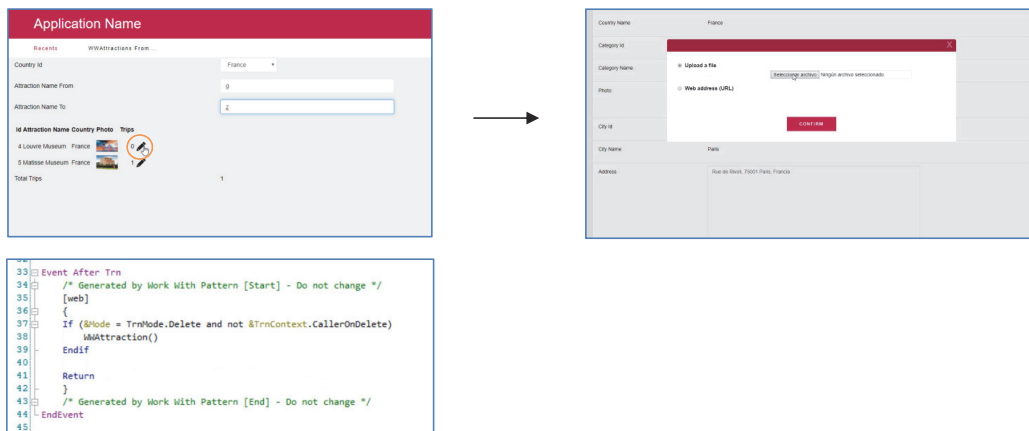


インタラクティブ画面

コンテキスト情報の保存方法

GeneXus[™]

Web パネル WWAttractionFromScratch



この例では、あるオブジェクトから別のオブジェクトを呼び出し、また元のオブジェクトに戻る場合に、データが失われないようにメモリーに保持する方法を学習します。

どのような状態か説明するために、前の章と同じ例を使用します。

概要を説明します。

画面の Web パネルでフィルタに値を入力すると、グリッドが再表示され、入力した値に基づいて対応するデータのみが表示されます。

いずれかの明細行で更新アクションを選択すると、Attraction トランザクションが Update モードになります。ここで、選択した観光名所に対応する値を編集できます。

たとえば、ルーブル美術館を選択して編集するとします。

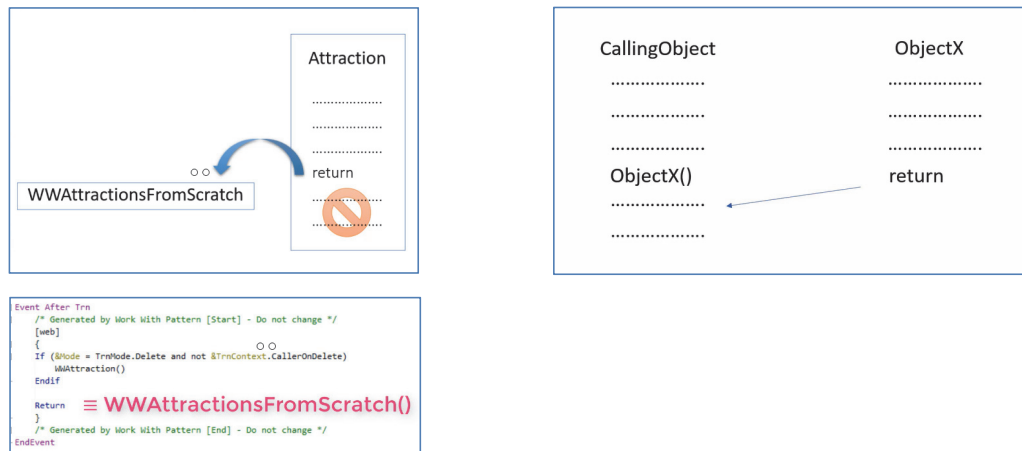
AttractionPhoto 項目属性に対応する画像を変更し、[実行] を選択します。

このアクションによって Web パネルに戻ります。

これは、Attraction トランザクションに適用されている Work With パターンが、呼び出し元オブジェクトに戻るために Return コマンドを自動的に追加したためです。Attraction トランザクションの [Events] エLEMENTを見ると、After Trn イベント内にこれがプログラミングされていることが分かります。

After Trn イベントは、トランザクションで 1 つのサイクルが完了したときに、コミット操作の直後にトリガーされます。つまり、ヘッダーとそれに対応する明細行が記録された後です。

