Instruments新機能ユーザガイド

Tools > Performance

2011-10-13

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
U.S.A.

アップルジャパン株式会社
〒163-1450 東京都新宿区西新宿
3 丁目20 番2 号
東京オペラシティタワー
http://www.apple.com/jp/

目次

図、リスト

はじめに

『*Instruments New Features User Guide*』では、Instruments 4の主な新機能を簡単に紹介します。

At a Glance

Instruments 4は、iOSやMac OS Xのコードを動的にトレース（動作を追跡）し、プロファイリング（処理性能を分析）するアプリケーションの最新版です。InstrumentsはXcodeに付属しています。これは、iOSやMac OS Xで動作するアプリケーションの構築に必要な、あらゆる機能を備えたツールセットです。

Prerequisites

この文書の内容を理解するためには、旧版のInstrumentsに備わっていた基本的な機能について知っておく必要があります（『*Instruments User Guide*』を参照）。

Instruments 4.0の新機能

この章では、Instruments 4.0の主な新機能を簡単に紹介します。

データマイニングにおける表示のしぼり込み機能

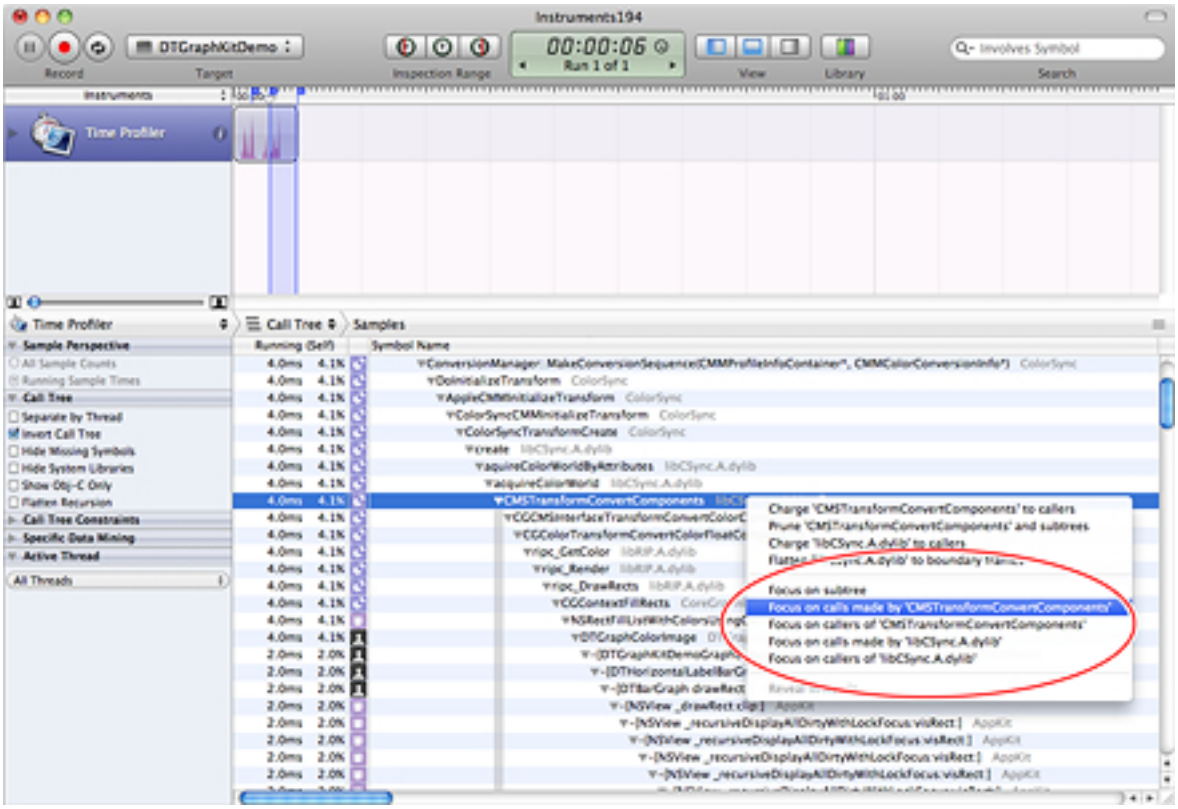
呼び出しツリーを対象とするデータマイニングにおいて、呼び出しツリーの特定の部分に関する情報を選択表示できるようになりました。この機能は、呼び出しツリーシンボルのコンテキストメニューから起動できます。

呼び出しツリーのあるシンボルを対象とするコンテキストメニューは、次のように操作すると現れます。

1. 呼び出しツリーを表示します。
2. Controlキーを押した状態にします。
3. そのまま該当するシンボルを選択（クリック）してください。

この機能に関係するコマンド名はすべて、先頭が「Focus on」となっています（図 1-1を参照）。

図 1-1 データマイニングにおける表示のしぼり込み機能



関係するコマンドは5つあり、いずれも呼び出しツリーのある特定の部分に着目して表示するために使います。

- Focus on subtree

  選択したシンボル以下のサブツリーのみを表示します。

- Focus on calls made by <symbol name>

  選択したシンボルからの呼び出しによるサブツリーのみを表示します。

- Focus on callers of <symbol name>

  選択したシンボルの呼び出し元が入っているサブツリーのみを表示します。

- Focus on calls made by <library name>

  選択したシンボルと同じライブラリに属する、各シンボルからの呼び出しによるサブツリーのみを表示します。

- Focus on callers of <library name>

  選択したシンボルと同じライブラリに属する、各シンボルの呼び出し元が入っているサブツリーのみを表示します。

DTPerformanceSessionフレームワーク

新たに加わったDTPerformanceSessionフレームワークをMac OS Xアプリケーションに適用すれば、性能データを収集し、Instruments上で開いて表示できるようになります。DTPerformanceSessionに付属するCのAPIを介して、Time Profiler、Allocations、Leaks、Activity Monitorなどのinstrumentsが使えます。今後、ほかのinstruments用のAPIを追加していく予定です。自分が開発している範囲以外のプロセスも対象にすることができます。APIを使って、Core Foundationを利用する独自のツール（回帰テストハーネスなど）を開発することも可能です。

DTPerformanceSessionの出力形式は.dtpsバンドルなので、Instrumentsで開き、必要な機能を完備したトレースドキュメントを作成できます。バンドル自身にはシンボル情報が含まれていないため、アプリケーションを構築し直せば使えなくなります。トレースデータを.dtpsバンドルに保存したい場合は、Instrumentsで開き、トレースドキュメントとして保存してください。

DTPerformanceSessionには*iprofiler*というコマンドラインツールが付属しています。iprofilerには次のような機能があります。

- 動作中のプロセスにアタッチ
- 性能測定時にのみ動作するプロセスの起動
- 動作中のプロセスすべてのプロファイル取得

iprofilerについて詳しくはmanページを参照してください。

DTPerformanceSessionを利用するためには、アプリケーションにこのフレームワークをリンクする必要があります。DTPerformanceSessionフレームワークは次の位置にあります。

```
/Library/Developer/4.0/Instruments/Frameworks/DTPerformanceSession.framework
```

アプリケーションのコードでは、次のように記述してAPIのヘッダファイルをインポートします。

```
#import <DTPerformanceSession/DTPerformanceSession.h>
```

リスト 1-1にDTPerformanceSessionの使用例を示します。

リスト 1-1　DTPerformanceSessionの使用例

```
CFStringRef process = CFStringCreateWithFormat(kCFAllocatorDefault, NULL,
CFSTR("%d"), getpid());
CFErrorRef error = NULL;
DTPerformanceSessionRef session = DTPerformanceSessionCreate(NULL, process,
NULL, &error);
DTPerformanceSessionAddInstrument(session,
(CFStringRef)@DTPerformanceSession_TimeProfiler, NULL, NULL, &error);
CFMutableArrayRef instrumentIDs = CFArrayCreateMutable(kCFAllocatorDefault, 0,
 &kCFTypeArrayCallBacks);
CFArrayAppendValue(instrumentIDs, @DTPerformanceSession_TimeProfiler);
DTPerformanceSessionStart(session, instrumentIDs, &error);

// アプリケーションの処理

DTPerformanceSessionStop(session, instrumentIDs, &error);
DTPerformanceSessionSave(session, (CFStringRef)@"/tmp/myAppProfile", &error);
```

```
DTPerformanceSessionDispose(session, &error);
```

System Trace

System TraceはInstruments用の新しいテンプレートです。いくつかのinstrumentを組み合わせて、Mac OS X側が実行する、アプリケーション性能に影響を与えうる処理のプロファイルを、さまざまな観点から収集できます。各instrumentは、システムコール、スレッドのスケジューリング、仮想メモリ（VM）処理に関する情報を収集します。たとえばゲームを開発していて、想定外の位置でフレームレートが落ちる、所定の時間内に処理が実行されない、などの現象に悩まされている場合、System Traceが原因の特定に役立つでしょう。

System Traceには*ストラテジー制御*バーという新しいUI要素が組み込まれています（トラックペインのすぐ上）。このバーの左端にあるボタンで、トラックを表示するストラテジーを切り替えることができます。

