日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2021年1月7日木曜日

Oracle APEXアプリケーションのデプロイメントに使用できる設定の紹介

幸運なことに作成したOracle APEXのアプリケーションを使う人が増え、色々機能を追加していくと、次第にページが増えていきます。ある程度ページが増えてくると、アプリケーションを更新するのが辛くなります。

基本的にOracle APEXアプリケーションのデプロイメントの単位はアプリケーションなので、それに大量のページが含まれていると、その内の１ページだけの変更であっても、エスクポートは全ページで、インポートも全ページの置き換えになります。また、変更が１ページであっても、それに伴って共有コンポーネントも変更しているかもしれませんし、改変したアプリケーションと現行のアプリケーションとの差分を、稼働しているアプリケーションに適用する、というのも難しい作業です。置き換える単位がアプリケーションであれば、テストの単位もアプリケーションとなるのが通常でしょう。

本記事では、上記のような難しさを軽減するために、比較的容易に採用できる方法について紹介します。項目は以下になります。

1. 例題アプリケーションの作成
2. アプリケーションの分割
3. セッションの共有
4. ホーム・ページ、ログイン・ページの共有
5. ナビゲーションの更新
6. 共有コンポーネントのサブスクリプション
7. 別名の付け替えと可用性の設定
8. その他の利用可能な機能

最初に例題となるアプリケーションを作成し、そのアプリケーションに対して、ここでリストした作業を実施していきます。

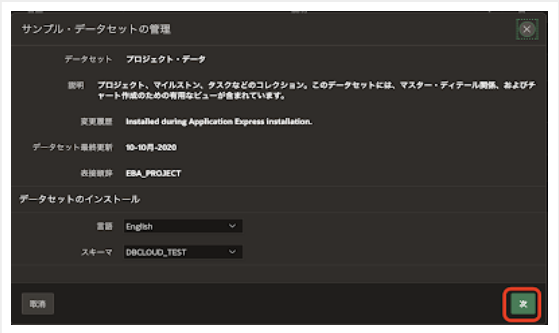
例題アプリケーションの作成

SQLワークショップのサンプル・データセットより、プロジェクト・データをインストールし、それを元にアプリケーションを作成します。

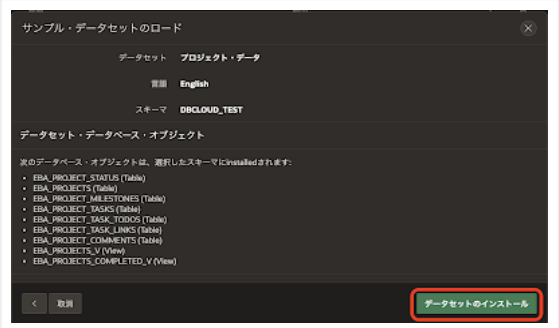
SQLワークショップの**ユーティリティ**に含まれる**サンプル・データセット**を開きます。