Web パネル WWAttractionFromScratch



Return コマンドの機能は、このオブジェクトの現在の実行を終了し、呼び出し元オブジェクトに戻ることです。

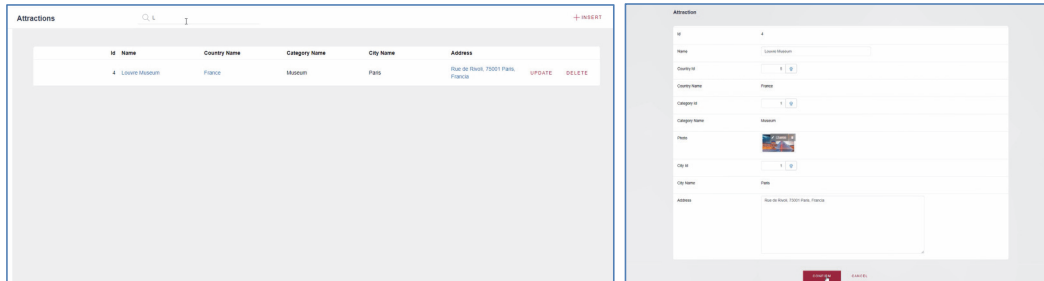
この例では、WWAttractionsFromScratch Web パネルと Attraction トランザクション (つまり、呼び出し元オブジェクトと呼び出し先オブジェクト) の両方が、グラフィカルインターフェースを持つオブジェクトです。そのため、この場合の Return コマンドは、Web パネルを初めて呼び出すのと同じになります。

どちらかのオブジェクトが、プロシージャータのようにグラフィカルユーザーインターフェースがない場合は異なります。呼び出されたオブジェクトの Return コマンドにより、呼び出しの直後の行に戻ります。

この場合は、Web パネルのロードと関連付けられたイベントが実行されます。まず Start イベント、次に Refresh イベント、その後に Load イベントが実行されます。Load イベントは、[Conditions] プロパティで指定された条件を満たす、グリッド内のレコードの数だけ実行されます。変数が空の場合は条件が適用されないため、実行時にすべての観光名所が再ロードされます。

なぜでしょうか。それについては後で説明します。

Work With Attraction パターン



まず、Attraction トランザクションの [Patterns] エlementから自動的に生成された Web パネルの動作を見てみましょう。

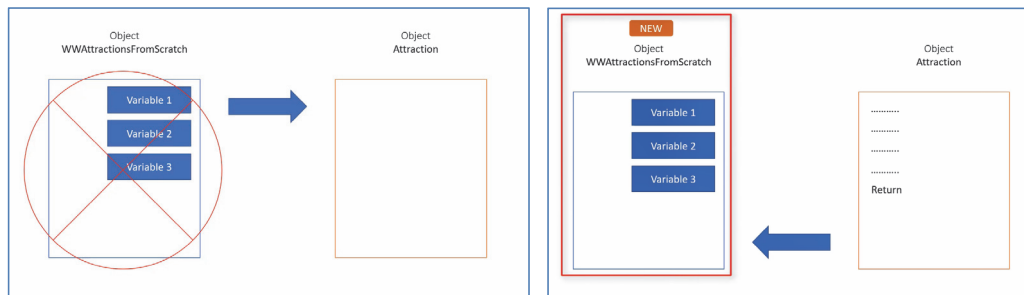
「L」と入力して観光名所を名前でフィルタリングすると、この文字で始まる観光名所がすべて表示されます。この場合、登録されている観光名所で該当するのはルーブル美術館だけです。

次に、この観光名所を更新するアクションを選択します。

これにより、Attraction トランザクションに移動し、選択した観光名所が表示されて編集できるようになります。これは、Web パネルを手動で実装した場合とまったく同じです。

写真を変更して更新処理を確定すると、呼び出し元の Web パネルに戻ります。フィルタが維持されていることに注目してください。これが、実装する必要がある機能です。

Work With Attraction パターン



次に、Web パネルでソリューションがプログラミングされ、このパターンによる処理に関する説明が表示されます。

各オブジェクトで指定されている変数は、オブジェクトがアクティブである間、それらの内部でのみ使用できることに留意してください。

たとえばこの場合は、Web パネルから Attraction トランザクションを呼び出すと WWAttractionFromScratch オブジェクトが破棄され、変数を受け取ります。そのため、保存されていた値はすべて失われます。一方、Attraction オブジェクトのステータスはアクティブになります。