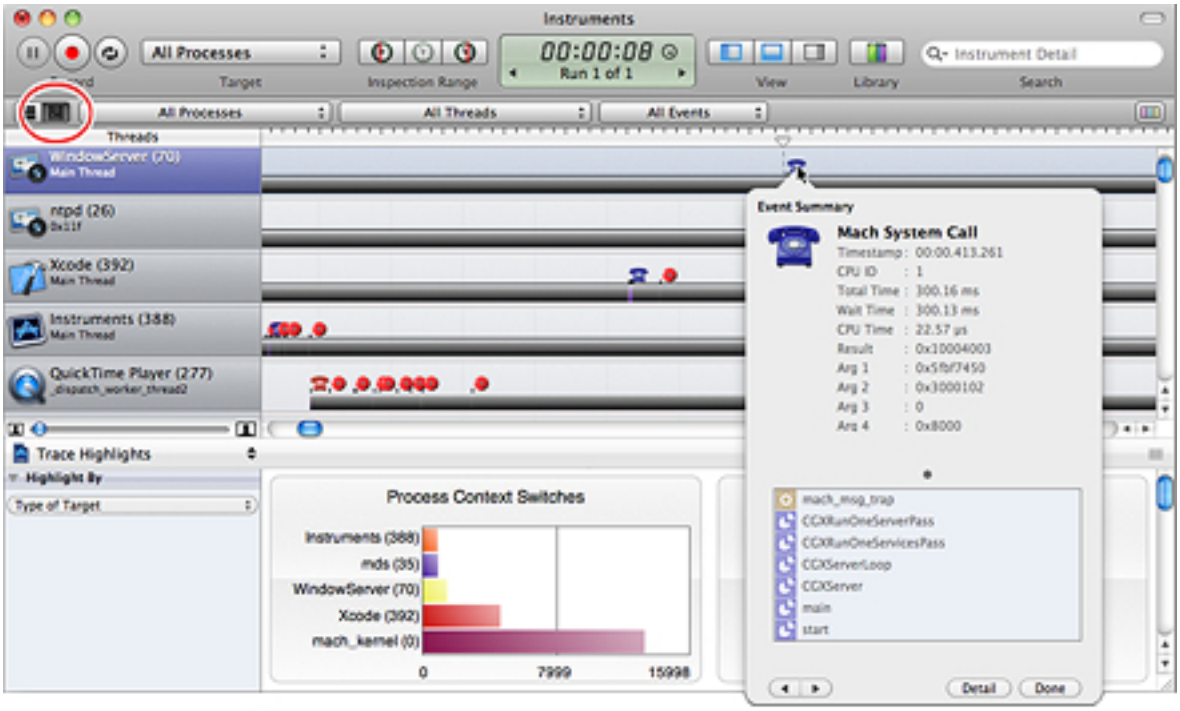
- Instrumentsストラテジー（1つ目のボタン）を選択すると、各instrumentが収集したイベントの単位時間当たり発生率が、専用のトラックに表示されます。これがデフォルトのストラテジーです。
- Threadsストラテジー（2つ目のボタン）を選択すると、個々のスレッドのスケジューリング状態が、主要なイベントをマーカで示す、という形で表示されます。

  個々のinstrumentが収集した情報を、該当するスレッドのトラックに振り分けて表示するようになっています。ある時点で発生した特定のイベントについて詳しく知りたい場合は、該当するマーカをクリックしてください。トラックの色によって、スレッドのスケジューリング状態が分かります。ストラテジーバーの右端にあるボタンを押すと、それぞれの色の意味が表示されます。

詳細ビューには、（イベントの）標本データに加え、有用なグラフ群を集めた「Trace Highlights」が表示されます。

図1-2に、Threadsストラテジーにもとづき表示したトレースドキュメントの様子を示します。「Threadsストラテジー」のボタンを押した状態になっていることに注意してください。

図 1-2 Threadsストラテジーで表示したSystem Traceの様子

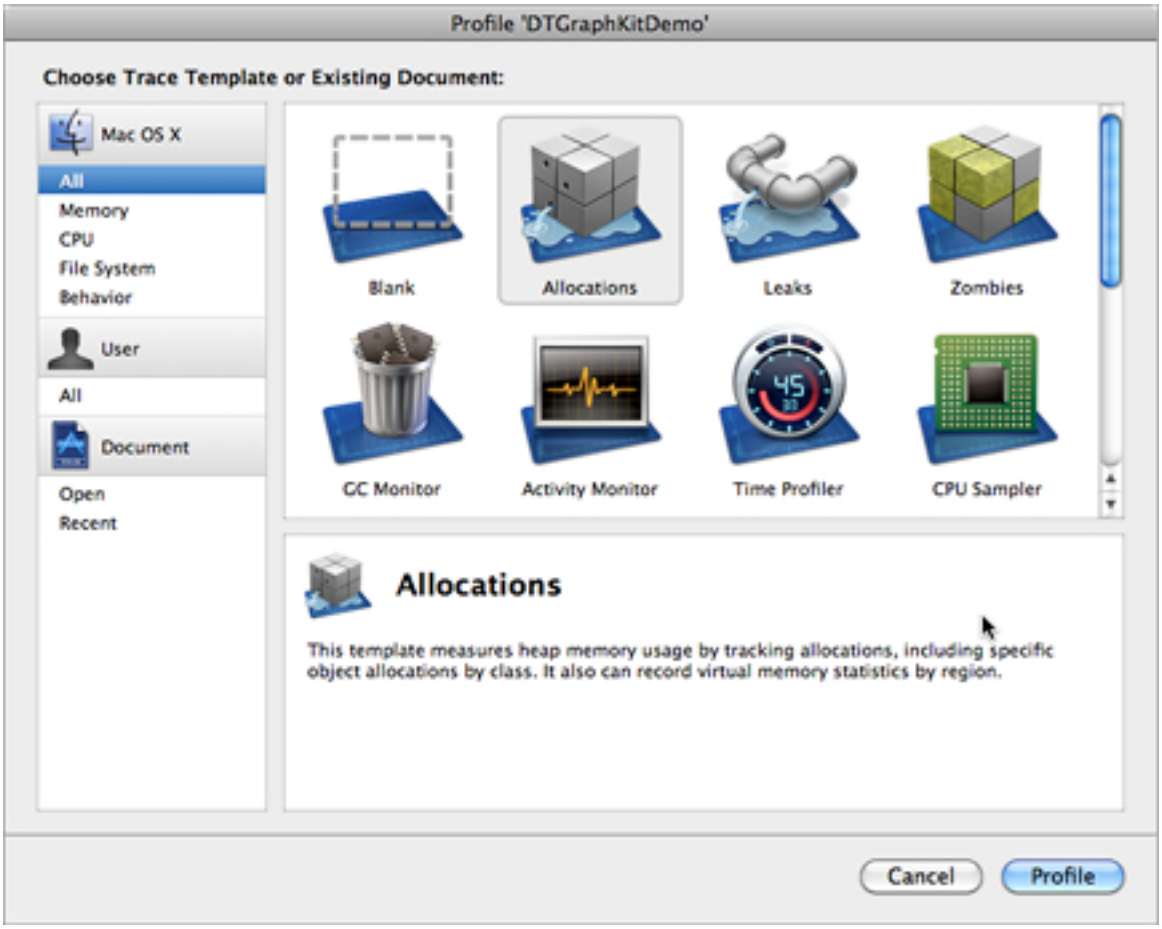


Xcodeからの起動

Xcode 4とInstruments 4を統合しました。Xcode上からInstrumentsを起動して、あるXcodeターゲットのプロファイルを取得することができます（「Product」＞「Profile」コマンド、またはCommand-I）。新規プロジェクトを作成してターゲットのプロファイルを取得する場合、特に指定しなければ、XcodeはDWARF形式でリリース用のターゲットを構築し、dSYMファイル形式でデバッグシンボルを出力するようになっています。

Instrumentsを起動する際、ターゲットのプロファイル取得に用いるテンプレートの選択ダイアログが表示されます（図 1-3を参照）。

図 1-3 「Profile」コマンドを実行すると表示されるダイアログ



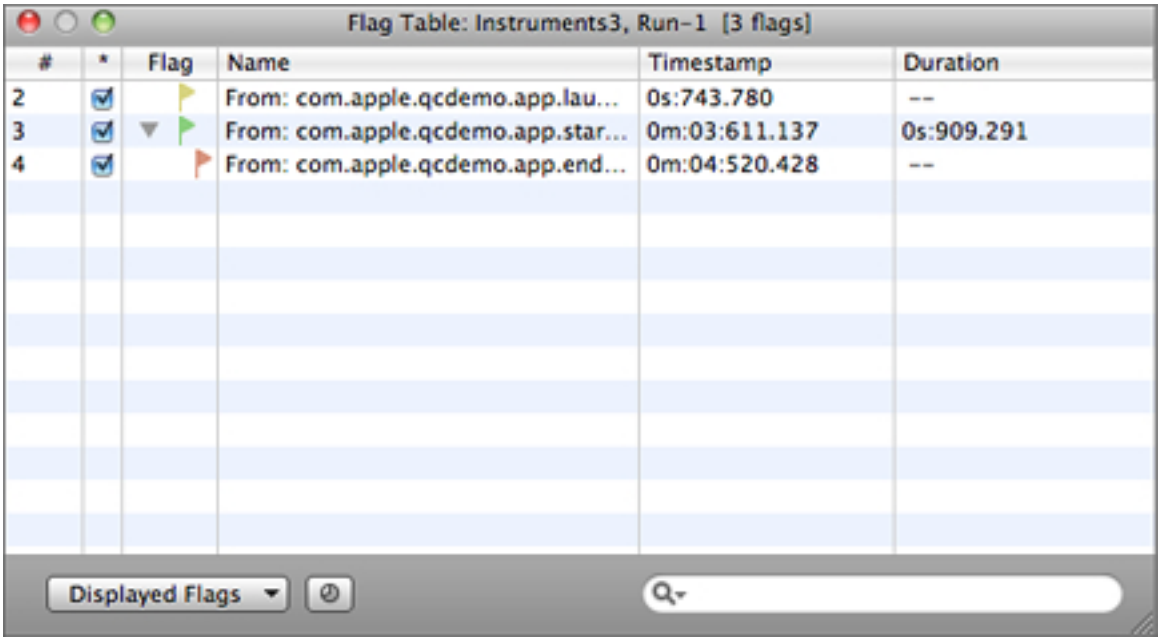
「Profile」ダイアログの外観は、Finder上でInstrumentsを起動したときに現れるダイアログに似ています。ここには、Xcodeのターゲットに応じ、適用できないテンプレートを除いて表示されます。「Profile」ダイアログの「Document」セクションには、「Open」と「Recent」という選択肢があります。Instrumentsはターゲットごとに、最近保存したほかのドキュメントを覚えており、新たにプロファイルを保存する際、その中から選択できるようになっているのです。

トレースデータを記録した後、生成されたトレースドキュメントを開いたまま、再びプロファイルを取得すると、Instrumentsは、新たに収集したデータを同じトレースドキュメントに追加するものと想定します。「Profile」コマンドの設定は、Xcodeプロジェクトのアクティブなスキームを編集することにより変更できます。スキームの編集方法や「Profile」コマンドの設定変更については、『*Xcode 4 User Guide*』の「Building and Running Your Code」を参照してください。

フラグテーブル

Instrumentsは従来から、トラックペインにさまざまなフラグを表示しています。Instruments 4.0には、各種のフラグを一覧表示し、検索するためのウインドウが追加されました。このフラグテーブルは、「View」＞「Flag Table」コマンド、またはShift-Command-Tで表示できます。