プロジェクト・データの**インストール**をクリックします。

データセットの説明が表示されます。**次**へ進みます。

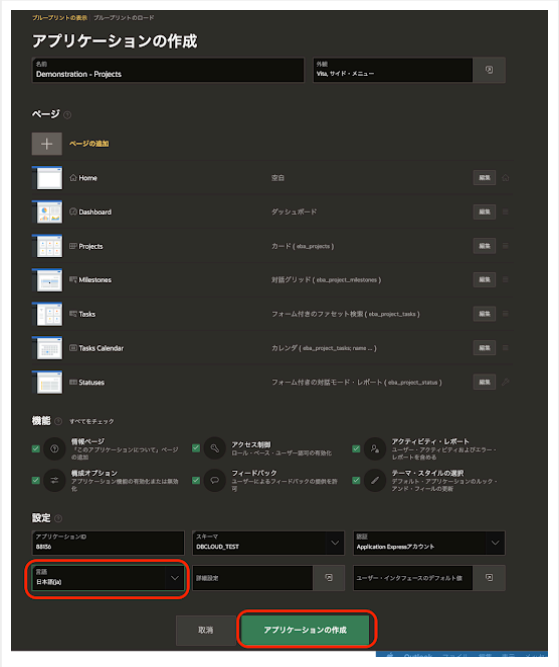


インストールされるデータセットがリストされます。確認して**データセットのインストール**を実行します。



ロード結果が表示されます。そこから**アプリケーションの作成**を起動します。

言語を**日本語(ja)**に変更し、それ以外はそのままにして**アプリケーションの作成**を行います。



以上で例題にするアプリケーションが完成しました。

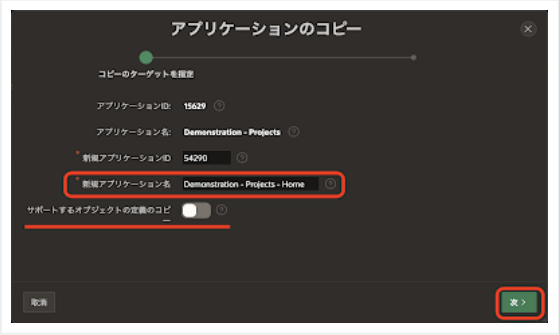
アプリケーションの分割

作成されたアプリケーションを以下に示すように、アプリケーションの別名**dm-projects-001**と別名**dm-projects-002**の、２つのアプリケーションに分割します。

Home, Dashboard, Statuses, Project Statusはdm-projects-001、Projects, Milestones, Tasks Project Task, Task Calendarはdm-projects-002、それ以外のページは双方のアプリケーションに残します。

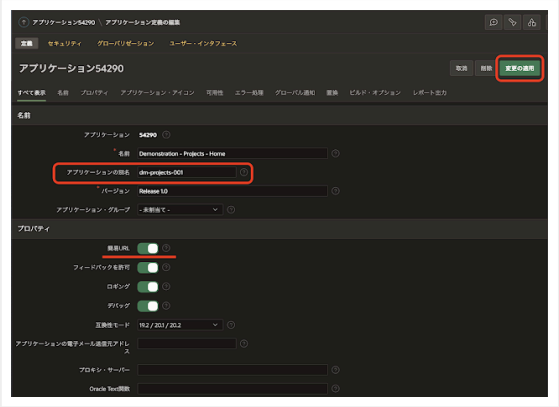
作成されているアプリケーションをコピーします。**タスク**から**このアプリケーションのコピー**を実行します。

新規アプリケーション名はDemonstration - Projects - Homeとします。**サポートするオブジェクトの定義のコピー**は不要なので、**OFF**です。**次**へ進みます。

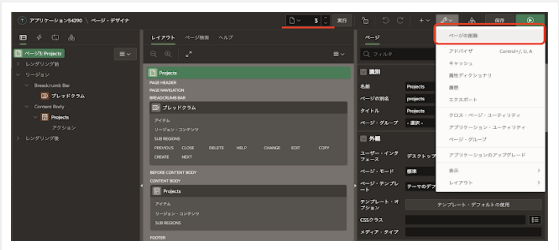


確認画面から**アプリケーションのコピー**を実行します。

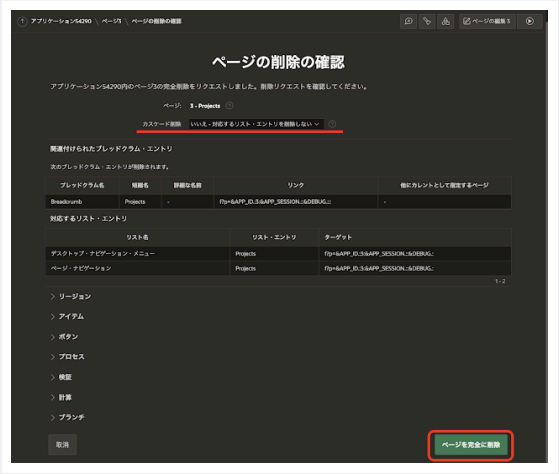
これで同一のアプリケーションが、ワークスペースに２つ存在することになります。最初にアプリケーション**Demonstration - Projects - Home**をアプリケーション・ビルダーで開き、**別名**を**dm-projects-001**と設定します。これからの説明は、簡易URLはONとOFFのどちらでも有効ですが、**簡易URL**がサポートされているバージョンであれば**ON**にするべきでしょう。設定ののち、**変更の適用**をクリックします。



