ページ遷移系ガイドライン

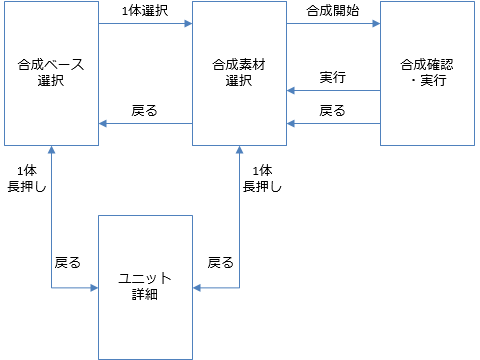
ページをまたぐ制御フローがカオスになりがちなので、ここを重点的にパターン化したい。

# サンプルの題材

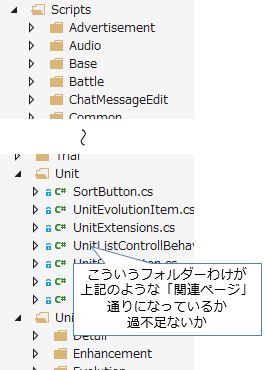
ページ数の的に手頃なので、ユニットの強化合成を題材に。

# ページ遷移仕様

まず、以下のような図を仕様として残してほしい。



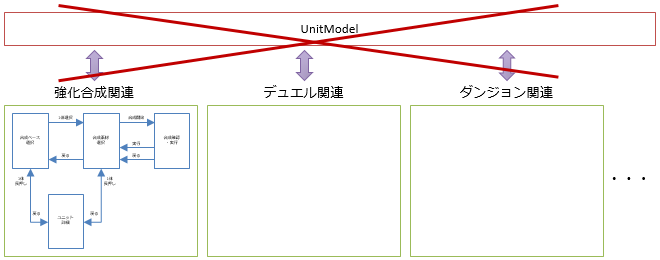
* 「関連」の範囲がわかるようにしたい。この範囲でソースコードのフォルダーも分ける
  + 仕様がちゃんとないと、適切な範囲で区切られてるかのレビューが素通しになりがち
  + フォルダー分けの方針が定まってないと、分ける側もしんどい



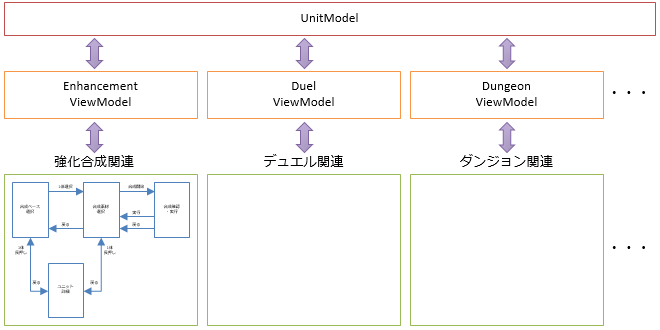
* こういう「関連ページ」を超える範囲でのデータ共有を極力しないでほしい(後述)
* このページ遷移仕様を意識した、ページ遷移のコントロール専用のクラス(navigator)がほしい(後述)

# モデルの債務

ゲーム全体で使うようなモデル(UnitModel)が債務を持ちすぎちゃダメ。



「関連ページ」内でだけ使うデータがあるはずで、そういうものは ViewModel を作って間に挟んでこっちに持つ。



UnitModel はゲーム中どこでも使うプレイヤー情報、例えば以下のようなものがあるべき

* プレイヤーが持っているユニット一覧
* 所持可能なユニット最大枚数

逆に、以下のようなものは、「強化合成関連ページ」にいるときに限り必要なもの(他でも似た情報は作るけど、それは似て非なるもの(共通の子データをもっててもいいけど、別クラス/別インスタンス))なので、ViewModelに持つ

* 選択されたベースユニット
* 選択された素材ユニット
* ソート条件 (手順、レベル順、レアリティ順、…)
* フィルタリング条件 (属性、合成可否、…)

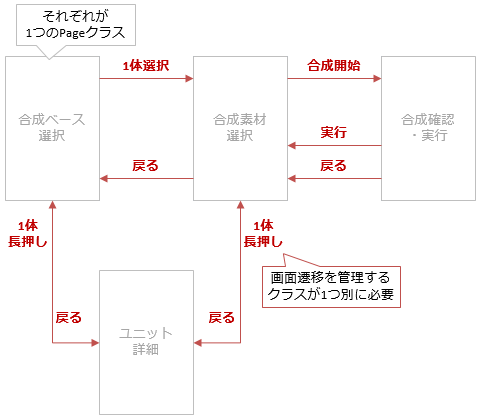
ソート条件とかフィルタリング条件は、ずっと同じものを使い続けたいわけだけども、UnitModelにはpublicプロパティとして持たない。必要なら、ViewModel作るときにサーバーと同期するとか、ローカルストレージからデシリアライズするとか。