図 1-4 フラグテーブル



実際に表示するフラグの種類をチェックボックスで選択できます。「Displayed Flags」メニューでも同じ選択が可能です。

このメニューの右側にあるボタンは、時間によるしぼり込みに使います。このボタンを押して2つ以上のフラグを選択すると、最初に選択したフラグの時刻から、最後の選択したフラグの時刻までがしぼり込み範囲になります。これにより、特定の時間帯に収集したデータのみ表示することができるのです。

さらに、フラグテーブルのコンテキストメニューでも、表示するフラグの切り替えや時間によるしぼり込みの実行が可能です。

シグナルフラグ

Instruments 4.0では、ソースコード内にマクロを記述することにより、アプリケーション実行中の特定のタイミングで、シグナルフラグを送出できるようになりました。マクロはユーザ空間で実装されており、非常に使いやすい設計です。Instruments上でプロファイルを取得すると、シグナルフラグに関するデータも記録されます。トレースドキュメントの時間軸に沿ってシグナルフラグを表示したい場合は、フラグテーブルを開き、「Displayed Flags」メニューの「Signal Flags」をオンにしてください。

シグナルフラグ用のマクロは、DTPerformanceSessionフレームワークのヘッダファイルに定義されています。マクロを使うだけならば、アプリケーションにDTPerformanceSessionフレームワークをリンクしなくても構いません。しかしリンクすれば、次のように記述することにより、コード中にヘッダをインポートできます。

```
#import <DTPerformanceSession/DTSignalFlag.h>
```

リスト 1-2に、DTSendSignalFlag関数を呼び出してシグナルフラグを送出する記述例を示します。フラグ名はユーザが自由に定義でき、フラグテーブルにはこの名前が表示されます。

リスト 1-2　DTSignalFlagの例

```
// 時点フラグ（ある特定の時点のイベント）
DTSendSignalFlag("com.mycompany.mytracepoints.app.point", DT_POINT_SIGNAL, TRUE);

// 開始点フラグ（何かの開始時点のマーク）
DTSendSignalFlag("com.mycompany.mytracepoints.app.start", DT_START_SIGNAL, TRUE);

// 終了点フラグ（何かの終了時点のマーク）
DTSendSignalFlag("com.mycompany.mytracepoints.app.end", DT_END_SIGNAL, TRUE);
```

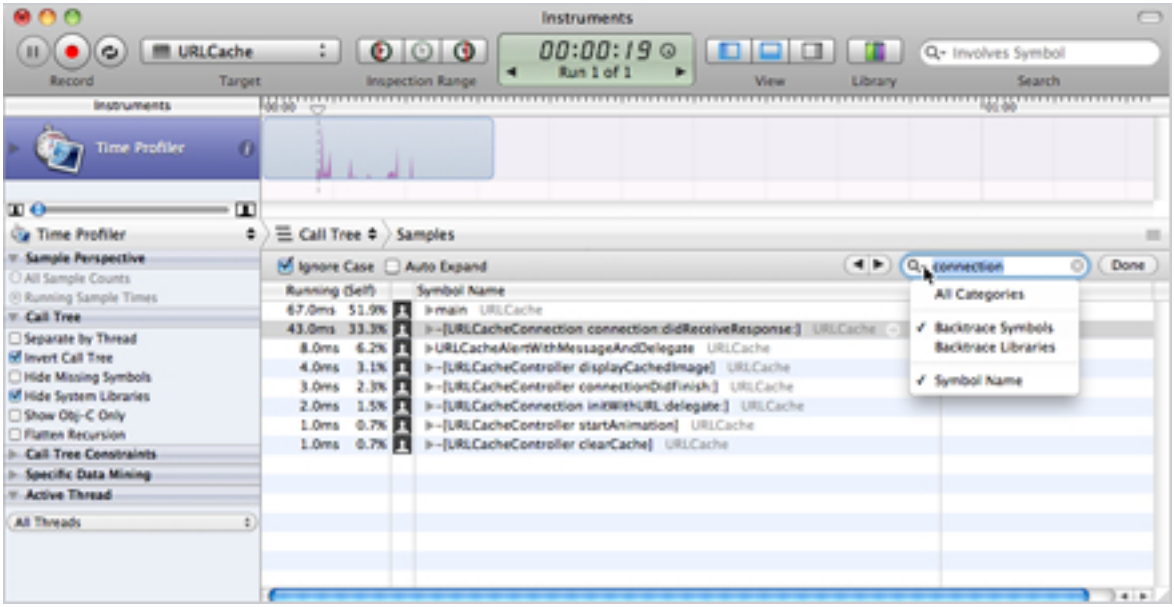
検索フィールド

Instrumentsには従来から、詳細ビューのテキストデータをしぼり込む、検索フィールドが付属していました。これを使うと、検索条件に合致しないデータを非表示にすることができます。検索フィールドはツールバーの右端にあります。

Instruments 4.0では、詳細ビューの検索フィールドに新しい機能が組み込まれました。前後のデータ（文脈）を表示したまま、条件に合致するデータを強調表示する、というものです。合致しない部分も非表示にはなりません。

検索フィールドは、「Edit」＞「Find」＞「Find」コマンドを実行（またはCommand-F）すると現れます。検索語は検索フィールドに直接入力します。条件に合致する箇所が複数ある場合、矢印ボタンで順次移動できます。検索フィールドの「虫眼鏡」をクリックすると検索メニューが開きます。ここで、検索対象とする欄その他の検索条件を指定できます（図 1-5を参照）。

図 1-5　検索操作



検索終了後、「Done」を押すと検索フィールドは消えます。

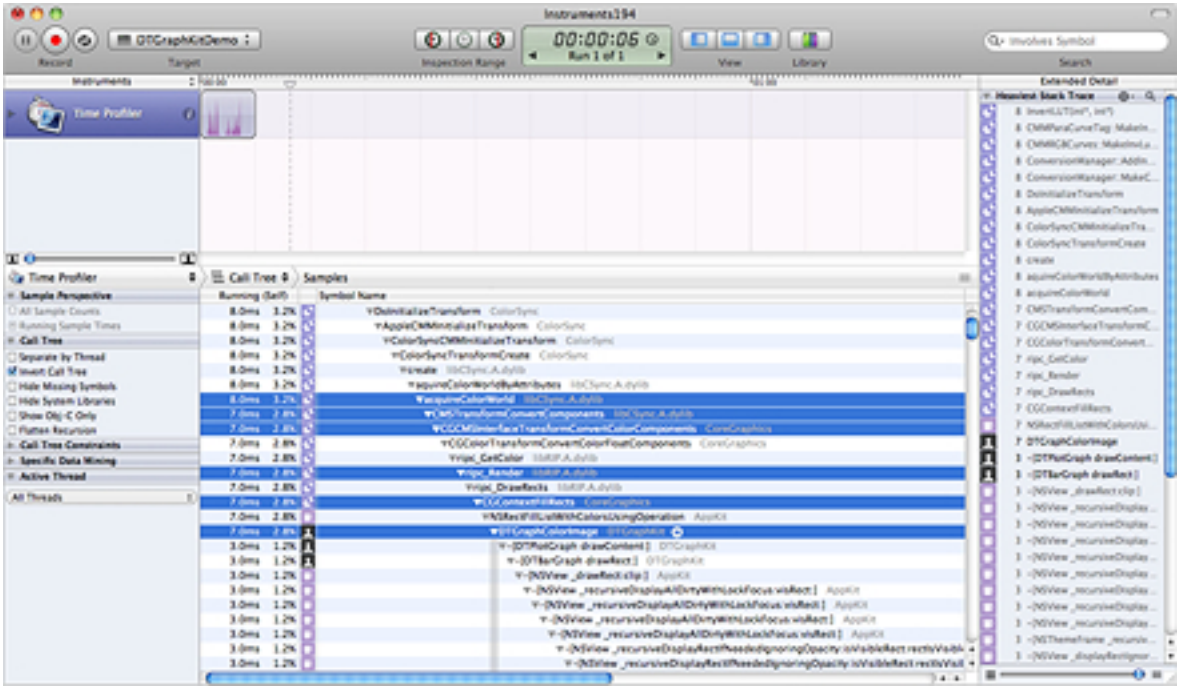
テーブル中の行のコピー

Instruments 4.0では、サンプルリスト、詳細ビューの呼び出しツリーなど、各種テーブルの任意の行を選択、コピーできるようになりました。注目して欲しいデータのみをバグレポートに転記し、あるいは電子メールで送信する際に便利でしょう。

Commandキーを押しながら該当する行を順次クリックすると選択状態になります（複数選択も可）。Command-Cを押すと、この行のテキストをクリップボードにコピーできます。呼び出しツリーで、あるノード以下のサブツリー全体をコピーしたい場合は、根のノードを選択し、Shift-Command-Cを押してください。

図 1-6に、呼び出しツリーの行を選択している様子を示します。連続していない行も選択されていることが分かります。

図 1-6　テーブルの行を選択している様子



このデータをコピーしてテキスト文書に貼り付けると、次のようになります。

```
Running (Self)        Symbol Name
8.0ms    3.2%                  acquireColorWorld
7.0ms    2.8%                   CMSTransformConvertComponents
7.0ms    2.8%                  CGCMSInterfaceTransformConvertColorComponents
7.0ms    2.8%                       ripc_Render
7.0ms    2.8%                         CGContextFillRects
7.0ms    2.8%                           DTGraphColorImage
```

ドラッグジェスチャによる拡大/縮小と時間範囲指定

Instruments 4.0では、発生するイベントをナノ秒単位でトレースできます。短い時間目盛でも分かりやすくイベントを表示するため、トラックビューの表示精度を増しました。さらに、ドラッグジェスチャにより表示範囲を拡大/縮小し、時間範囲を指定できるようにしました。新しいトレースドキュメントを開いた後、トラックビューの下部に表示される、次のような指示に従って操作してください。