このアプリケーションから、**ページ3, 4, 5, 6, 7を削除**します。ページ・デザイナにて、ひとつひとつページを開いて、削除していきます。



カスケード削除の確認を要求された場合は、**いいえ - 対応するリスト・エントリを削除しない**を選びます。そして**ページを完全に削除**を実行します。



アプリケーションdm-projects-001よりページを削除したら、次にアプリケーションDemonstration - Projectsを開き、**別名**を**dm-projects-002**として設定し、こちらからは**ページ1, 2, 8, 9を削除**します。

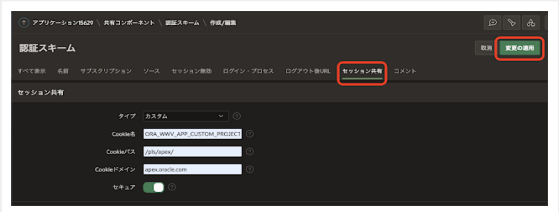
以上でアプリケーションの分割は完了です。次から、分割したアプリケーションを一体で動作するように設定を行なっていきます。

セッションの共有

それぞれのアプリケーションをアプリケーション・ビルダーで開き、**共有コンポーネント**の**認証スキーム**を開きます。

カレントの認証スキームを開き、**セッション共有**の設定を変更します。**Cookie名**は**任意の値**(ここでは**ORA_WWV_APP_CUSTOM_PROJECTS**としています)ですが、それ以外はOracle APEXをホストしているサーバーの環境によって決まる値ですので、ご自身の環境に合った値を設定してください。

- タイプ: **カスタム**
- Cookie名: **ORA_WWV_APP_CUSTOM_PROJECTS**
- Cookieパス: **/pls/apex/**
- Cookieドメイン: **apex.oracle.com**
- セキュア: **ON**



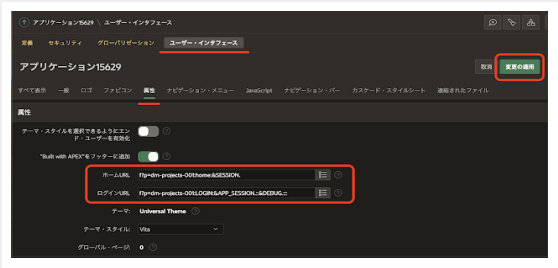
分割したアプリケーションの両方に同じ設定を行うことで、クッキーに設定されるセッション識別子をアプリケーションで共有します。

ホーム・ページ、ログイン・ページの共有

今回はDemonstration - Projects - Home(別名dm-projects-001)のアプリケーションにホーム・ページを残しているため、別名dm-projects-002のアプリケーションは、ホーム・ページとログイン・ページとしてdm-projects-001のものを使うようにします。

dm-projects-002のアプリケーションのアプリケーション定義の**ユーザー・インターフェース**を開き、**属性**の**ホームURL**と**ログインURL**を変更します。

- ホームURL: f?p=**dm-projects-001**:**home**:&SESSION.
- ログインURL: f?p=**dm-projects-001**:LOGIN:&APP_SESSION.::&DEBUG.:::



&APP_ID.置換文字列は自分自身のアプリケーションIDを意味するので、これを分割したアプリケーションである**dm-projects-001**へ置き換えます。また、ホームURLのページが**1**として数値で設定されていたので、別名の**home**に置き換えています。アプリケーションIDやページIDとして指定する部分には、できるだけ数値は使わない方がアプリケーションのメンテナンスが容易になります。

こちらについては、アプリケーションdm-projects-002だけの変更になります。アプリケーションに認証が必要なアクセスがあった場合、および、ホームページに移動する場合は、アプリケーションdm-projects-001へ遷移するようになります。

ナビゲーションの更新

ページ1, 2はアプリケーションdm-projects-001、ページ3, 4, 5, 6, 7はアプリケーションdm-projects-002をアクセスするよう、ナビゲーション・メニューを更新します。

共有コンポーネントの**リスト**を開きます。この中のデクストップ・ナビゲーション・メニューとページ・ナビゲーションを変更します。

他にも変更すべきリストがあるのですが、今回の例題の確認には影響がないため、そのままにします。

デスクトップ・ナビゲーション・メニューを開いて、**グリッド編集**をクリックします。

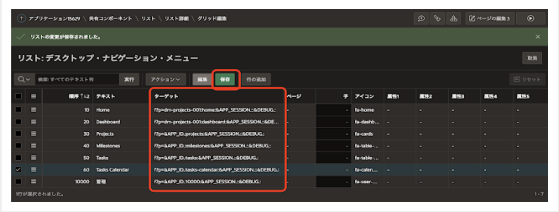
分割されたアプリケーションを指すエントリのターゲットに、アプリケーションの別名を指定します。