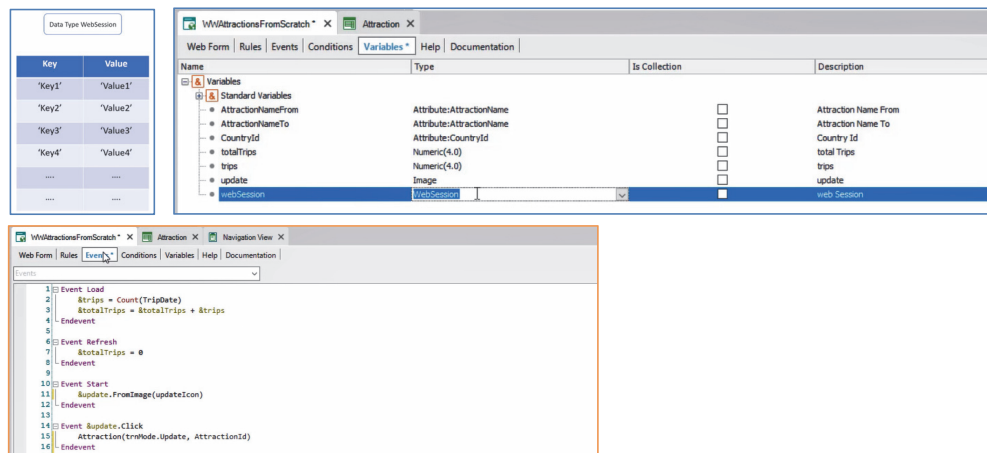
Return コマンドでトランザクションから Web パネルに戻る際には、後者のオブジェクトとその変数が再作成されます。

フィルタの値が表示されなくなるのはこのためで、実際には**新しいオブジェクト**になります。また、フィルタとして使用された変数が再作成され、Attraction オブジェクトを呼び出す前のものは破棄されます。

これが、フィルタに入力した値がレコードの更新後に維持されないのはなぜかという、先ほどの疑問の答えです。

各フィルタの情報をグローバル変数に格納して、1 つのオブジェクトから別のオブジェクトに移動する際に失われないようにする必要があります。この場合は、Web パネルからトランザクションに移動するときと、トランザクションから Web パネルに戻るときです。

WebSession データタイプ



変数の値をメモリに保持するにはどうすればよいでしょうか。GeneXus では、Web Session タイプの変数を使用して、この機能をプログラミングできます。

これらの変数を使用することで、グローバル変数のグループを処理し、それにデータを格納し、セッションがアクティブな間、任意のオブジェクトからアクセスできます。

それが、この例の目的です。

これらの変数の主なメリットは、key-value タイプのデータのセットを格納できることです。Web Session 変数を作成して、それぞれ一意のキーを持つすべてのフィルタを保存できます。

つまり、CountryId フィルタ、AttractionNameFrom フィルタ、および AttractionNameTo フィルタのそれぞれに、一意のキーと値があります。これら 3 つのフィルタは、パネルの実行間で保持する必要があります。

これにより、フィルタに入力した情報を一時的に保存し、後で取得することができるようになります。このプログラミング方法を見てみましょう。

まず、変数を作成し、WebSession という名前をつけます。

GeneXus は、この名前から推論して、WebSession データタイプを割り当てます。ここではそれが適切な処理になります。自動的に割り当てられたデータタイプが適切ではなかった場合は、ユーザーが変更できます。

次に、この変数がいつ目的の値を維持するかを検討します。

現在スケジュールされているメインイベントを思い出してください。

- Start イベントは、ページを最初にロードするときに 1 回だけ実行されます。
- Refresh イベントは、Start イベントの後にトリガーされるほか、グリッド条件内のフィルタが変更されるたびに、またはブラウザからページが再表示されるたびにトリガーされます。
- Load イベントは、Refresh イベントの後で、データをグリッドにロードするたびに実行されます。