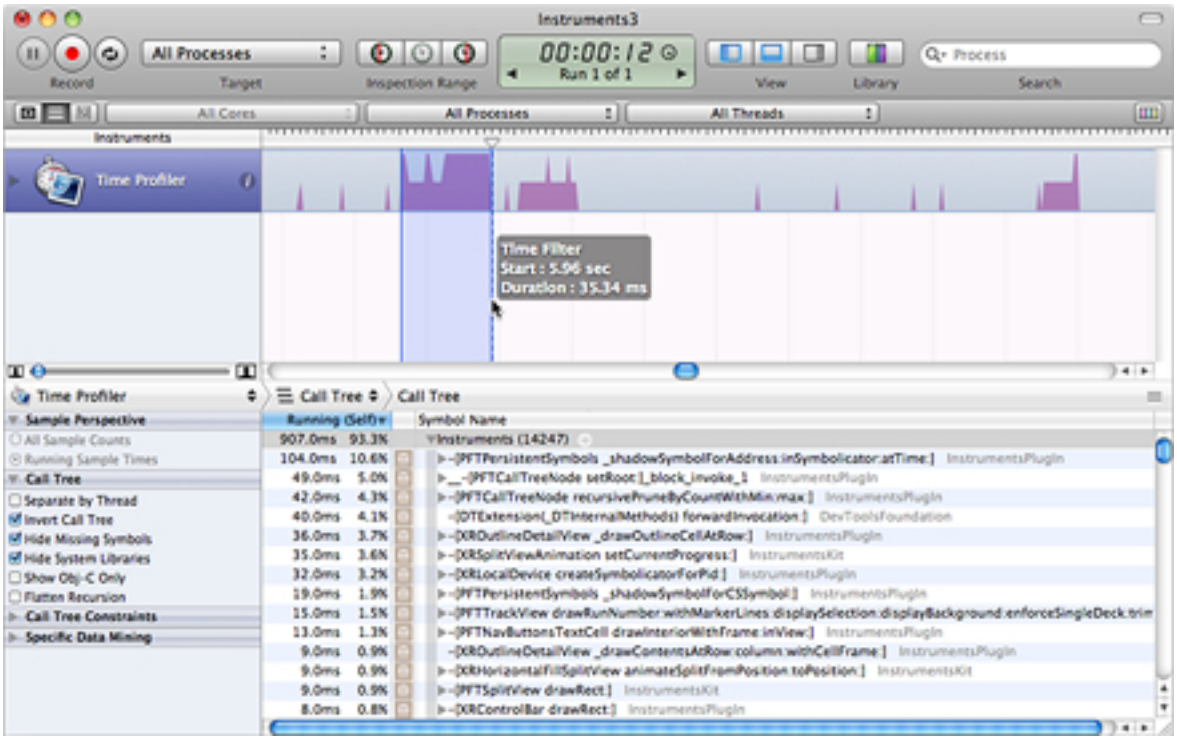
- Shift-drag to zoom in
- Control-drag to zoom out
- Option-drag to define a time filter

たとえば「Shift-drag」とは、Shiftキーを押したまま、見たい領域付近にマウスカーソルを置いてマウスボタンを押し、そのまま表示したい領域にわたってドラッグした後、マウスボタンを放す、という意味になります。このように操作すると、囲んだ領域が拡大されて画面全体に広がり、時間軸の目盛もそれに合わせて調整されます。ドラッグ中は、目盛がどう変わるか、ピクセル当たりのミリ秒数/マイクロ秒数/ナノ秒数の形で表示されます。

同様に「Option-drag」とは、Optionキーを押したまま、見たい領域付近にマウスカーソルを置いてマウスボタンを押し、そのまま表示したい領域にわたってドラッグした後、マウスボタンを放す、という意味になります。このように操作すると、表示する時間範囲を指定できます。ドラッグ中は、開始時刻と長さが表示されます。マウスボタンを放すと、詳細ビューにはその時間範囲内に収集したサンプルが表示されます。

図 1-7に、ドラッグジェスチャにより、トレースドキュメントに表示する時間範囲を指定している様子を示します。

図 1-7 ドラッグジェスチャにより時間範囲を指定している様子



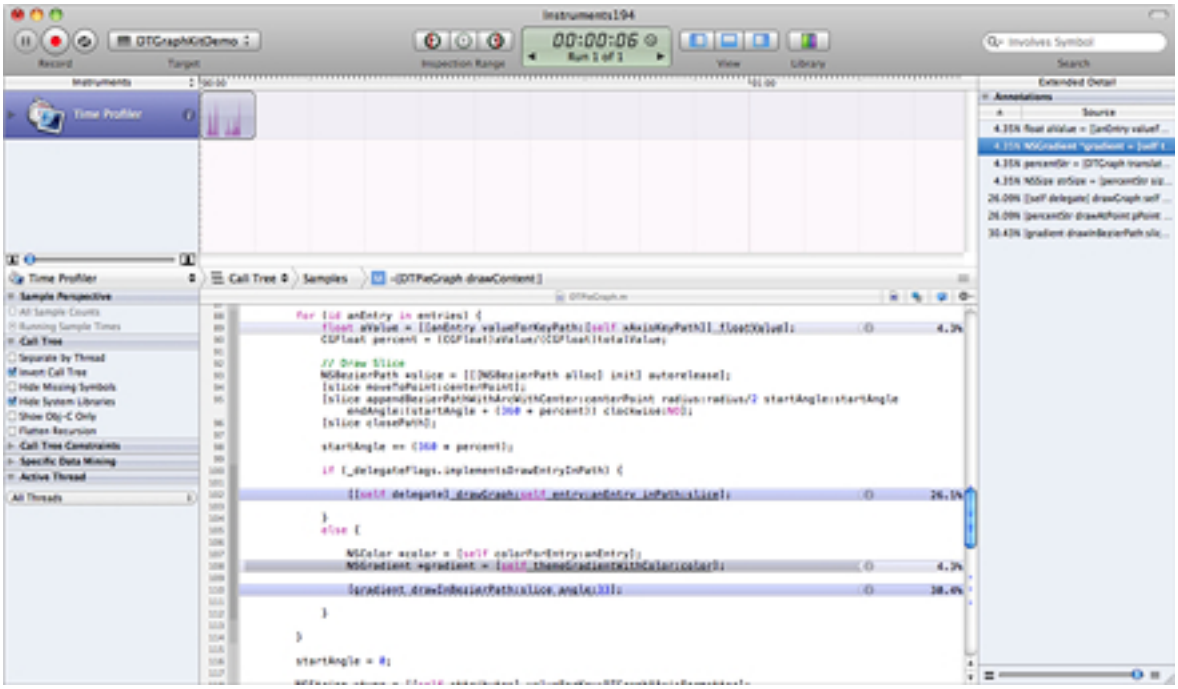
一般に、関心のある領域付近を拡大した後、時間範囲を精密に指定して、その間の呼び出しツリーサンプルを表示する、という手順になるでしょう。

ソースコードのアノテーションテーブル

Time Profiler instrumentを使って、アプリケーションがどの処理に多くの時間を費やしているか調べると、Instrumentsはソースコードに、有用な情報をアノテーション（注釈）としてつけ加えます。Instruments 4.0では、詳細ビューにソースコードを表示する際、拡張詳細ビューにアノテーションテーブルを表示できるようになりました。このテーブル上で、アノテーションの並べ替え（昇順/降順）が可能です。したがって、ソースコード上で、最も処理時間を費やしている箇所を容易に見つけることができます。また、テーブル上でアノテーションを選択すると、対応するコードが詳細ビューに表示されます。

図1-8に、テーブル上でアノテーションを選択し、対応するコードを表示している様子を示します。

図 1-8 アノテーションテーブルを使っている様子



Instruments 4.1の新機能

この章では、Instruments 4.1の主な新機能を簡単に紹介します。

DTPerformanceSessionとSystem Trace

Instruments 4.1では、DTPerformanceSessionフレームワークで、System Traceのプロファイルも収集できるようになりました。コマンドラインツールiprofilerはDTPerformanceSessionを使って実装されていますが、これもSystem Traceのプロファイル収集に対応しています。

DTPerformanceSessionはInstruments 4.0で新たに追加されたフレームワークで、Mac OS Xアプリケーションの性能データを、CのAPIを使って記録します。詳しくは「DTPerformanceSessionフレームワーク」（11 ページ）を参照してください。

遅延表示モード

Instruments 4.1では、いつでも遅延表示モードが使えるようになりました。このモードにすると、トレースデータをすべて記録してしまってから、トレースドキュメントのデータを分析、表示するようになります。これはアプリケーションの性能測定時に有用です。記録フェーズの間、Instrumentsアプリケーション自身のオーバーヘッドを最小限に抑えることができるからです。

常に遅延表示モードにしたい場合は、「General」ペインで「Instruments」＞「Preferences」を実行して環境設定画面を開き、「Always use deferred mode」をオンにしてください。

Time Profilerのストラテジー

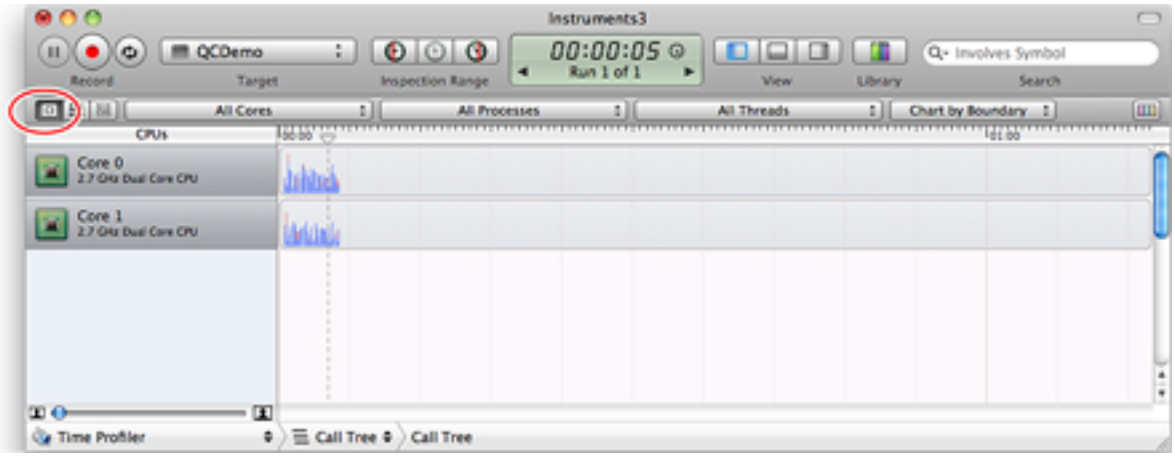
Instruments 4.1では、Time Profiler instrumentにストラテジー制御バーがつきました。これは、Instruments 4.0で、System Trace用に新たに組み込まれたUI要素です。