アプリケーション**dm-projects-001**では、以下のターゲットを変更します。HomeとDashboardはそのままでも大丈夫なのですが、ページ番号をページの別名に置き換えています。

- Home: f?p=&APP_ID.:home:&APP_SESSION.::&DEBUG.:
- Dashboard: f?p=&APP_ID.:dashboard:&APP_SESSION.::&DEBUG.:
- Projects: f?p=dm-projects-002:projects:&APP_SESSION.::&DEBUG.:
- Milestones: f?p=dm-projects-002:milestones:&APP_SESSION.::&DEBUG.:
- Tasks: f?p=dm-projects-002:tasks:&APP_SESSION.::&DEBUG.:
- Tasks Calendar: f?p=dm-projects-002:tasks-calendar:&APP_SESSION.::&DEBUG.:

アプリケーション**dm-projects-002**では、以下のターゲットを変更します。

- Home: f?p=dm-projects-001:home:&APP_SESSION.::&DEBUG.:
- Dashboard: f?p=dm-projects-001:dashboard:&APP_SESSION.::&DEBUG.:
- Projects: f?p=&APP_ID.:projects:&APP_SESSION.::&DEBUG.:
- Milestones: f?p=&APP_ID.:milestones:&APP_SESSION.::&DEBUG.:
- Tasks: f?p=&APP_ID.:tasks:&APP_SESSION.::&DEBUG.:
- Tasks Calendar: f?p=&APP_ID.:tasks-calendar:&APP_SESSION.::&DEBUG.:



ページ・ナビゲーションも同様に変更します。ページ・ナビゲーションはホーム・ページでのみ使用されるので、dm-projects-001だけ変更します。

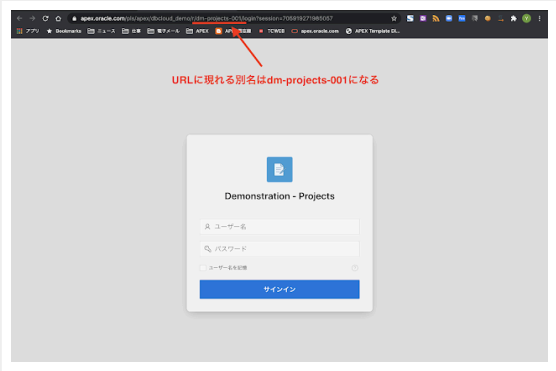
以下のターゲットを変更します。

- Dashboard: f?p=&APP_ID.:dashboard:&APP_SESSION.::&DEBUG.:
- Projects: f?p=dm-projects-002:projects:&APP_SESSION.::&DEBUG.:
- Milestones: f?p=dm-projects-002:milestones:&APP_SESSION.::&DEBUG.:
- Tasks: f?p=dm-projects-002:tasks:&APP_SESSION.::&DEBUG.:
- Tasks Calendar: f?p=dm-projects-002:tasks-calendar:&APP_SESSION.::&DEBUG.:

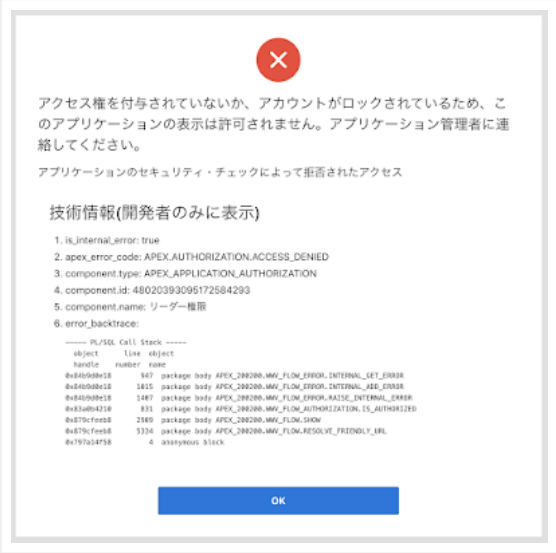
アプリケーションにたいして簡易URLを設定していても、ナビゲーションに指定するURLについてはf?p=形式のURLを使用できます。簡易URLの形式で指定するにはワークスペース名を含める必要があるなど、ナビゲーションの宛先URLとしては使いにくいのでf?p=形式で指定できるのは助かります。

