Oracle® Developer Studio 12.6 リリースの新機能

Part No: E83251-01
2017 年 7 月

ORACLE

Part No: E83251-01

ドキュメントのアクセシビリティについて

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWeb サイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info) か、聴覚に障害のあるお客様は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

目次

このドキュメントの使用法

- **概要** - この Oracle Developer Studio 12.6 リリースでのコンパイラとツールの新機能や変更された機能について説明します。
- **対象読者** – アプリケーション開発者、システム開発者、設計者、サポートエンジニア
- **必要な知識** - プログラミング経験、ソフトウェア開発テスト、ソフトウェア製品を構築およびコンパイルできる能力

製品ドキュメントライブラリ

この製品および関連製品のドキュメントとリソースは http://www.oracle.com/pls/topic/lookup?ctx=E83241-01 で入手可能です

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

♦ ♦ ♦ 第 1 章

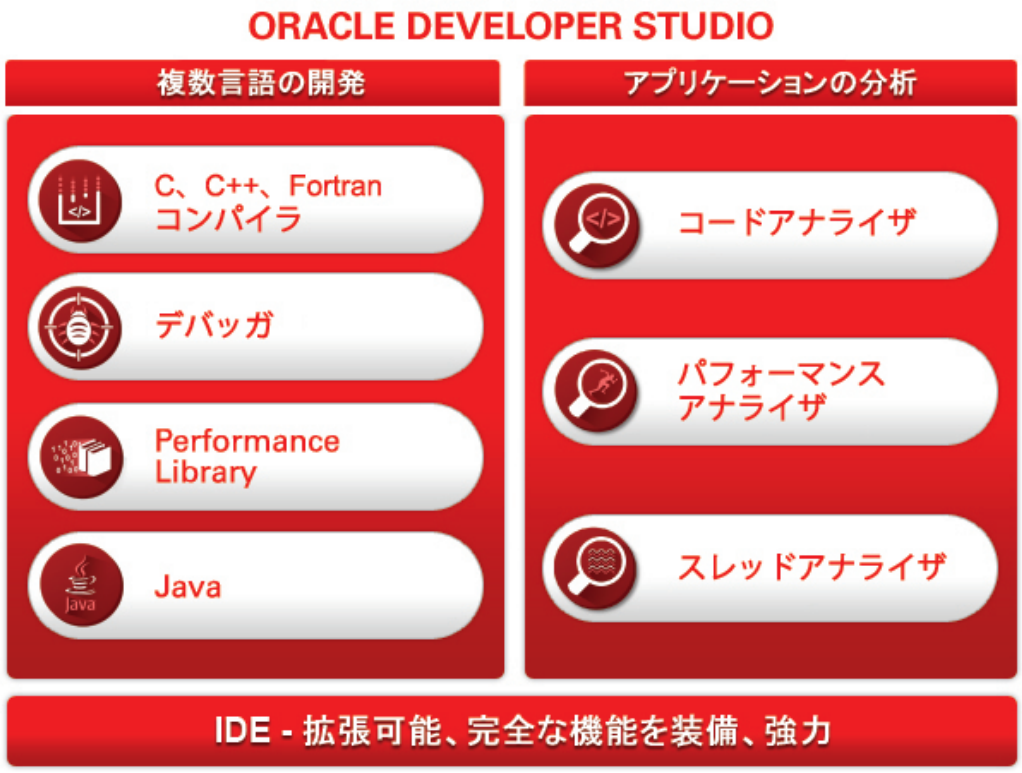
Oracle Developer Studio 12.6 リリースの紹介

この章では、このリリースの重要な更新の概要について説明します。

Oracle Developer Studio の概要

Oracle Developer Studio には、コンパイラスイート、分析スイート、および両方のスイートからコンパイラおよびツールとともに使用するように調整されたグラフィカル統合開発環境 (IDE) が組み込まれています。これらを組み合わせることで、Oracle Sun ハードウェア上で最高のパフォーマンスを発揮するアプリケーションの開発用に最適化された開発環境が提供されます。

図 1 IDE に統合されたコンパイラスイートおよび分析スイートを示す図



このリリースでの主な特長

Oracle Developer Studio 12.6 は、高度に最適化されたコンパイラ、高度な分析ツール、および多言語に対応した IDE を提供することにより、オンプレミスまたはクラウドで Oracle Solaris および Linux オペレーティングシステム用の高速で信頼性の高い、セキュアなアプリケーションを簡単に開発できるようにします。Oracle Developer Studio ツールは、完全なハードウェアおよびソフトウェアスタックに合わせて最適化されるため、開発チームはより優れたコードを迅速に記述できるようになります。