# ナビゲーション

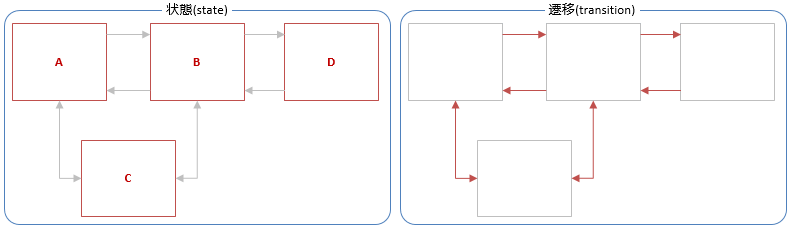
各ページには、ページの中身を統括するクラスがいるとして(よくある造りだと、シーン名+Controller って名前のクラスがそれにあたることが多いけども。個人的には、Pageみたいなクラス名がいいかもと思ってる)。

ページ間の遷移を主に制御するクラスが必要(何でもかんでもControllerって名前でやってしまってたりするけど、分けたい。ほぼページ遷移管理だけを債務とするクラスがほしい。。

名前は、navigator とか navigation manager がいいかも。ページ遷移の管理 = ナビゲーション(運行指示)。



で、これを作るときに意識してほしいのは、ページ遷移フローは、実装的にいうとステートマシンなこと。



要するに

* ある状態にあるときに(ユーザーがあるページを開いているときに)
* どういうトリガーがかかると(ユーザーがどういう操作をすると)
* 何が起きて、次にどういう状態になるか(次にどのページに移動するか)

例えば、合成ベース選択ページ(という状態にあるとき)で言うと

* 「戻る」ボタンを押す(というトリガーを受けて) → 合成終了(「強化合成関連ページ」を抜けて、上位メニューとかに戻る)(という状態に遷移)
* ユーザーが1体タップ → ViewModel中のベースユニットを設定して、素材選択画面に遷移
* ユーザーが1体ホールド → ユニット詳細画面に遷移

# ページの債務

ページ遷移のトリガーを発するだけがページの債務にする。

さっきの「ベース選択ページ」の、

* 「戻る」ボタンを押す(というトリガーを受けて) → 合成終了(「強化合成関連ページ」を抜けて、上位メニューとかに戻る)(という状態に遷移)
* ユーザーが1体タップ → ViewModel中のベースユニットを設定して、素材選択画面に遷移
* ユーザーが1体ホールド → ユニット詳細画面に遷移

でいうと、トリガーの部分(→ 矢印の左側)だけがページの債務。つまり、以下のようなメソッドだけ用意。

* 「戻る」ボタンが押されるのを待つ(待てる口を用意)
* ユニット1体タップされるのを待って、その1体のユニットを返す
* 同上、ホールドを待って、ユニットを返す

コード的にいうと、以下のようなメソッドだけが public (他の、アニメーション付けたりとかは全部 private/internal)。

// 「戻る」ボタン待ち。

public Task AwaitReturn(CancellationToken ct)

// ベースユニット選択待ち。

public Task<UnitWithMaster> AwaitUnitSelect(CancellationToken ct)

// ユニット詳細を開く操作待ち。

public Task<UnitWithMaster> AwaitUnitDetail(CancellationToken ct)

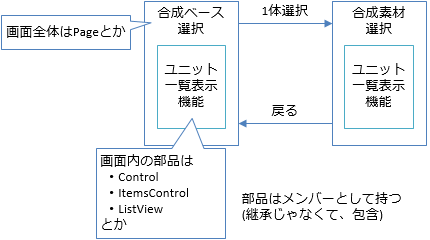
トリガー待ちは、このプロジェクト的には Task を使ってやるのを推奨(過去には event を使ってやるしかなかったけど、今だったらこれが一番コード書くの楽だと思う)。

# ページ内の部品

ちなみに、「ベース選択ページ」と「素材選択ページ」は構成が似てるわけだけども。

よくやってしまいがちな実装方法だと、ユニット一覧系の共通規定クラス(例えばUnitListViewとか)がいて、それを継承して強化合成選択ページ(EnhancementSelectionController)とか進化合成選択ページ(EvolutionSelectionController)とかがいたりするけども、この造りあんまりよくない。継承の乱用。

本当は、「包含」がいい。「ユニット一覧コントロール」みたいなのがいて、それを、ベース選択ページと素材選択ページがメンバーとして持つ。



# サンプルコード

このリポジトリ中のサンプルはコンセプトまとめるために作っただけなので WPF 製で、Unity 上ではそのままは使えない。Unityで使う版も作る予定:

* Task クラスが .NET の標準のやつ → IteratorTasksに要変更。意外と標準TaskからIteratorTasksへの移植は困らないんで大丈夫だと思う
* Frameの代わりに、シーン遷移管理クラスが必要。ISceneChanger とか。こいつのChangeSceneみたいなメソッドが、遷移さきページの ～Pageクラスを返せるようにしたい