Time Profilerでは、このバーの左端にあるボタンで、トラックビューペインの表示を3通りに切り替えることができます。

- CPUストラテジー（左のボタン）を選択すると、アクティブなコアごとにCPUの稼働状況が表示されます。アプリケーションが並行動作しているかどうかを判断するために有用です。
- Instrumentsストラテジー（中央のボタン）を選択すると、CPUの稼働状況が1つのトラックにまとめて表示されます。これがデフォルトのストラテジーです。
- Threadsストラテジー（右のボタン）を選択すると、CPUの稼働状況がスレッドごとに表示されます。

図 2-1に、CPUストラテジーでトレースドキュメントを表示している様子を示します。

図 2-1 CPUストラテジー



iPad 2用のアプリケーションを開発する場合、Time ProfilerをCPUストラテジーにして使い、CPUコアごとの稼働状況を調べるとよいでしょう。並行動作に対応したアプリケーションでは、iPad 2の各コアが実際に並行稼働していることを確認する必要があります。

CPUストラテジーで表示できるのは、現状ではTime Profilerに限ります。

プロセッサのコア数を制限する設定

Instruments 4.1では、アクティブなプロセッサコア数を制限できるようになりました。コア数が少ない以外は同じ構成のシステムで、アプリケーションの動作がどのようになるか確認できます。たとえば、手元にあるMacBook Proのアクティブなコア数が4であっても、これを2に制限することにより、コア数が2のMacBook Proで実行した場合のプロファイルを取得できるのです。

ハードウェアによるマルチスレッド処理（ハイパースレッディング）に対応したCPUの場合、物理コアそれぞれに対応して、もう1つ論理コアがあります。したがってこの場合、たとえば物理コアが4つのシステムには、全部で8つのコアがあることになります。

アクティブなコア数の設定は、「Instruments」＞「Preferences」コマンドで環境設定画面を開いて行います。ハードウェアによるマルチスレッド処理に対応したシステムであれば、「General」ペインで、「Hardware Multi-Threading」のオン/オフを切り替えてください。次に、スライダを操作してコア数を設定します。

この設定は、環境設定を変更するか、システムを再起動するまで有効です。

注意： アクティブなコア数を変更しても、実際にプロセッサコアを切るわけではありません。アクティブでないコアには処理を割り当てないよう、システムに指示するだけです。

トラックの表示幅に関する設定

Instruments 4.1では、トラックペインの幅に合わせて、トラックの大きさを自動的に変更する設定ができるようになりました。アプリケーションを実行する都度、大きさが調整されます。

この設定は、「Instruments」＞「Preferences」コマンドで環境設定画面を開いて行います。「Display」ペインで「Always snap track to fit at end of run」をオンにしてください。

なお、これがオフであっても、トラックを選択して「View」＞「Snap Track to Fit」を実行することにより、手動で大きさを調整することは可能です。

Cocoa Layout

Cocoa Layoutは新たに加わったinstrumentで、`NSLayoutConstraint`オブジェクトに施された変更を、実行時に監視できます。Cocoa Autolayoutシステムに関連する問題のデバッグ作業の一貫として、このinstrumentを使うことになるでしょう。AutolayoutシステムはMac OS X v10.7以降使えるようになった新しい機能です。詳しくは『*Cocoa Auto Layout Guide*』を参照してください。

全画面表示

Instruments 4.1は、OS X v10.7で新たに加わった全画面表示オプションに対応しました。

全画面表示モードへは、「View」＞「Full Screen」をオンにするか、またはトレースドキュメントの右上隅にある「Enter Full Screen」ボタンで切り替えます。このモードから抜けるには、「View」＞「Full Screen」をもう一度選択してオフにするか、メニューバーの右端にある「Exit Full Screen」ボタンを押してください。

「Flag Table」メニューコマンド

フラグテーブルを表示するためのメニューコマンドが「Window」＞「Manage Flags」に変わりました。キーボードで操作する場合は、従来通りShift-Command-Tです。

標識イベント

Instrumentsアプリケーションにはシステムレベルのさまざまなinstrumentが付属していますが、それでもなお足りないと感じることがあるでしょう。Instruments 4.1では、コード中に「標識(signpost)」を埋め込んで、システムレベルのイベントを送出できるようになりました。長時間にわたってトレースを取得する、ある操作に要する時間を測定する、などの場合に、所定の時点の目印として使うことができます。イベントを高頻度で送出する、あるいはカーネル拡張のイベントを送出する場合、シグナルフラグを使うよりも便利です。

Instrumentsは次の2種類の標識に対応しています。

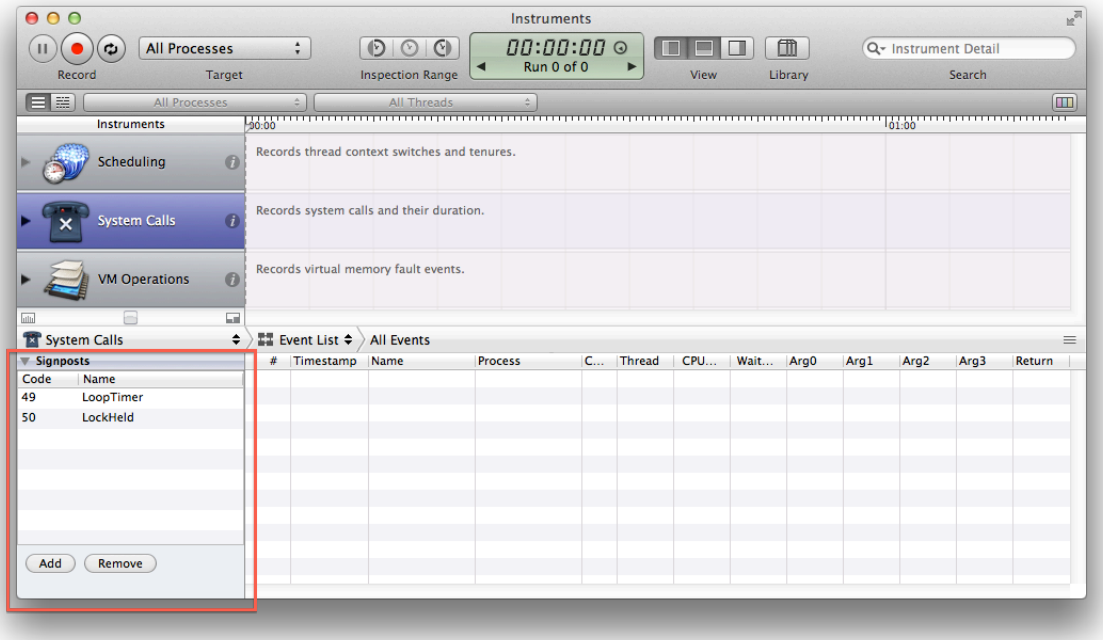
- 時点イベント（時間の長さなし）
- 間隔イベント（開始点と終了点を設定）

時点イベントは、長時間にわたるトレースで、時間軸上の所定の位置を表す目印として使います。たとえば、ビデオフレームを操作するアプリケーションを開発している場合、新しいフレームの処理が始まる箇所に時点イベントを挿入する、という使い方が考えられます。間隔イベントは、ロックを施している時間、ビデオの1フレームをデコードするのに要する時間など、ある操作の時間を計るために使います。

標識を記録するためには、System Calls instrumentをトレースドキュメントに組み込まなければなりません。このようなトレースドキュメントは「System Trace」テンプレートを使って作成します。標識イベントはSystem Callsのイベントリストに表示されます。

イベントリスト中の標識を検索できるようにするためには、あらかじめ、独自の標識定義をInstrumentsに認識させる必要があります。これは、System Calls instrumentの「Signposts」テーブルに、必要事項を記入して行います。各行に1つずつ、標識の定義として、イベントコードとイベント名を入力してください（図 2-2を参照）。

図 2-2 標識テーブル



標識イベントのコードは、0～16383の範囲の整数です。

標識定義をトレースドキュメントに追加すると、ソースコードに標識を埋め込めるようになります。その手順は、コードをどこで実行するかによって異なります。

- **ユーザアプリケーション**。Mac OS XやiOSのアプリケーションは、システムコールを呼び出して標識イベントを生成します。<sys/syscall.h>および<sys/kdebug.h>をインクルードし、次のように記述してください（標識の型を表す接尾部は、時点標識であればNONE、間隔標識の開始点はSTART、終了点はENDとなります）。

```
syscall(SYS_kdebug_trace, APPSDBG_CODE(DBG_MACH_CHUD, <your event code>) |
DBG_FUNC_<type>, arg1, arg2, arg3, arg4);
```

- **カーネル拡張**。Mac OS Xのカーネル拡張では、debugid引数にAPPSDBG_CODEマクロを指定して、kernel_debug関数を直接呼び出さなければなりません。<sys/kdebug.h>をインクルードし、次のように呼び出し文を記述してください（型を表す接尾部はsyscallの場合と同じ）。

```
KERNEL_DEBUG(APPSDBG_CODE(DBG_MACH_CHUD, <your event code>) | DBG_FUNC_<type>,
 arg1, arg2, arg3, arg4, 0);
```

いずれの場合も、呼び出し元ではユーザ定義のデータ値を、標識ごとに4つまで記録できます。この値は「System Calls」イベントリストに表示されます。

間隔イベントの場合、開始点と終了点を合体し、イベント間の経過時間に等しい長さの、1つの標識として表示するようになっています。開始点と終了点には同じコード値を割り当てなければなりません。標識を入れ子にすることも可能ですが、開始点と終了点のコード値を同じにすることだけ注意してください。

リスト 2-1に、間隔標識を利用して、ループの処理に要する時間を計る例を示します。

リスト 2-1　標識の使用例

```
#include <sys/syscall.h>
#include <sys/kdebug.h>

#define LOOP_TIMER 49
#define ITERATIONS 1000

/* 末尾4つの引数はユーザ定義データ */
syscall(SYS_kdebug_trace, APPSDBG_CODE(DBG_MACH_CHUD, LOOP_TIMER) |
DBG_FUNC_START, ITERATIONS, 0, 0, 0);

for (int ii = 0; ii < ITERATIONS; ii++) {
    do_some_stuff();
    do_more_stuff();
}
/* コード値が先に記述した値と同じであることに注意 */
syscall(SYS_kdebug_trace, APPSDBG_CODE(DBG_MACH_CHUD, LOOP_TIMER) | DBG_FUNC_END,
 ITERATIONS, 0, 0, 0);
```