このリリースでの主な特長のサマリーは次のとおりです。

- **Oracle システムおよび Oracle Cloud での最速のコード生成** - 最新の Oracle SPARC M8、Fujitsu M12、および Intel x86 Skylake ベースのサーバーを活用するようにコンパイラ、コードジェネレータ、および実行時ライブラリが調整されます。これらの新しいシステムまたはプロセッサでは、そこで実行されるアプリケーションのパフォーマンスが最適になるようにコンパイラで考慮された新しい命令、新しいキャッシュサイズ、および新しいパイプラインが導入されています。詳細は、コンパイラのユーザーガイドを参照してください。
  - SPARC M8/T8 キャッシュおよび命令のスケジューリングの最適化
  - SPARC での境界整列していないデータアクセスをサポートする新しいプラグマ
  - パフォーマンスアナライザからアクセスできる SPARC M8 および Intel Skylake ハードウェアカウンタ情報
  - SPARC M8 チップ機能を活用するように調整されたパフォーマンスライブラリの主要ルーチン
  - Fujitsu SPARC M12 の最適化
  - Intel Skylake AVX512 のサポート
- **オープンソースの互換性と相互運用性による開発の簡略化** - コンパイラごとの正式な言語標準のサポートは、開発者が近年のほとんどのエンタープライズデータセンター内の各種システム間でコードを移動できるようにするためにきわめて重要です。また、オープンソースの移動により大量のアプリケーションが作成され、エンタープライズシステムに組み込まれています。こうしたオープンソースコードの大部分は GNU ツールセットを使用して記述されているため、この標準との互換性も重要です。Oracle Developer Studio 12.6 では次をサポートしています。
  - C++ 2011 および C++ 2014 の完全な言語標準のコンプライアンス。詳細は、第2章「C++ コンパイラ」を参照してください。
  - 40 を超える gcc 互換性機能。以前のリリースで提供された gcc 互換性機能を拡張したもの。詳細は、51 ページの「全コンパイラに共通の新機能および変更点」を参照してください
  - IDE およびパフォーマンスアナライザでの Java 9 のサポート
- **Oracle Developer Studio Cloud Service プラグインによるクラウドコンピューティングの高速化** -企業がオンプレミスのデータセンターから集中管理されたクラウドプロバイダに移行するにつれて、Cloud へのアプリケーションの配備がより一般的なものになっています。さらに開発者は、開発環境も集中管理されたクラウド提供のデータセンターに移行しています。このリリースでは、開発者がこうした移行を活用しやすくするための機能を提供します。
  - Oracle Developer Studio IDE では、IDE をローカルデスクトップ上で実行したり、Oracle Developer Studio Cloud Service にアクセスしたりできるようにする新しいプラグインを提供します。これにより、問題の追跡、ソースコード管理、構築サーバー、およびアプリケーションを Oracle Cloud 内に簡単に配備する方法が提供されます。詳細は、第7章「Oracle Developer Studio IDE」を参照してください。
  - Oracle Cloud Service では、ssh を使用してその Infrastructure as a Service Compute VM にアクセスします。Oracle Developer Studio のリモートパフォーマンスアナ

ライザは、ssh トンネリングをサポートしているため、開発者は Oracle Cloud Compute VM で実行されていたパフォーマンス実験を開始したり、実験結果を分析したりできます。パフォーマンスアナライザのリモートでの使用方法の詳細は、『Oracle Developer Studio 12.6: Performance Analyzer』 の 「Using Performance Analyzer Remotely」を参照してください

- マイクロサービスモデルを採用する開発者が増えるにつれ、こうしたマイクロサービスをホストするためにコンテナテクノロジも使用されるようになりました。Oracle Developer Studio IDE では、Docker コンテナを作成して、あとでそのようなコンテナをサポートしている Oracle Cloud やその他のシステムに配備できます。詳細は、42 ページの「Docker」を参照してください。