ページの遷移にまつわる設定は以上で完了です。アプリケーションを実行して動作を確認してみます。

どちらのアプリケーションを実行しても、ログイン・ページはアプリケーションdm-projects-001に含まれるページが開きます。

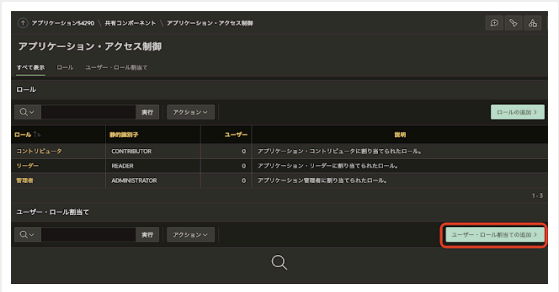


ログインすると以下のエラーが発生しました。



コピーとして作成したアプリケーション(今回の手順ではdm-projects-001がコピー)の、**共有コンポーネント**の**アプリケーション・アクセス制御**を開きます。

ユーザー・ロール割当ての追加を実行します。アプリケーション作成時に追加したアクセス制御の機能が有効な範囲はアプリケーション単体で、コピーしたアプリケーションやエクスポート/インポートしたアプリケーションにユーザー・ロールの割当てはコピーされません。

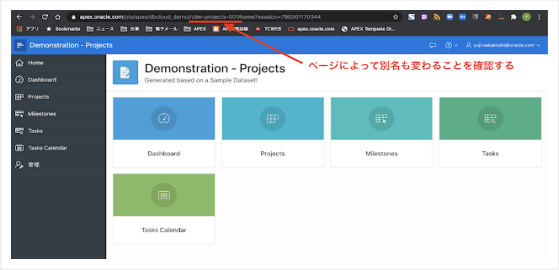


今現在ログインしている**ユーザー名を指定**し、**アプリケーション・ロール**として**管理者**と**コントリビュータ**にチェックを入れ、**割当ての作成**を行います。

アプリケーション・ロールがユーザーに割当たったことを確認します。

アプリケーション・ロールを有効にするためには再度ログインする必要があります。エラーが表示されているページのURLより**session=**を取り除いてアクセスする、ブラウザを一旦閉じるなどして、セッションを初期化してログインし直します。

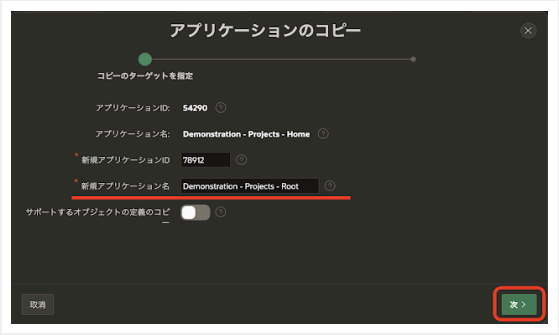
サイド・メニューやホーム・ページのナビゲーションをクリックして、アプリケーション自体も遷移すること、アクセスするアプリケーションが変わっても再認証が求められないことを確認します。



共有コンポーネントのサブスクリプション

共有コンポーネントはサブスクリプションの設定によって、異なるアプリケーションに作成されている共有コンポーネントの定義を流用することができます。

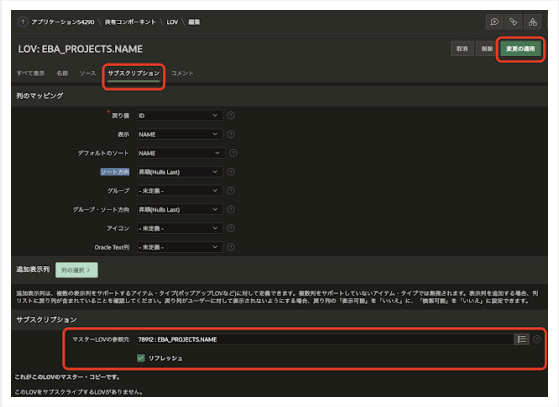
サブスクリプションの元となる共有コンポーネントを保持するためのアプリケーションを**名前Demonstration - Project - Root、別名dm-projects-root**として作成します。dm-projects-001をコピーします。アプリケーションのコピーの手順、別名の設定については、すでに説明済みですので、そちらの手順に沿って行います。