このイベントには関連付けられたベーステーブルがあるため、N 回実行されます。これら 3 つのうち、データを保存するのが最適なのはどれでしょうか。これらのうちで最も便利なのは Refresh イベント内です。このイベントは、フィルタの値を変更するたびに必ずトリガーされます。そのときにセッション変数に保存され、トランザクションから戻るときに取得することができます。ほかに、Update 変数の Click イベントもあります。このイベントは、Update アクションをクリックするとすぐにトリガーされます。このイベントでも、フィルタの値を保存できると便利です。これは Attraction トランザクションを呼び出すイベントであり、既に見たように、そのときに呼び出し元オブジェクトが破棄され、その変数を取得するからです。実行前にフィルタの変数の値を保存する必要があるのはそのためです。

Refresh イベントを使用する場合は、データを変更するたびに、フィルタの値を保存します。グリッド条件内にあるため、変更を行うたびにこのイベントがトリガーされます。ただし、現在の実行ではフィルタされた観光名所を更新する必要がない場合もあります。その場合、変数が削除されないのにこれらのフィルタをセッションに保存する必要はありません。しかし、Update アクションに関連するイベントを使用する場合は、必要に応じて、これらの値を **1 回だけ**、オブジェクトとその変数が破棄される直前に保存します。これは、変数の Click イベントで実行します。

WebSession データタイプ

```

Event Start
  &update.FromImage(updateIcon)
  &CountryId = &webSession.Get('CountryId').ToNumeric()
  &AttractionNameFrom = &webSession.Get('AttractionNameFrom')
  &AttractionNameTo = &webSession.Get('AttractionNameTo')
Endevent

```

```

Event &update.Click
  &webSession.Set('CountryId', &CountryId.ToString())
  &webSession.Set('AttractionNameFrom', &AttractionNameFrom)
  &webSession.Set('AttractionNameTo', &AttractionNameTo)
  Attraction(trnMode.Update, AttractionId)
Endevent

```

webSession 変数を入力し、Set メソッドを適用して値をそれに格納します。最初のパラメーターとしてキーを入力する必要があります。これは Character タイプの値である必要があるため、引用符で囲みます。

この場合は CountryId です。この一意のキーは、保存した値を回復するために後で使用します。

次に、メモリーに格納するデータを割り当てます。この場合は、CountryId 変数の値を保存する必要があります。これは、画面上の 1 つ目のフィルタの値です。

この値も Character タイプである必要があります。CountryId 変数は Numeric タイプであるため、Character タイプに変換する必要があります。これを行うには、この変数に ToString メソッドを適用します。

AttractionNameFrom フィルタと AttractionNameTo フィルタにも同じ処理を実行します。

次に、このアプリケーションを再度実行すると、値がフィルタに入力されます。いずれかのフィルタの値を変更するたびに Refresh イベントがトリガーされ、その後で、グリッドにレコードをロードするたびに Load イベントがトリガーされます。

1 つの観光名所の Update アクションを選択します。

このときに、&Update.Click イベントがトリガーされます。これは、フィルタの値を &webSession 変数に保存するためにプログラミングされたものです。これは、Attraction トランザクションを後で呼び出すために行われます。前述のように、Web Panel オブジェクトとその変数が破棄されるときです。

この観光名所のデータを変更し、変更処理を確定した後、Return コマンドが適用されます。この場合は Web Panel オブジェクトを初めて呼び出すのと同じ処理であるため、グリッドに値をロードするたびに、Start イベント、Refresh イベント、および Load イベントが再度実行されます。

これで、webSession 変数に保存した値を取得できるようになるはずです。

これらの値を取得するのに最適なタイミングはいつでしょうか。

それぞれのイベントについて考えてください。

Start イベントでしょうか。

Refresh イベントでしょうか。

Load イベントでしょうか。

&Update.Click イベントでしょうか。

この場合に最適なのは Start イベントです。説明したように、このイベントは、Web パネルが最初にロードされるときに 1 回だけ実行されます。

トランザクションから戻るときは、Web パネルを最初に呼び出すときと同じ処理が行われるため、これらの値を取得するのに最適なタイミングです。

キーと値を webSession 変数に割り当てるには、Set メソッドを使用します。次は、格納されている値を取得する方法です。

これを行うには Get メソッドを使用します。その際には、取得する値のキーを指定する必要があります。

次に、フィルタとして使用する各変数に Get メソッドを適用し、パラメーターを使用して、それぞれに対応するキーを送ります。

この処理を、まず CountryId 変数に対して実行します。

CountryId 変数を入力し、webSession 変数の Get メソッドの値を割り当て、パラメーターとしてキー (CountryId) を渡します。

前に説明したことを思い出してください。Web Session 変数は Character タイプのデータのみを格納します。CountryId 変数は Numeric タイプであるため、この変数に Character タイプである WebSession 変数の値を割り当てるには、Numeric タイプに変換する必要があります。この変換を行うには、ToNumeric() メソッドを使用します。