- **アプリケーションのセキュリティーの向上** - アプリケーションのセキュリティーは近年のアプリケーションにとって重要な要件です。Oracle Developer Studio では、次の機能を提供することで、開発者がセキュアなアプリケーションを簡単に作成できるようにしています。
  - IDE でのセキュアなコーディングの支援。Oracle Solaris のセキュアなコーディングの推薦事項に準拠しないコードを強調表示し、よりセキュアと思われる代替案も提案します。詳細は、29 ページの「IDE の新機能および変更された機能」を参照してください。
  - `discover` ツールでは、SPARC Silicon Secured Memory (SSM) を活用して、誤ったアプリケーションメモリーへのアクセスやメモリーの破損 (一般的なセキュリティー違反の弱点) に関するリアルタイム情報を提供できます。詳細は、『Oracle Developer Studio 12.6: Discover and Uncover User's Guide』 の「Hardware-Assisted Checking Using Silicon Secured Memory (SSM)」を参照してください。
  - コンパイラでは、コンパイルのたびに自動コード検査を実行できます。この機能の副作用は、Oracle Developer Studio 12.6 コンパイラを使用したコードの構築時に追加の警告が表示される可能性があることです。こうした検査を無効にするコンパイラオプションがあります。

♦ ♦ ♦ 第 2 章

2 C++ コンパイラ

この章では、このリリースの Oracle Developer Studio の C++ コンパイラの新機能と更新された機能について説明します。

C++ コンパイラについて

このセクションでは、Oracle Developer Studio 12.6 の C++ 5.15 コンパイラリリースに導入されている新機能や変更点のサマリーを記載します。

C++ コンパイラ (CC) は、指定されたコマンド行オプションに従って、特定のオペレーティングシステム、プロセッサ、アーキテクチャー、メモリーモデル (32 ビットおよび 64 ビット)、浮動小数点演算などを対象とするコードを生成します。コンパイラは、シリアルソースコードに対して自動並列化を実行し、マルチコアシステムで優れたパフォーマンスを発揮するバイナリを生成するほか、ほかの Oracle Developer Studio ツールによる強化されたデバッグまたは分析用にバイナリを作成することもできます。このコンパイラは GNU C および C++ 互換性機能もサポートしています。

C++ コンパイラの変更点

このセクションには、51 ページの「全コンパイラに共通の新機能および変更点」に記載されている変更点も含まれます。

このリリースでの C++ コンパイラに固有の新機能は次のとおりです。

- **新しい** `section` **属性** – C++ コンパイラでは、特定のセクションに存在する変数または関数を指定する新しい `section` 属性を提供するようになりました。

- `-xO3` の**インライン化動作の変更** - 本体が呼び出し側のオーバーヘッドよりも小さい関数の自動インライン化をサポートします。また、`-xO3` および `-xO4` は、`-xspace` オプションとともに使用すると、最小のコードサイズになります。
- `-xlibmopt` **オプションの変更** – `-xlibmopt` オプションは、`-%none`、`-archive`、および `-shared` サブオプションで拡張されました。
- `libCrunG3` **が Linux では静的にリンクされる** – `libCrunG3` は、`-compat=g`、`-std=c++03`、`-std=c++11`、または `-std=c++14` のいずれかのオプションでコンパイルされる C++ プログラムで必要な実行時サポートライブラリです。

  Oracle Solaris では、`libCrunG3` は必要に応じて、デフォルトで動的にリンクされます。これは Oracle Solaris オペレーティングシステムの一部であるため、常にユーザープログラムによって検出できます。Linux では、このライブラリに動的にリンクする場合、使用しているプログラムで `libCrunG3.so.1` を指定することが必要になります。この問題を回避するために、Linux では `libCrunG3` は必要に応じて、デフォルトで静的にリンクされます。
- **関数の多重定義の解決におけるエラー通知の改善** – Oracle Developer Studio および Oracle Solaris Studio の以前のコンパイラでは、多重定義された関数のあいまい呼び出しや解決不能呼び出しを通知する際に詳細メッセージが含まれていませんでした。Oracle Developer Studio 12.6 では、エラー通知が改善されました。次の例は、改善されたエラー通知機能を示しています。

```
Example 1:

% cat ex1.cc
void f(int n, int* ptr);
void f(int, int, int);
void f(int);
void foo()
{
    f(1, 2);
}
```

以前のコンパイラでは、次のようなメッセージが通知されました。

```
% CC -c ex1.cc
"ex1.cc", line 7: Error: Could not find a match for f(int, int) needed in
foo().
1 Error(s) detected.
```