他のアプリケーションからサブスクライブされている共有コンポーネントを保持しているアプリケーションは、入れ替えや削除を行うと影響が大きいので、エンドユーザーに提供する画面などを含まず、別のアプリケーションとすることを推奨します。今回は例題なので、作成したコピーからのページ削除の実施は省きます。

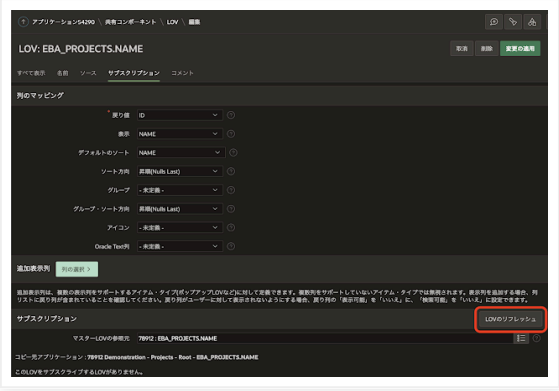
続いて、共有コンポーネントのLOVにサブスクリプションの設定を行なってみます。アプリケーション**dm-projects-001**をアプリケーション・ビルダーで開き、**共有コンポーネント**の**LOV**を開き、**EBA_PROJECTS.NAME**を開きます。

サブスクリプションとして、**マスターLOVの参照元**を設定します。**リフレッシュ**にチェックが入っていると、**変更を適用**したときにマスターLOVの設定で上書きされます。



サブスクリプションの設定がされていると、リストに**サブスクライブ元**となる**アプリケーションID**が表示されます。

再度、EBA_PROJECTS.NAMEを開きます。



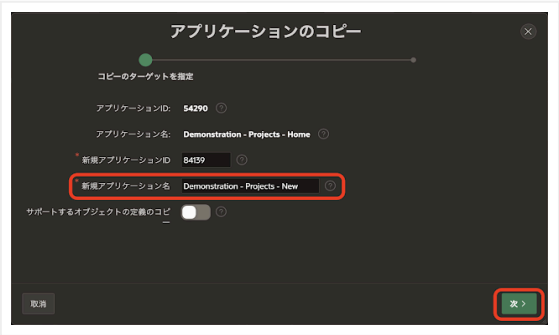
マスターLOVおよび、それをサブスクライブしているLOVの双方で、それぞれLOVの設定を変更することが可能です。**LOVのリフレッシュ**を実行するまでは、マスターLOVの設定で上書きされることはありません。

サブスクリプションをサポートしている共有コンポーネントでは、親となるアプリケーションにコンポーネントを作成し、そのアプリケーションからコンポーネントをサブスクライブするように設定することで、設定作業の冗長化を避けることができます。

別名の付け替えと可用性の設定

小さな単位に分割したアプリケーションのコピーを作成して、アプリケーションの改良といった作業を進めることもできます。

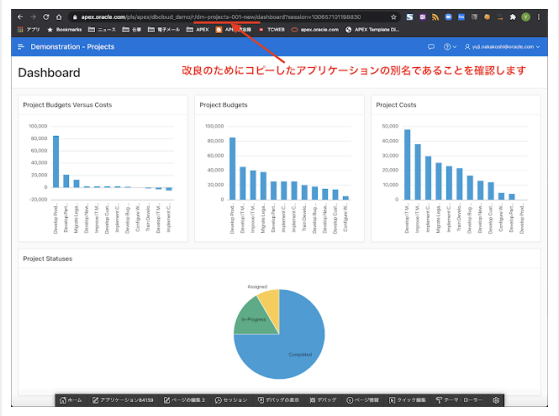
例えばアプリケーションdm-projects-001のコピーを作成し、Dashboardページを改良することができます。新規アプリケーション名は**Demonstration - Projects - New**とします。