次に、AttractionNameFrom 変数と AttractionNameTo 変数にも同じ処理を実行します。

アプリケーションを再度実行し、動作をテストします。

中国のすべての観光名所で A から M のものをフィルタリングします。

この場合、1 つの観光名所だけが表示されます。

その観光名所を更新するアクションを選択します。このとき、トランザクションを呼び出す前のフィルタの変数のデータがメモリーに保存されることを思い出してください。

写真を変更し、処理を確定します。

説明したように、Web パネルに戻る際に最初に実行されるのは Start イベントです。このイベントで、セッション変数に保存した情報の取得をプログラミングし、値をフィルタ変数に割り当てます。

フィルタに入力した値は、想定どおりに維持されます。

また、Start イベントの後に、Refresh イベントがトリガーされます。これにより、フィルタに基づいてグリッドのロードがトリガーされます。変更後の写真でグリッドが再度フィルタされるのはこのためです。

この動作がプログラミングされた状態で、ユーザーがこの Web パネルを初めてブラウザで開いたらどうなるでしょうか。その場合、何も保存されていないため、何も取得されません。

ブラウザでアクティブなセッションを閉じ、GeneXus からアプリケーションを起動して、サイクル全体を再度実行します。

作成した Web パネルを開きます。

このセッションは Web パネルを最初に実行するセッションであるため、まず Start イベントがトリガーされます。

セッション変数は入力されたキーを参照して、取得する情報があるかどうかを確認します。&webSession 変数にはまだ値が格納されていないため、取得する情報はありません。ここでは、フィルタに使用されている変数は空のままです。これらの変数が空の場合はどのフィルタも適用しないように、グリッド条件で指定されているため、すべての観光名所が表示されます。

たとえば、フランスという国名でフィルタすると、グリッドにフィルタが適用され、フランス国内の観光名所が表示されます。

[Attraction Name From] フィルタに F と入力すると、Refresh イベントが再度トリガーされ、その後に Load イベントがトリガーされます。最後に [Attraction Name To] フィルタに O と入力します。

マチス美術館に対する更新アクションを選択します。

ここで &Update.Click イベントがトリガーされます。このイベントでは、Attraction トランザクションの呼び出し前に、フィルタ変数のデータが &webSession セッション変数に保存されるようにプログラミングされています。そのために、セッション変数の Set メソッドが使用され、パラメーターに保存するキーと値が渡されます。この場合は、メモリーに 3 つの値を保存します。

&CountryId 変数、&AttractionNameFrom 変数、および &AttractionNameTo 変数です。

この時点で Attraction オブジェクトが呼び出され、アクティブになります。また、Web Panel オブジェクトが、その値とともに破棄されます。

写真など、観光名所のデータの一部を変更し、変更処理を確定します。

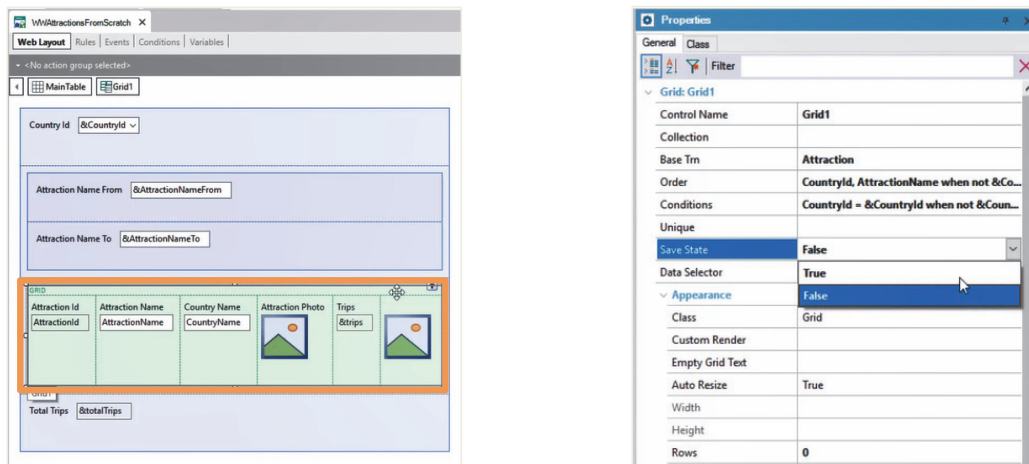
これにより、呼び出し元のオブジェクトである Web パネルに戻ります。これは、前に説明したように、Attraction トランザクションに適用された Work With パターンによって、After Trn イベント内に Return コマンドが追加されたためです。この時点で、グリッドにロードするレコードごとに、Start、Refresh、Load の各