Instruments 4.2の新機能

この章では、Instruments 4.2の主な新機能を簡単に紹介します。

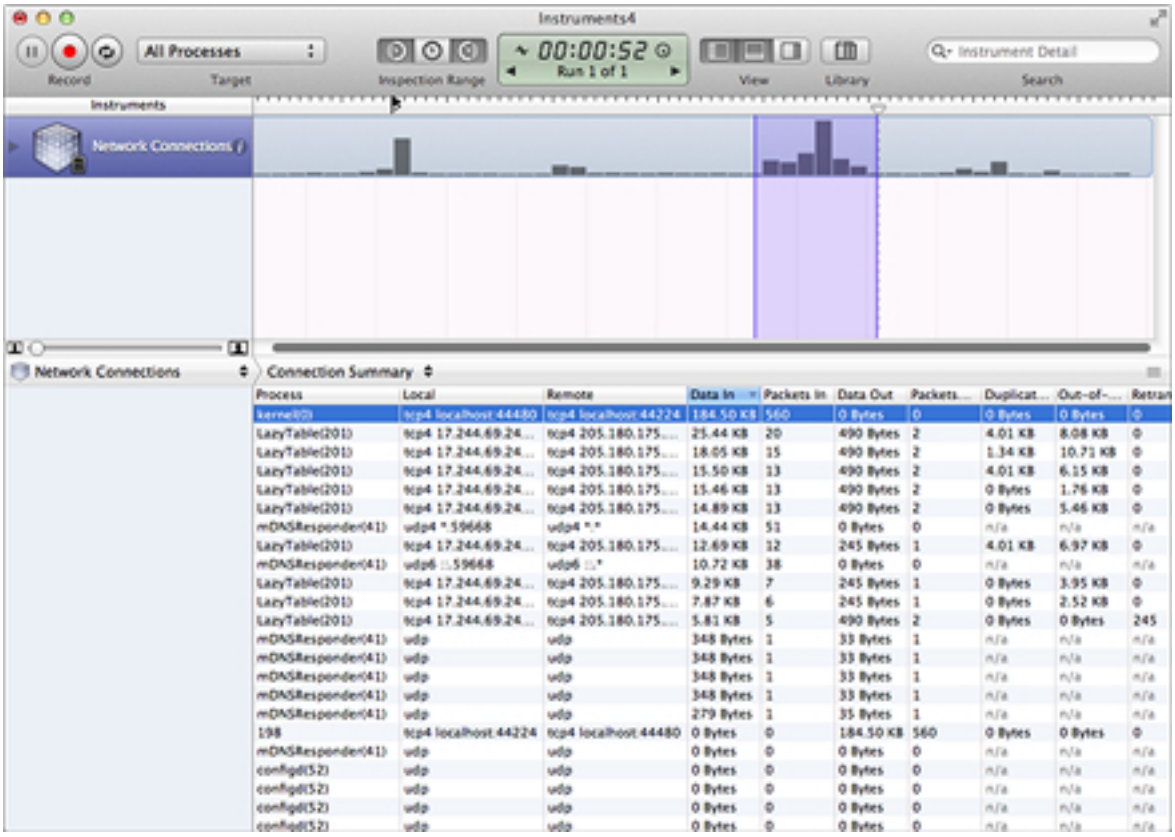
Network Connections

iOSアプリケーションがTCP/IPやUDP/IPでネットワーク接続している状況を、Networking Connections instrumentで調査できるようになりました。個々の接続ごと、アプリケーションごとに、実際に流れているデータ量を調べ、また、往復時間や再送信要求などの統計データも取得することができます。この情報を利用すれば、ネットワークトラフィックやエネルギー消費量の削減にも役立つでしょう。

記録を開始すると、instrumentは開いているポートすべてのスナップショットを取り、その時点までのネットワーク累積流量を詳細ビューに表示します。その後、流量を測定し、グラフ化してトラックビューに表示します。

詳細ビューの「Connection Summary」テーブルには、外向き/内向きのネットワーク接続が表示されます。図 3-1のトレースドキュメントには、指定された時間範囲のすべてのプロセスについて、開いている接続が表示されています。

図 3-1 Network Connections instrument



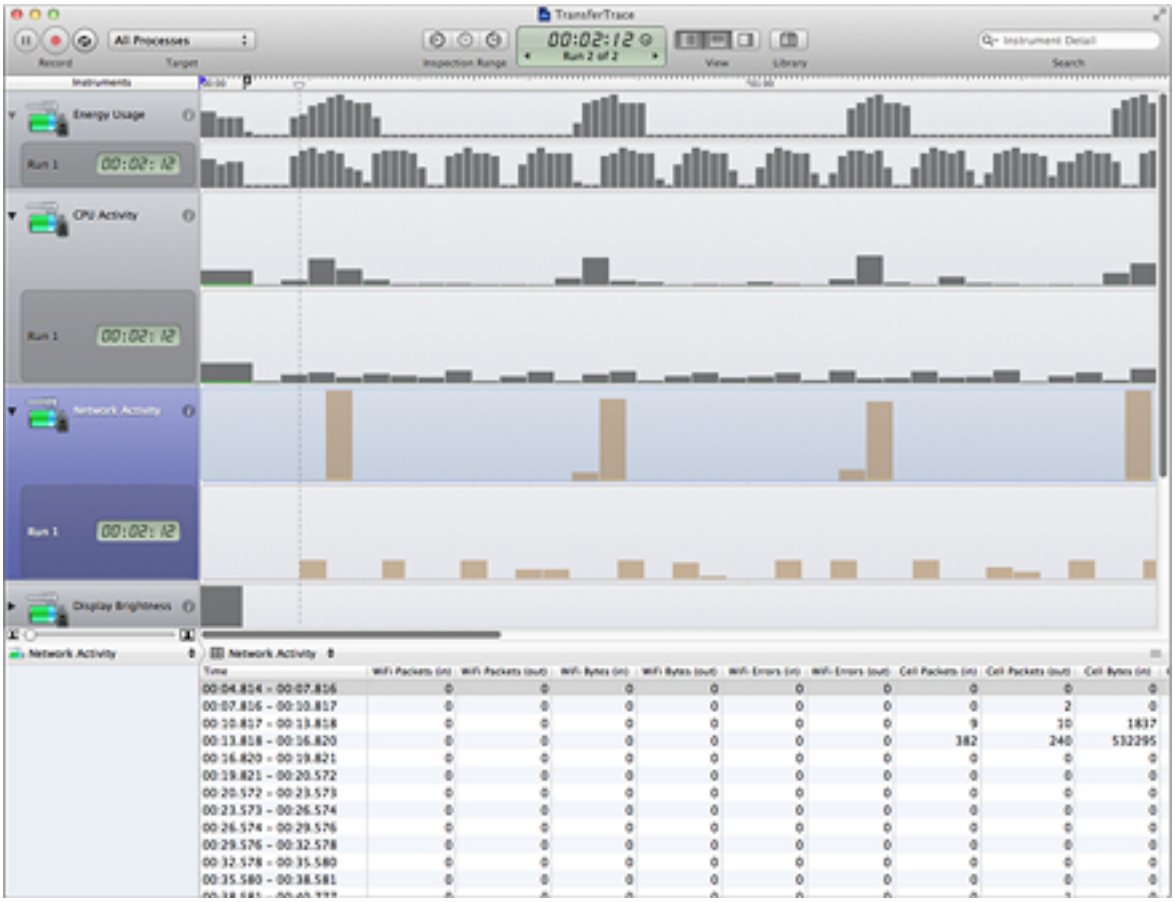
詳細ビューにはほかにも2つのテーブルを表示できます。「Process Summary」テーブルは、プロセスごとの累積データを集約して表示したものです。「Interface Summary」テーブルは、ネットワークインターフェイスごとに集約したデータが表示されます。詳細ビューにはほかに、「Trace Highlights」として、有用な一連のグラフをまとめて表示することも可能です

Network Activity

iOS用の「Energy Diagnostics」テンプレートに組み込まれているNetwork Activity instrumentを使うと、ネットワーク（携帯電話網およびWi-Fi接続）とエネルギー消費量を関連付けて調べることができます。デバイス全体のデータフローを、各ネットワークインターフェイスと、電池から直接取得した、エネルギー消費量のデータをもとに追跡できるのです。

iOSデバイスにおける、ネットワーク流量とエネルギー消費量の相関を把握するために有用です。たとえば図3-2には、「Energy Diagnostics」のトレースドキュメントが表示されています。1回目と2回目の実行で、エネルギー消費量が大幅に違っていることが分かります。

図 3-2 「Energy Diagnostics」テンプレートと、これに組み込まれたNetwork Activity instrument



1回目の実行では、小さい単位で頻繁にデータを送信しているため、無線通信路が長時間にわたって接続されたままであり、したがってエネルギー消費量も多くなっています。2回目の実行では、同じデータを大きな単位にまとめて伝送しており、その都度、無線通信路が切れています。

iOS用のSystem Trace

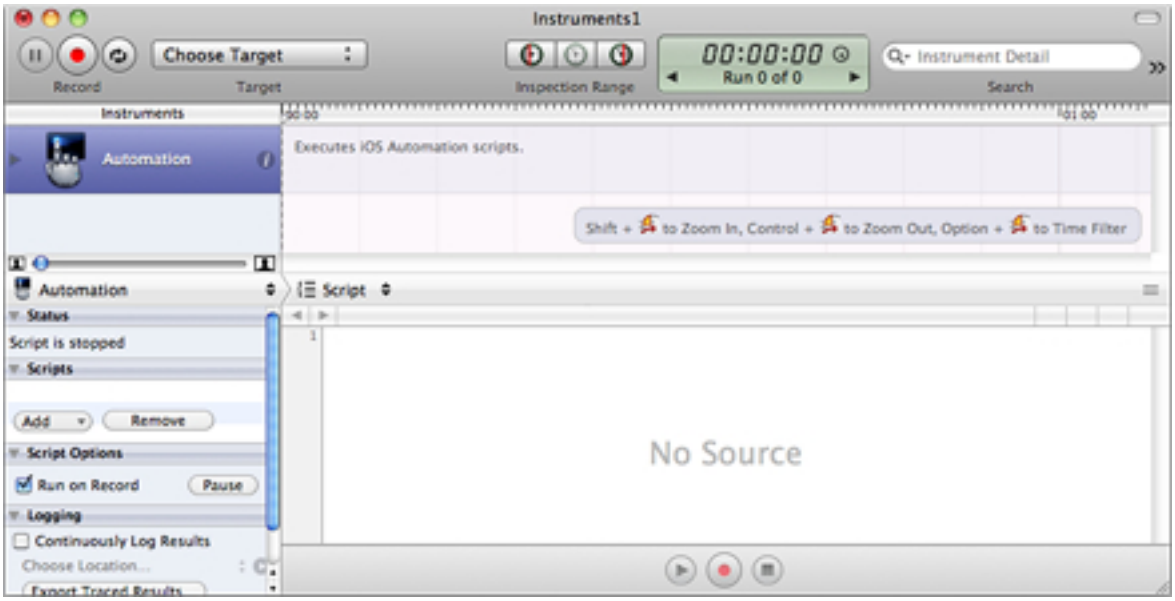
System TraceはInstruments用のテンプレートです。いくつかのinstrumentを組み合わせて、オペレーティングシステムが実行する、アプリケーション性能に影響を与えうる処理のプロファイルを、さまざまな観点から収集できます。Instruments 4.2では、iOSとMac OS Xの両方に対応しました。