作成したアプリケーションのコピーは、**別名**として**dm-projects-001-new**を設定します。

コピーしたアプリケーションを実行します。アプリケーションを実行してサイド・メニューやナビゲーションを辿ると別のアプリケーションへ遷移することがあるので注意が必要です。

以下のようにレイアウトを変えてみました。(Dashboardの改良自体が本題ではないので、どのように変更してもかまいません)。



アプリケーションの別名を入れ替えることにより、利用中のアプリケーションと更新したアプリケーションを交換することができます。

アプリケーションの定義を開き、別名dm-projects-001をdm-projects-001-oldに変更し、その後、dm-projects-001-newをdm-projects-001へ変更します。

先に、現在利用中のアプリケーションの別名を変更します。

続いて、機能を変更したアプリケーションの別名を変更します。

以上でアプリケーションの入れ替えが完了です。入れ替えたアプリケーションに問題があれば、別名の割り当てを元に戻せば、以前の状態に戻ります。(同時にバージョンやアプリケーション・グループなども更新すると、対応としてはより良くなります)。

このままの状態であれば、URLを意図的に指定することで、以前のアプリケーションと新しいアプリケーションの両方にアクセスできます。古いアプリケーションのアプリケーション定義を開き、**可用性**を定義することで、古いアプリケーションへのアクセスを禁止し、新しいアプリケーションの利用を強制することができます。

可用性の**ステータス**として**使用不可(URLにリダイレクト)**を選択し、**使用できないアプリケーションに対するメッセージ**として、**リダイレクト先となるURL**を指定します。

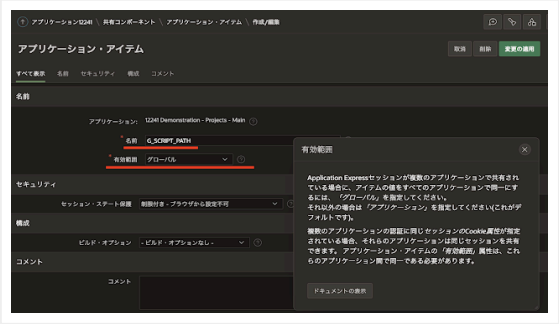
URLは簡易URLが使用される場合は簡易URLで、そうでない場合はf?p=の形式で指定します。リダイレクト先となるアプリケーションは別ワークスペースや別インスタンスのアプリケーションでも指定可能ですし、単なるHTTPのリダイレクト先ですので、Oracle APEXのアプリケーションでなくても指定できます。

ブックマークされたURLがすでに簡易URLの場合は、アプリケーションの指定は名称によるので、別名の入れ替えだけでブックマークはそのまま有効です。そのため、可用性の設定は必ずしも必要ではありません。簡易URLが導入される前のf?p=形式で、アプリケーションIDとして数値が使われている場合は、可用性の設定により新しいアプリケーションにリダイレクトさせることが可能になり、以前のブックマークにアクセスするとエラーが発生するといったことを抑制できます。

その他の利用可能な機能

アプリケーション・アイテム

アプリケーション・アイテムをアプリケーション間で共用するには、**有効範囲**を**グローバル**とします。



APEXコレクション

APEXコレクションはアプリケーション間では共用できません。そのため分割したアプリケーション間でデータの受け渡しを行うには、APEXコレクションの代わりに実表を作るなどの対応が必要になります。

APEX_PAGE.GET_URL

ページ遷移のターゲットとなるURLの生成には、APEX_PAGE.GET_URLをできるだけ使用し、APEX_UTIL.PREPARE_URLの使用は避けます。APEX_UTIL.PREPARE_URLを使用してチェックサムをつけるには、引数として指定するURL(p_url)を、文字列として作る必要があります。

APEX_UTIL.HOST_URL

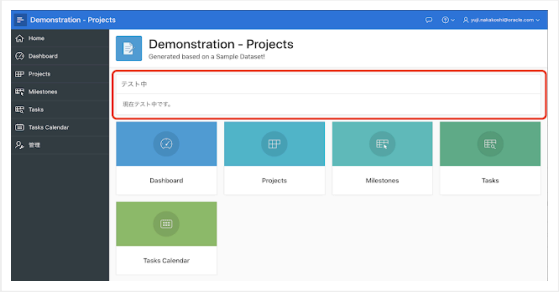
ターゲットを指定する完全なURLを作成する際にAPEX_PAGE.GET_URLが使えない場合は、APEX_UTIL.HOST_URLを使用し、サーバー名などの直書きを避けます。アプリケーションをコピーすることで別名を変更したり、エクスポート/インポートすることで開発環境と本番環境の間でアプリケーションをコピーしたときに、URLを直書きしていると手作業で修正しなければなりません。