Oracle Developer Studio 12.6 C++ コンパイラでは、一致するものが見つからなかった理由について詳細な情報を提供します。

```
% CC -c ex1.cc
"ex1.cc", line 7: Error: Could not find a match for f(int, int) needed in
```

```
foo().
"ex1.cc", line 1: Note: Candidate 'f(int, int*)' is not viable: argument
'ptr' can't be converted from 'int' to 'int*'.
"ex1.cc", line 3: Note: Candidate 'f(int)' is not viable: too many arguments.
"ex1.cc", line 2: Note: Candidate 'f(int, int, int)' is not viable: too few
arguments.


Example 2:

% cat ex2.cc
struct A;
struct B {
    B(const A&);
};
struct D1 : B {};
struct D2 : B {};
struct A {
    A();
    operator  D1() const;
    operator  D2&() volatile ;
};
void foo(A& ra)
{
    A va;
    B vb  = ra;
}
```

以前のコンパイラでは、次のようなメッセージが通知されました。

```
% CC -c ex2.cc
"ex2.cc", line 19: Error: Overloading ambiguity between "A::operator D1()
const" and "B::B(const A&)".
"ex2.cc", line 19: Error: Cannot use A to initialize B.
2 Error(s) detected.
```

Oracle Developer Studio 12.6 C++ コンパイラでは、あいまいさについて詳細な情報を提供します。

```
% CC -c ex2.cc
"ex2.cc", line 19: Error: Type conversion 'A' -> 'B' is ambiguous.
"ex2.cc", line 12: Note: Viable candidate 'A::operator D1() const'.
"ex2.cc", line 4: Note: Viable candidate 'B::B(const A&)'.
"ex2.cc", line 13: Note: Viable candidate 'A::operator D2&() volatile'.
"ex2.cc", line 19: Error: Cannot use A to initialize B.

You can disable generation of the Notes by using the CC option
```

```
-features=no%note
```

詳細は、『Oracle Developer Studio 12.6: C++ User's Guide』および cc(1) のマニュアルページを参照してください。

♦ ♦ ♦ 第 3 章

3 C コンパイラ

この章では、Oracle Developer Studio 12.6 リリースでの C コンパイラの変更点について説明します。

C コンパイラについて

このセクションでは、Oracle Developer Studio 12.6 C 5.14 コンパイラリリースに導入されている新機能と変更された機能のサマリーを記載します。

C コンパイラ (cc) は C コンパイルシステムへのインタフェースです。コンパイルプロセスには、プリプロセッサ、コンパイラ、コードジェネレータ、オプティマイザ、アセンブラ、およびリンカーが組み込まれています。C コンパイラは、指定されたオプションを処理してから、さまざまなコンポーネントを適切な引数で実行します。詳細は、cc(1) のマニュアルページを参照してください。

C コンパイラの変更点

C コンパイラの変更点には、51 ページの「全コンパイラに共通の新機能および変更点」に記載されている変更点と、次の変更点が含まれます。

- 新しいコンパイラオプションは次のとおりです。
  - `-fcommon` および `-fno-common`
  - `-features=[no%]gcc_enums`
  - `-fexceptions`
  - `-fsemantic-interposition`
  - `-fno-semantic-interposition`

    - `-fshort-enums`
    - `-fvisibility`
    - `-shared`
    - `-std=gnu11 -std=gnu99 -std=gnu90 -std=gnu89 -std=c90` および同等の別名
- 新しい `lint` オプションは次のとおりです。
    - `-features=[no%]gcc_enums`
    - `-fshort-enums`
- gcc との互換性の改善に伴う機能は次のとおりです。
    - 型におけるゼロサイズの構造体
    - `__builtin_offsetof` 組み込み関数
    - `__builtin_expect` 組み込み関数
    - 行継続文字 (バックスラッシュ \) のあとの空白を受け入れます
    - BOM 文字を正しく処理します
    - `-fshort-enums` オプションの下に char および short 列挙を許可します
    - `-features=[no%]gcc_enums` 付きの符号なし int 列挙を許可します
    - `-features=[no%]gcc_enums` 付きの long および符号なし long 列挙を許可します
    - 列挙での packed 属性を許可します
- サポートされている新しい属性は次のとおりです。
    - `__packed__`
    - `section(…)`
- `__cpuid` 組み込み関数が追加されました
- C11 文字列リテラル `u8""`、`u""`、および `U""` のサポートが追加されました

詳細は、cc(1) のマニュアルページおよび『Oracle Developer Studio 12.6: C User's Guide』を参照してください。

♦ ♦ ♦ 第 4 章

4 コード分析ツール

コード分析ツールスイートは、メモリーリークやアクセス違反といった一般的なコーディングエラーを検出することでアプリケーションの信頼性および安定性を確保します。これにより、開発者は優れたコードを少ないエラーで迅速に記述できるようになります。