イベントが再度実行されます。これは、この場合の Return コマンドは、Web パネルを初めて呼び出す場合と同じ処理になるためです。

セッション変数に保存された値は、Start イベントでロードされます。これは Get メソッドを使用して実行され、Set メソッドで &Update.Click イベントに値を保存するのに使用するキーが、パラメーターで渡されます。
このデータは、対応する各変数、つまり、この場合はフィルタ用に使用される 3 つの変数に割り当てられます: &CountryId、&AttractionNameFrom、および &AttractionNameTo です。

次に Refresh イベントが実行されてから、グリッドにロードするレコードごとに Load イベントが実行されます。
この場合、条件を満たすのは 2 つの観光名所で、それらがグリッドにロードされます。そのため、Load イベントは 2 回実行されます。
画面を見ると、フィルタに入力された値がレコードの更新後も維持されていることが分かります。

[Save State] プロパティ



ここまで、GeneXus で提供されるセッション変数を使用し、情報をそれに保存してから、必要なときにその情報を取得できることを学習しました。GeneXus ユーザーは、グリッドの状態を保存し、後でその状態を回復する必要があることがあります。先ほど説明したように、ほかのターゲットを呼び出して、グリッドが設定された Web パネルに戻ったときに、入力したフィルタが維持されている必要がある場合などです。そのために、GeneXus の最新バージョンでは、グリッドコントロールに [Save State] プロパティが追加されています。このプロパティは、基本的に、メモリー内 (Web セッション変数内) にグリッドの状態 (表示されているグリッドページ、適用されているフィルタなど) に関するすべての情報を保存するために使用します。このページを再ロードすると、最新の構成が自動的に復元されます。これが、この章でセッション変数を使用して手動で行ったことです。ただし、プロパティを設定し、自動的かつ完全に透過的な方法で実行され、生成されたオブジェクトには追加されたコードは表示されません。実行環境で確認してみましょう。まず、Web パネルのイベントセクションの内容を削除し、この章の最初の状態にして、どこにも保存されないようにします。Web Panel オブジェクトからグリッドを選択すると、新しいプロパティが表示されます。既定の値 False では非表示になりませんが、値を True に変更して再度実行します。これまで見てきた通り、フィルタに値を入力し、観光名所の 1 つの情報を選択し、変更します。1 つのオブジェクトから別のオブジェクトが呼び出され、呼び出されたオブジェクトでアクションを確定すると、呼び出し元のオブジェクトに戻ります。変更を加えて処理を確定し、Web パネルに戻ると、フィルタが維持されているのが分かります。Attraction トランザクションを呼び出す前の状態になっています。このグリッドにページングがあった場合でも、レコードを更新する前の位置が維持されます。ここでは詳しくは説明しませんが、コンテキスト情報を保存して取得する方法は、ほかに 2 つあります。1 つは、グリッドコントロールのメソッドを使用する方法です。SaveSessionState を使用すると、グリッドの状態を保存できます。また、LoadSessionState で、前に保存したグリッドの状態を取得できます。これらのメソッドを使用すると、今説明した [Save State] プロパティを使用するのと同じことになります。

もう 1 つは ClientStorage 外部オブジェクトを使用する方法です。
このオブジェクトは、コンテキスト情報をモバイルデバイス向けアプリケーションに格納するための API です。この用途と操作は、WebSession 変数の場合と似ています。

この方法では、キーと値のペアを使用して、情報をローカルに、つまり使用しているモバイルデバイスに格納できます。また、接続されていない状態でも、後でその情報にアクセスできます。

これらの実装の詳細については、GeneXus の Wiki を参照してください。