毎回APEX_UTIL.HOST_URLを呼び出すことを避けるために、認証後や新規インスタンス(新規セッション)開始時に、アプリケーション・アイテムへアプリケーションの計算によって値を設定しておくこともできます。

ビルド・オプション

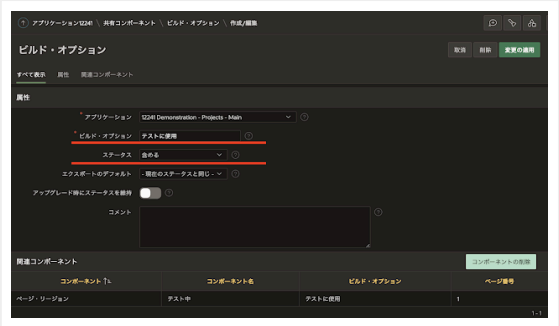
アプリケーションのテストに使用するページ、リージョンなど、本番利用時にはアクセス不可にするためにビルド・オプションが使えます。

例えば、以下のようにテスト中を示す静的コンテンツのリージョンをページに追加します。

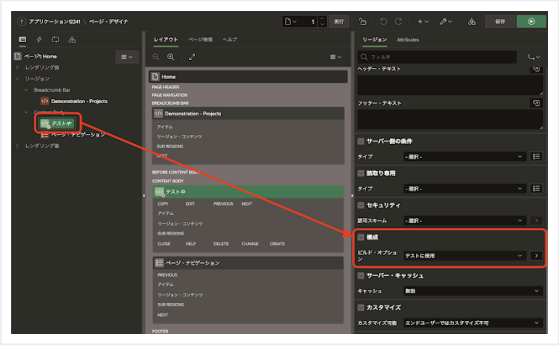


共有コンポーネントのビルド・オプションを開きます。

ビルド・オプションとして**テストに使用**、**ステータス**を**含める**、として作成します。

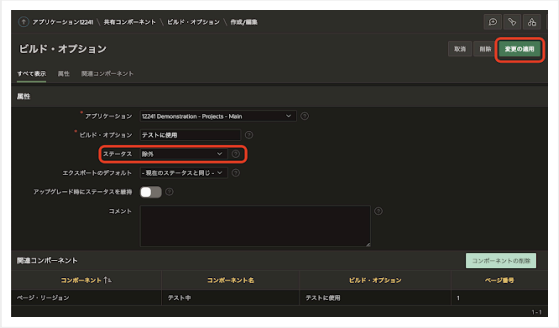


作成したリージョン（リージョンに限らず、ほとんどのコンポーネントにはビルド・オプションの指定があります）の**構成**の**ビルド・オプション**として**テストに使用**を選択します。

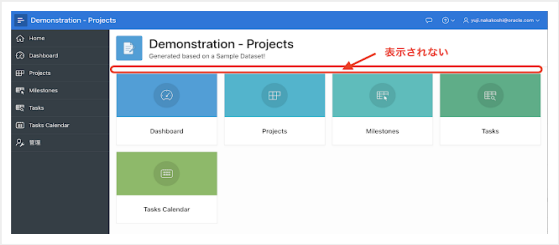


これで**テスト中です。**と表示しているリージョンとビルド・オプションが紐づけられました。

ビルド・オプションの**ステータス**を**除外**に変更します。**変更を適用**します。



この状態で最初のページを表示すると、リージョンが表示されないことが確認できます。



ビルド・オプションによる除外は、サーバー側の条件とは異なり、除外されたコンポーネントはアプリケーションに存在自体していない扱いになります。

ビルド・オプションのその他の機能やSQLclでのLiquibaseの使用など、より進んだOracle APEXのアプリケーションのデプロイメントについては、また記事を改めて紹介したいと思います。

完

Yuji N. 時刻: 15:36

共有

‹ ホーム ›

ウェブ バージョンを表示

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示

Powered by Blogger.