この章では、Oracle Developer Studio のこのリリースにおけるコード分析ツールの新機能や変更点について説明します。次のセクションで構成されています。

コード分析ツールについて

コード分析ツールを使用すると、静的分析、動的分析、およびコードカバレージ分析を使用して、メモリーリークやメモリーアクセス違反など一般的なコーディングエラーの多くを検出することにより、アプリケーションの信頼性を高めることができます。Previse ではコンパイル時に静的分析を実行し、`discover` ではアプリケーション実行時に動的分析を実行することにより、コード品質の問題を特定します。`uncover` ツールは、コードカバレージデータを分析して、テストスイートの対象にならない関数に関する情報と、それらの関数を対象とすることで得られる利点について通知します。

コードアナライザグラフィックツールまたは `codean` コマンド行ユーティリティーを使用すると、3 種類の分析を表示できます。この分析でアプリケーションの脆弱性について包括的な見地を得ることにより、その妥当性や信頼性を向上させることができます。

discover の新機能

このリリースでは、次の機能が discover メモリー分析ツールに追加されました。

- **以前に計測機構が組み込まれたバイナリでの discover 使用の機能強化** - discover ユーティリティーは、計測機構が組み込まれたバイナリバージョンで上書きする前にバイナリを自動的に保存します。元のバイナリを保存しなくても、このユーティリティーをすぐに実行できるようになりました。さらに、バイナリを明示的に保存しないで uncover と同じバイナリに対して discover を実行することもできます。Linux と Oracle Solaris のどちらのバイナリでも、注釈がデフォルトで生成されるようになりました。

詳細は、『Oracle Developer Studio 12.6: Discover and Uncover User's Guide』および discover(1) のマニュアルページを参照してください。

uncover の新機能

このリリースでは、次の機能が uncover コードカバレージツールに追加されました。

- **以前に計測機構が組み込まれたバイナリでの uncover 使用の機能強化** - uncover ユーティリティーは、計測機構が組み込まれたバイナリバージョンで上書きする前にバイナリを自動的に保存します。計測機構の組み込まれたバイナリに対してこのユーティリティーをすぐに実行できるようになりました。さらに、バイナリを保存しないで discover と同じバイナリに対して uncover を実行することもできます。Linux と Oracle Solaris のどちらのバイナリでも、注釈がデフォルトで生成されるようになりました。

詳細は、uncover(1) のマニュアルページおよび『Oracle Developer Studio 12.6: Discover and Uncover User's Guide』を参照してください。

♦ ♦ ♦ 第 5 章

5 パフォーマンス解析ツール

パフォーマンス解析ツールを組み合わせて使用することで、ユーザーはアプリケーションの動作を解析し、パフォーマンスに影響を及ぼす問題点を見つけることができます。

この章では、Oracle Developer Studio のこのリリースにおけるパフォーマンス解析ツールの新機能や変更点について説明します。

パフォーマンスアナライザについて

パフォーマンスアナライザは、アプリケーションの動作をユーザーが把握できるようにすることで、コード内で問題となっている領域を見つけることができます。これは、システムリソースをもっとも多く使用している関数、コードセグメント、およびソース行を識別します。パフォーマンスアナライザは、単一スレッド、マルチスレッド、およびマルチプロセスのアプリケーションのプロファイルを作成でき、プロファイリングデータを提供することで、アプリケーションのパフォーマンスを改善できる場所を特定するのに役立てることができます。

パフォーマンスアナライザは、次のような一連のコマンドおよびツールで構成されています。

collect ユーティリティー	ユーザーレベルのプログラム上のプロファイリングデータを収集します。
`er_kernel` ユーティリティー	Oracle Solaris カーネル上のプロファイリングデータを収集します。
`er_print` ユーティリティー	プロファイリング情報をテキスト形式で表示します。
パフォーマンスアナライザ GUI	プロファイリング情報をグラフィカルに表示します。

