iOS 開発の自動化テストとツールによる効率化

このブログ記事は、ChatGPT-4o の協力を得て整理されました。

ユニットテストの重要性

LeanCloud では、プロジェクトの初期段階からユニットテストを実施しており、これが非常に価値あるものだと証明されています。各プルリクエスト（PR）は Jenkins 上でユニットテストをトリガーし、私たちのカバレッジ目標は約 80% です。テストを書く主なシナリオは 2 つあります：新しいインターフェースの検証と、バグの再現と修正です。蓄積されたテストが多ければ多いほど、私たちのコードベースは強固になります。自動化されたテストにより、手動での検証なしに、自信を持ってコードをリリースし、リファクタリングすることができます。

テストプロセスと実際の応用

以下は、ユニットテストがどのように私たちを助けるかの実際の例です：

テストプロセス 1: ユーザーから、description キーを持つオブジェクトを保存する際にエラーが発生するとの報告がありました。この問題を再現するためのテストを書き、問題を見つけて修正しました。そして、将来の検証のためにこのテストを残しました。

テストプロセス 2: 新しいインターフェースを開発する際、実装コードを書いた後にそれに対応するテストを作成し、コードが正常に動作することを確認しました。

テストプロセス 3: `AVObject.m` のコードを変更した後、`AVObjectTest.m` を実行して、変更がテストの失敗を引き起こしていないか確認します。

テストプロセス 4: PR を提出すると、Jenkins 上で自動テストがトリガーされます。

ユニットテストを書くメリット

- **手動検証の削減:** ユニットテストは、手動でのチェックをなくすことで時間を節約します。
- **エラーの検出:** コード変更による問題を早期に検出し、エラーがプロジェクトの他の部分に影響を与えるのを防ぎます。
- **共同プロジェクト:** 複数の開発者が関わるプロジェクトでは、ユニットテストが一貫性と信頼性を保証し、プロジェクトが他の人に引き継がれた場合でも安心です。

- **高品質なオープンソースプロジェクト:** 人気のあるオープンソースプロジェクトは通常、広範なユニットテストを持っており、これが信頼性と人気に寄与しています。

効果的なユニットテストの書き方

- **モジュール化されたコード:** データ層と UI 層を分離してテストしやすくする。
- **カバレッジの最大化:** 最小限のテストコードで最大のカバレッジを実現する。
- **非同期処理:** テストが非同期操作を処理できることを確認する。
- **フレームワークの選択:** ニーズに合ったテストフレームワークを選ぶ。
- **カバレッジレポート:** カバレッジレポートを使用して、コードのどの部分がテストされたかを把握する。

テストフレームワークの評価

いくつかのフレームワークを評価しました：- **Expecta:** `expect(error).not.beNil()` - **Specta:** `describe("") it("")` - **Kiwi:** `describe("") it("")` - TDD および BDD フレームワークにはいくつかの制限があります。例えば、Xcode との統合が不十分で、テストボタンがなく、サイドバーにすべてのユニットテストがリストされていないなどです。

非同期テストの処理

非同期テストは、すぐに完了しない操作にとって重要です。フレームワークが非同期テストを効果的にサポートしていることを確認してください。例えば、XCTest では expectations を使用して非同期操作が完了するのを待ち、それからアサーションを行います。

カバレッジレポート

Xcode 7 には、組み込みのカバレッジレポート機能が導入されました。有効にする手順は以下の通りです：1. スキーム設定で `Gather Coverage Data` を有効にします。2. Test Target ではなく、App Target に対してテストを行います。

この機能により、開発者はどのコード行がテストされているかを正確に確認でき、未テストのコード部分を特定するのに役立ちます。詳細については、Big Nerd Ranch のブログをご覧ください。

Jenkins を使用したリモート自動テスト

Jenkins を自動テストに設定するには、いくつかの手順があります：1. **Jenkins のインストール:** ローカルマシンまたはデータセンターサーバーに Jenkins をセットアップします。2. **GitHub 統合:** GitHub PR ビルドプラグインを使用して、プルリクエストが提出されたときにテストをトリガーします。- Webhook を設定して、イベントを Jenkins に送信します。- Jenkins がプルリクエストの最新コードにアクセスできることを確認します。3. **テストスクリプト:** Jenkins 内でテストスクリプトを設定し、テストプロセスを自動化します。- Jenkins が GitHub にテスト結果を通知できることを確認します。- テストが失敗した場合に Slack やメールで通知するように設定します。

Jenkins を使用したリモートでの自動化テストは、クリーンで制御された環境でテストを実行することにより、ローカルテストを超える自動化テストのすべての利点を提供します。

リモートでのパッケージングとデプロイ

すべてのプロジェクトでリモートパッケージングが必要なわけではありませんが、SDK やその他の再利用可能なコンポーネントのデプロイプロセスを簡素化することができます。手順は以下の通りです：- Jenkins がコードを読み取るように設定します。- リリースバージョンを読み取ります。- コマンドラインでキーチェーンをロック解除して署名証明書にアクセスします。

追加ツールとヒント

- **Reveal:** 脱獄した iPhone 上で任意のアプリの UI インターフェースを分析する。
- **Flex:** 脱獄した iPhone 上でネットワークリクエスト、UI、ローカルファイル、NSUserDefaults、ログを分析する。
- **Pod 管理:** ローカル Pod の使用、高度な Podfile 設定、および Pods の公開。
- **フレームワーク作成:** 動的ライブラリと静的ライブラリの違い、およびシミュレータと実機の両方に対応するフレームワークのパッケージング方法。
- **Xcode ヒント:** 便利なショートカット、例えば Shift + Command + J でナビゲーターにファイルを表示、Shift + Command + O でファイルを素早く開く。

結論

自動化テストと適切なツールは、開発プロセスを大幅に向上させます。早期にユニットテストを導入し、非同期処理を活用し、カバレッジレポートを使用することで、より信頼性が高く保守

しやすいアプリケーションを構築できます。Jenkins のような CI/CD ツールと Xcode 開発ツールを組み合わせた強力なテスト戦略により、高品質なソフトウェアの提供が保証されます。

謝辞

LeanCloud チームと、私たちのテストプロセスに貢献してくださったすべての方々に、特に感謝を申し上げます。