Automation

Automation instrument（Instrumentsアプリケーションに付属、図3-3を参照）を使うと、テストスクリプトを記述することにより、iOSアプリケーションが接続先デバイス上で動作している状態で、ユーザインターフェイス要素の操作をシミュレートできます。テストスクリプトはJavaScriptで記述し、UI Automation APIを呼び出します。シミュレート結果はログ情報としてホストコンピュータに返されます。

UI Automationの使い方について詳しくは、『Instruments User Guide』の「Built-in Instruments」を参照してください。また、APIその他のレファレンスは、『*UI Automation Reference Collection*』を参照してください。

図 3-3 Automation instrument



今回の版では、次のような新機能が追加されています。

- 組み込みのスクリプトエディタによる編集機能
- 実際にユーザインターフェイス要素を操作してその手順を記録し、自動化スクリプトに取り込んで使う機能
- Xcodeプロジェクトからテストスクリプトを実行する機能
- APIのさまざまな改善。デバイスの位置移動をシミュレートする機能、ホスト上のAutomation instrumentからタスクを実行する機能など

Automation instrument上でテストスクリプトを編集する機能

Automation instrumentに組み込みのスクリプトエディタを使えば、トレースドキュメント上でテストスクリプトを編集できます。トレースドキュメント上で新しくスクリプトを作ることも、既存のスクリプトをインポートすることも可能です。エディタにはコード補完などの編集機能が備わっています。ただし、トレースドキュメント上で作成したスクリプトをほかの用途に使うためには、いったんエクスポートする必要があります。

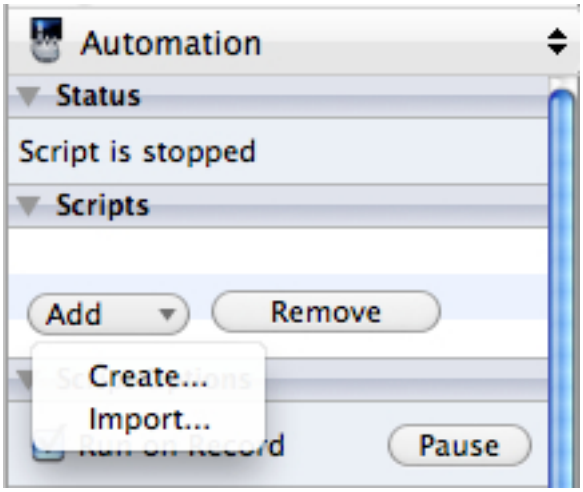
エディタ上でスクリプトを作成

トレースドキュメント上で編集するスクリプトを作成する手順を以下に示します。

1. 「Automationトレースドキュメントの作成」（33ページ）に示す手順で、トレースドキュメントを作成します。
2. 画面の左側、「Instruments」領域の「Automation」セクションで、必要ならば「Scripts」というラベルに添えられた三角印をクリックして、展開表示にしてください。
3. 「Add」ポップアップメニュー（図 3-4を参照）の「Create」を実行してください。

   右側ペインに新しいスクリプトが現れます。

図 3-4 トレースドキュメントにスクリプトを追加する様子



スクリプトをエディタ上にインポート

ディスク上にある既存のスクリプトをインポートして、トレースドキュメント上で編集できるようにする手順を以下に示します。

1. 「Automationトレースドキュメントの作成」（33ページ）に示す手順で、トレースドキュメントを作成します。
2. 画面の左側、「Instruments」領域の「Automation」セクションで、必要ならば「Scripts」というラベルに添えられた三角印をクリックして、展開表示にしてください。

3. 「Add」ポップアップメニュー（図 3-4を参照）の「Import」を実行してください。
4. ファイルシステム上にある、該当するスクリプトを選択し、「Open」をクリックしてください。

   右側ペインにスクリプトが開きます。

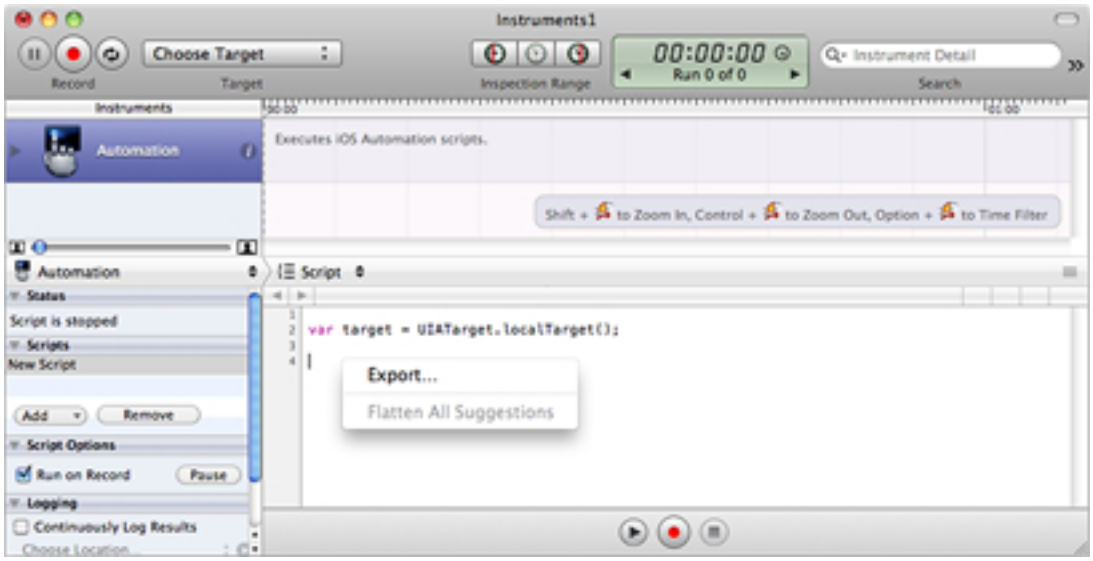
スクリプトをファイルにエクスポート

スクリプトエディタ上で変更を施すと、トレースドキュメントを保存する際、これも保存されます。エディタ上で作成したスクリプトの場合、トレースドキュメントそのものの一部として保存されます。ファイルとしてディスク上に保存し、ほかの目的でアクセスするためには、いったんエクスポートする必要があります。

その手順を以下に示します。

1. トレースドキュメント上のスクリプトを開いてください。
2. Controlキーを押しながら編集領域上をクリックすると、コンテキストメニューが現れます（図 3-5を参照）。

図 3-5 ディスク上のファイルとしてスクリプトをエクスポートしている様子



3. このメニューから、「Export」を実行してください。
4. ファイルシステム上の保存先を指定し、「Save」をクリックします。

ユーザインターフェイス要素に対する操作をAutomationのスクリプトとして取り込み

新たに加わった「取り込み」機能を使うと、スクリプトの作成が容易になります。ターゲットのiOSデバイス上、あるいはiOSシミュレータ上で実際にUI要素を操作し、その過程を記録できるのです。Automationのトレースドキュメントをあらかじめ作成しておき、デバイス上で実際に操作して、その動作（アクション）を取り込むことになります。取り込んだアクションはスクリプト中に埋め込まれ、後でこれを編集することも可能です。

Automationトレースドキュメントの作成

Automation instrumentとiOSターゲットを用い、次の手順でトレースドキュメントを作成します。

1. Instrumentsアプリケーションを起動します。
2. 「Automation」テンプレートを選択して、トレースドキュメントを作成します。

   注意： あるいは、Automation instrumentをinstrumentライブラリの「UI Automation」グループから既存のトレーステンプレートに追加し、トレースドキュメント上にドラッグしても構いません。

3. 必要ならば「View」＞「Detail」コマンドを実行して、詳細ビューを表示しておきます。
4. ウインドウの上部、「Target」メニューからテスト対象のiOSデバイスを選択し、iOSアプリケーションのリストから、テストしようとするアプリケーションを選んでください。

Automation instrumentによるアクションの取り込み

Automationのトレースドキュメントを開いた状態で、アクションを取り込んでスクリプトに埋め込む手順を以下に示します。

1. 必要ならば「Scripts」の三角印をクリックして、展開表示にしてください。
2. リストからスクリプトを選択します。

   注意： 該当するスクリプトがリストにない場合は、インポート（「Add」＞「Import」コマンド）するか、または新規に作成（「Add」＞「Create」コマンド）してください。

3. スクリプトエディタの編集領域上、アクションを埋め込みたい箇所をクリックし、その位置にカーソルを置いておきます。
4. テキストエディタの下部にある「Record」ボタンを押してください。

   ターゲットアプリケーションが起動し、スクリプト状態表示が「取り込み中」を表すものに変わります。

5. デバイス上で実際にUI要素を操作してください。

   重要： 適切にアクションを取り込むためには、ゆっくり、正確に操作することが大切です。

6. テキストエディタの下部、「Stop」ボタンを押すと、アクションの取り込みが終了します。

実行したアクションに対応するコードが、スクリプト中に埋め込まれます。コードによっては、その構成要素に対し、別の候補が含まれていることがあります。当該要素の右側にある矢印を押すと、その候補が表示されます。ある要素について、現在表示されている候補をそのまま採用し、コードを確定する場合は、その要素をダブルクリックしてください。