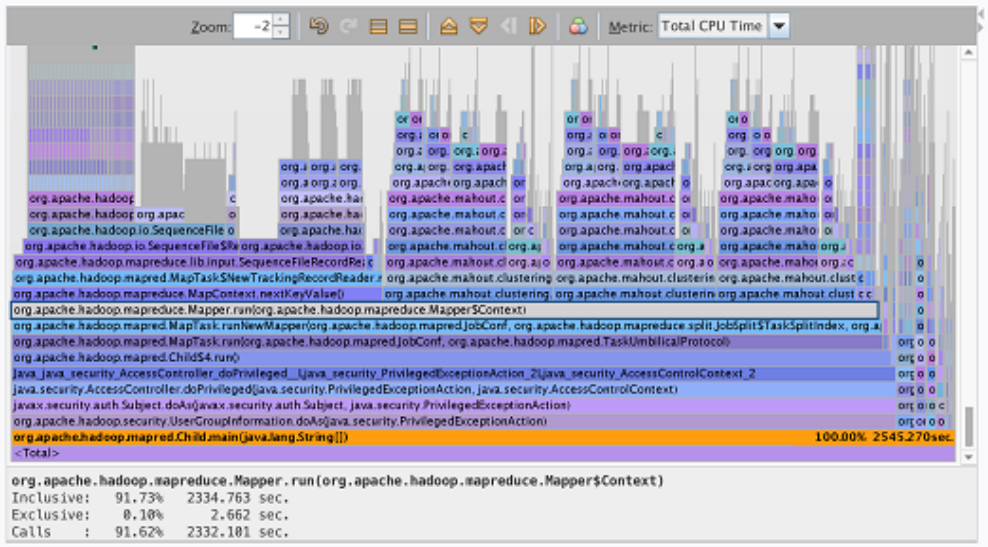
スレッドアナライザは、マルチスレッドの問題に的を絞ることが可能な関連ツールです。

詳細は、『Oracle Developer Studio 12.6: Performance Analyzer』および『Oracle Developer Studio 12.6: Performance Analyzer Tutorials』を参照してください。

パフォーマンスアナライザの新機能

このセクションでは、このリリースでのパフォーマンスアナライザおよび関連ツールの新機能の概要を示します。詳細は、パフォーマンスアナライザのヘルプを参照してください。

- **新しい「ビジュアル呼び出しツリー」ビュー** - 「ビジュアル呼び出しツリー」という新しいビューには、実行のホットパスがフレームグラフと呼ばれるグラフィカル形式で表示されます。



詳細は、『Oracle Developer Studio 12.6: Performance Analyzer』の「Flame Graph View」のマニュアルを参照してください。

- **Scala アプリケーションの新規サポート** - Scala アプリケーションの分析の初期サポートが追加されました。
- **GNU インライン化の分析の機能強化** - GNU コンパイラでコンパイルされたバイナリの場合、「PC」および「逆アセンブリ」ビューで、インライン化された命令と、もっとも深いインライン化されたコードに関連するソースが特定されるようになります。

- **「メモリー領域」の機能強化** - ユーザーレベルのビューの名前と説明を示すように「メモリー領域」ビューの名前が更新されました。
- **<no Java callstack recorded> の分析の向上** - JVM が特定のサンプルの現在の Java 呼び出しスタックを取得できない場合、アナライザはそのサンプルを *<no Java callstack recorded>* として報告します。このリリースでは、「PC」ビューで、欠落している Java 呼び出しスタックに関連するエラーコードが説明されます。また、フィルタを使用して *<no Java callstack recorded>* を分離したり、マシンモードに切り替えて欠落している Java 呼び出しスタックおよび関数を分析したりできます。
- **大規模な jar ファイルのサポート** - 大規模な jar ファイルを使用する Java および Scala アプリケーションがサポートされるようになりました。
- **インテルコンパイラ** `-ipo` **のサポート** - パフォーマンスアナライザでは、インテルの `-ipo` フラグでコンパイルされたコードをサポートするようになりました。
- **ビュー履歴のナビゲーション機能** - 新しいコントロールでは、以前開いていたビューへのナビゲーションをサポートします。

コマンド行ツールの変更点

このセクションでは、コマンド行のさまざまなパフォーマンス解析ツールに対して行われた変更について説明します。詳細は、各コマンド行ツールの対応するマニュアルページを参照してください。

データ収集ツールの変更点

データ収集ツールには、`collect` コマンド、`dbx collector` コマンド、および `er_kernel` コマンドがあります。これらの各ツールは、プログラムをプロファイルしてデータを収集し、パフォーマンスアナライザまたは `er_print` によって読み取ることが可能な実験を作成するために使用されます。

`collect` ユーティリティーの変更点

`collect` ユーティリティーは、アプリケーションの実行中にアプリケーションをプロファイルしてデータを収集し、パフォーマンスアナライザまたは `er_print` によって読み取り可能な実験を作成するために使用するツールです。

すべてのデータ収集ツールに共通する変更点に加え、`collect` ユーティリティーはこのリリースでは次のように変更されています。

プロファイリング中に呼び出しスタックの取得を無効にできるようになりました。x86 で Java をプロファイリングするときに、呼び出しスタックの取得

を無効にすると、スタックの巻き戻しに関する致命的エラーのリスクを減らすことができます。詳細は、collect(1) のマニュアルページに記載されている SP_COLLECTOR_NATIVE_MAX_STACKDEPTH の情報を参照してください。

er_kernel ユーティリティーの変更点

er_kernel コマンドは Oracle Solaris カーネルをプロファイルし、パフォーマンスアナライザまたは er_print で調査できる実験を生成します。

er_kernel ユーティリティーは次のように変更されています。

- OVM (xen) の下で動作している Oracle Solaris VM で、カーネルプロファイリングがサポートされるようになりました。

詳細は、er_kernel(1) のマニュアルページを参照してください。

er_print ユーティリティーの変更点

er_print ユーティリティーは、パフォーマンスアナライザで提供されるデータビューのプレーンテキストバージョンを生成します。その出力は、標準出力ウィンドウに表示されます。

er_print ユーティリティーはこのリリースでは次のように変更されています。

- コマンド行の編集および履歴が Oracle Solaris で使用できるようになりました。
- 大規模な jar ファイルを使用する Java および Scala アプリケーションがサポートされるようになりました。
- パフォーマンスアナライザでは、インテルの -ipo フラグでコンパイルされたコードをサポートするようになりました。
- Java ガベージコレクタイベントをダンプ出力するユーティリティー、dumpgc が使用できるようになりました。
- 欠落している Java 呼び出しスタックに関連するさまざまなエラーコードの説明が -pcs オプションで表示されるようになりました。
- GNU コンパイラでコンパイルされたバイナリの場合、「PC」および「逆アセンブリ」ビューで、インライン化された命令と、もっとも深いインライン化されたコードに関連するソースが特定されるようになります。
- Scala アプリケーションの分析がサポートされるようになりました。

詳細は、er_print(1) のマニュアルページを参照してください。

その他のコマンドと API の変更点

`er_archive` コマンドには、次の更新があります。

- 新しいコマンドオプション `-d` *path* - このオプションでは、絶対パス *path* で指定された共通ディレクトリにアーカイブを作成します。
- 新しいコマンドオプション `-r` *path* - このオプションでは、相対パス *path* で指定された共通ディレクトリにアーカイブを作成します。

コマンド `bw`、`ripc`、`spot`、`spot_cmds`、`spot_cmds_timing`、`spot_diff`、`traps` はサポートされなくなりました。

♦ ♦ ♦ 第 6 章

6 デバッグツール

Oracle Developer Studio には、コマンド行の dbx デバッガ、および dbx を使用するための dbxtool グラフィカルツールが用意されています。また、デバッガは IDE に統合されています。IDE を使用したデバッグの詳細は、第7章「Oracle Developer Studio IDE」を参照してください。

この章の内容は、次のとおりです。

dbx デバッガについて

dbx デバッガは、対話型でソースレベルの事後およびリアルタイムのデバッグツールです。コマンド行、dbxtool グラフィカルインタフェース、および Oracle Developer Studio IDE で使用できます。dbx デバッガは、スクリプティング可能で、マルチスレッドに対応しています。

dbx の新機能と変更された機能

dbx で追加または変更された機能は次のとおりです。

- discover エンジンが実行時検査 (RTC) でサポートされるようになりました。
- ベクトルレジスタをさまざまな形式で表示できます。詳細は、help examine および help registers を参照してください。
- プリティプリンティングフィルタの作成を主な目的とする組み込み Python インタプリタと新たなプラグイン API が Linux でもサポートされるようになりました。

詳細は、『Oracle Developer Studio 12.6: Debugging a Program with dbx』、dbx(1) のマニュアルページ、および dbx のヘルプファイルを参照してください。dbx のヘルプファイルにアクセスするには、次のように入力します。

```
% dbx
(dbx) help
```

詳細は、dbx で help changes コマンドを発行して、dbx のヘルプファイルにアクセスしてください。

♦ ♦ ♦ 第 7 章

7 Oracle Developer Studio IDE

Oracle Developer Studio IDE は、グラフィカルプログラミング環境を好むユーザー向けに Oracle Developer Studio の多くのコンポーネントを統合します。