Xcodeプロジェクトからテストスクリプトを実行

iOS 5 SDKとXcode 4.2では、UI Automationのテストスクリプトを、アプリケーションを開発しているXcodeプロジェクトから起動できるようになりました。その大まかな手順を以下に示します。

1. アプリケーション用のAutomationテストスクリプトを記述します。詳しくは、『*UI Automation Reference Collection*』、および『*Instruments User Guide*』の「Built-in Instruments」（UI Automationに関する節）を参照してください。

2. Instruments上で、当該アプリケーションおよびテストスクリプトに合わせて、Automation instrumentのテンプレートを作成します。その手順については、以下の「Automation instrument用に、独自のテンプレートを作成」（34 ページ）を参照してください。

3. アプリケーションをXcode 4のプロジェクトとして開き、対象アプリケーションのプロファイルを設定することで、Instrumentsを起動し、テストスクリプトを実行します。その手順については、以下の「Automation instrumentのスクリプトをXcode上で実行」（35 ページ）を参照してください。

Automation instrument用に、独自のテンプレートを作成

Automation instrument用に、独自のテンプレートを作成する手順を以下に示します。

1. Instrumentsアプリケーションを起動します。

2. 「Automation」テンプレートを選択して、トレースドキュメントを作成します。

> **注意：** あるいは、Automation instrumentをinstrumentライブラリの「UI Automation」グループから既存のトレーステンプレートに追加し、トレースドキュメント上にドラッグしても構いません。

3. 必要ならば「View」＞「Detail」コマンドを実行して、詳細ビューを表示しておきます。

4. 必要ならば「Scripts」の三角印をクリックして展開表示し、スクリプトを選択します。

   該当するスクリプトがリストにない場合は、インポート（「Add」＞「Import」コマンド）するか、または新規に作成（「Add」＞「Create」コマンド）してください。

5. 「File」＞「Save as Template」コマンドを実行し、テンプレート名を指定し、次に示す、デフォルトのInstrumentsテンプレート保存場所に保存してください。

   ```
   ~/Library/Application Support/Instruments/Templates/
   ```

Automation instrumentのスクリプトをXcode上で実行

独自のAutomationテンプレートを作成すると、次の手順で、テストスクリプトをXcode上で実行できます。

1. Xcodeでプロジェクトを開きます。
2. 「Scheme」ポップアップメニュー（ワークスペースウインドウのツールバー）から「Edit Scheme」コマンドを実行すると、スクリプトの実行に用いるスキームの編集ダイアログが開きます。
3. スキーム編集ダイアログの左側の欄から、「Profile」を選択します。
4. 「Executable」ポップアップメニューから、対象アプリケーションを選択してください。
5. 「Instrument」ポップアップメニューから、先に作成したAutomation instrument用のテンプレートを選択します。
6. 「OK」を押すと、ここまでの設定が有効になり、スキーム編集ダイアログは消えます。
7. 「Product」＞「Profile」を実行してください。

   Instrumentsが起動し、テストスクリプトの実行が始まります。

Automation instrumentのスクリプトをコマンドラインから実行

テストスクリプトはコマンドラインから実行することも可能です。独自のAutomationテンプレートが作成済みであれば（「Automation instrument用に、独自のテンプレートを作成」（34 ページ）を参照）、次の簡単なコマンドで実行できます。

```
instruments -w deviceID -t templateFilePath targetAppName
```

deviceID
: 40字のデバイス識別子。XcodeのDevices organizerやiTunesで調べることができます。

注意： デバイス識別子の指定（上の例では「`-w deviceID`」）を省略すると、デバイスではなくシミュレータがターゲットになります。

templateFilePath
: 独自に作成したAutomationテンプレートの完全パス。デフォルト値は`~/Library/Application Support/Instruments/Templates/templateName`、ただし`templateName`はテンプレートを保存する際に指定した名前です。

targetAppName
: アプリケーションのローカル名。デバイス上で動かす場合は、パス名と拡張子「`.app`」を省いて指定します。シミュレータ上で動かす場合は完全パスを指定してください。

独自のテンプレートを用意しない場合は、デフォルトのトレーステンプレートを使っても構いません。その場合、環境変数`UIASCRIPT`および`UIARESULTSPATH`で、スクリプトと、結果を出力するディレクトリを指定します。

```
instruments -w deviceID -t defaultTemplateFilePath targetAppName
   -e UIASCRIPT scriptFilePath -e UIARESULTSPATH resultsFolderPath
```

defaultTemplateFilePath
: デフォルトテンプレートの完全パス。

```
/Developer/Platforms/iPhoneOS.platform/Developer/Library/Instruments/PlugIns/
AutomationInstrument.bundle/Contents/Resources/Automation.tracetemplate
```

scriptFilePath
: ファイルシステム上の、テストスクリプトの位置。

resultsFolderPath
: テストスクリプトの実行結果を出力する、ファイルシステム上のディレクトリの位置。

スクリーンショットの使用

スクリプトに、スクリーンショットを取る旨の記述ができます。UIATargetクラスの`captureScreenWithName`メソッド、`captureRectWithName`メソッドを使います。スクリーンショットの出力先フォルダは、テンプレートの左側にある「Logging」セクションを開き、「Continuously Log Results」オプションをオンにし、「Choose Location」ポップアップメニューで指定してください。スクリーンショットは、スクリプトに指定した名前で、このフォルダに格納されます。

注意： 結果を保存するフォルダのパス名は、コマンドラインから、`UIARESULTSPATH`環境変数で指定できます。この指定はトレーステンプレートでの設定よりも優先します。

タイムアウト期限の活用

新しい要素に初めてアクセスする際には、デフォルトのタイムアウト期限が適用されます。新しい要素を取得しようとして失敗した場合、UIAutomationは（若干の間隔をはさみながら）、成功するか、所定のタイムアウト期限に達するまで繰り返し試みます。タイムアウトになると、UIAutomationは`UIAElementNil`オブジェクトを返します。これは常に「無効な」UI要素を表すと考えればよいでしょう。

要素の取得に失敗する要因は、次のようにさまざまな要因が考えられます。

- アプリケーションが起動処理の途中である。
- 新しい画面の描画が完了していない。
- 要素（たとえばスクリプトでクリック操作を行うボタン）が描画されていても、その中身がまだ描画されていない、あるいは更新されていない。

タイムアウト期間の既定値は数秒ですが、必要ならばスクリプトで変更可能です。たとえば、要素が存在するかどうか確認するだけであれば、ないと分かった時点でそれ以上待つ必要はないので、タイムアウト期間を短くしても構わないでしょう。一方、スクリプトから要素にアクセスしなければならない場合は、タイムアウト期間を長くすることも考えられますが、ユーザインターフェイスの更新は遅くなります。タイムアウト期間を操作する、次のようなメソッドがUIATargetクラスに用意されています。

- `timeout`: タイムアウト期間の設定値を返します。
- `setTimeout`: タイムアウト期間の値を設定します。

- `pushTimeout`: 現在のタイムアウト期間設定値をスタックに積み、新しい値を設定します。
- `popTimeout`: スタックに積まれた、前回のタイムアウト期間設定値に戻します。

詳しくは『*UIATarget Class Reference*』を参照してください。

注意： タイムアウト値を`0`とすると、要素に1度アクセスした時点で失敗すれば、直ちに`UIAElementNil`オブジェクトを返すようになります。

スクリプトからのアクセスに用いるlabel属性、identifier属性

スクリプトからUI要素にアクセスする場合、label属性、identifier属性は重要な役割を果たします。この使い方を把握しておくことが大切です。

必須というわけではありませんが、label属性には、意味のある値を設定するよう推奨します。ラベル文字列の設定や表示は、Interface BuilderのIdentityインスペクタを起動し、「Accessibility」セクションの「Label」テキストフィールドで行います。意味が分かりやすく、しかし短いラベルを与えるようにしてください。ひとつには、Appleの音声インターフェイス「VoiceOver」など、障がいのある方に対するアクセス性向上のための技術が、UI要素の名前として使っているからです。UI Automationでは、このラベルを`label`メソッドで取得できます。また、`identifier`属性を設定していなければ、`name`メソッドもこのラベル文字列を返します。詳しくは『*UIAccessibilityElement Class Reference*』を参照してください。

identifier属性には、UI要素の内容をより詳しく説明する文字列を与えます。必須ではありませんが、スクリプトで次のような処理をする場合は必要です。

- コンテナビューに名前でアクセスし、同時にその子要素にもアクセスする必要がある場合。
- UILabelビューに名前でアクセスして、表示されているテキストを（その`value`属性として）取得する場合。

UI Automationでは、identifier属性が設定されている場合、`name`メソッドはこの値を返します。一方、未設定であれば、`label`属性の値を返します。

現状、identifier属性の設定は、`accessibilityIdentifier`プロパティを介してプログラム的に行う方法しかありません。詳しくは『*UIAccessibilityIdentification Protocol Reference*』を参照してください。

APIの変更

UI Automationに関係するAPIは、iOS 5.0に対応するため、次のようにいくつか改善が施されています。

- `UIATarget`は回転ジェスチャや位置変更に対応しました。
- `UIAHost`クラスを追加しました。テストスクリプトに、ホストコンピュータ上で動作しているAutomation instrumentのプロセス内で、ある処理を実行する旨を記述できるようになりました。

- `UIAElement`は、指定した要素を中心とする回転ジェスチャに対応し、ナビゲーションバー、タブバー、ツールバーの操作機能も組み込まれました。これは従来、`UIAWindow`クラス、`UIAPopover`クラスにしかなかった機能です。
- `UIAKeyboard`は、キーボードのキーをタップして、指定した文字列を入力する操作に対応しました。

APIの変更について詳しくは、『*UI Automation Reference Collection*』を参照してください。

書類の改訂履歴

この表は「*Instruments新機能ユーザガイド*」の改訂履歴です。

日付	メモ
2011-10-13	標識についての情報を追加しました。
2011-09-28	Instruments 4.2の新機能についての情報を追加しました。
2011-07-20	Instruments 4.1の新機能についての情報を追加しました。
2011-06-06	Instrumentsの新機能についてまとめた新規文書。

書類の改訂履歴