この章は次の各節から構成されています。

Oracle Developer Studio IDE について

Oracle Developer Studio では、NetBeans プラットフォーム上に構築されたグラフィカル統合開発環境 (IDE) を提供します。この IDE は、Oracle Developer Studio C、C++、および Fortran コンパイラ、`dmake` 分散 make コマンド、および `dbx` デバッガを使用するように構成されます。また、IDE は分析スイートの一部のアナライザツールを統合するため、IDE から離れることなくコードを分析できます。

IDE を起動するコマンドは `devstudio` です。詳細は、`devstudio(1)` のマニュアルページを参照してください。

IDE の完全なドキュメントは、IDE のヘルプを参照してください。IDE の基本機能を使用する手順については、『Oracle Developer Studio 12.6: IDE Quick Start Tutorial』を参照してください。

IDE の新機能および変更された機能

Oracle Developer Studio IDE で追加または変更された機能は次のとおりです。

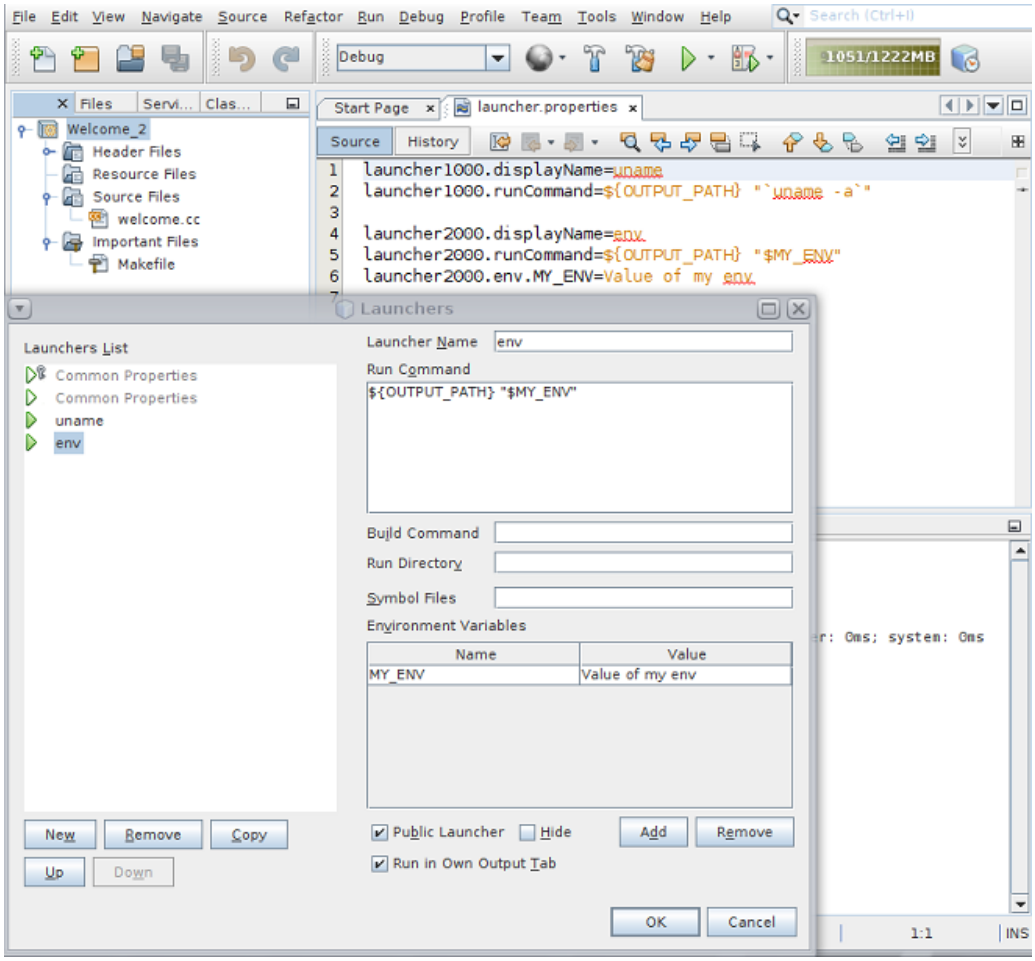
- **起動ツールの実行/デバッグ構成の管理** - 詳細は、30 ページの「実行/デバッグ起動ツールの UI」を参照してください。

- **PATH 変数の変更** - 詳細は、31 ページの「ツールチェーンパスの前または後への付加」を参照してください。
- **テキストフィールドの環境変数** - 詳細は、33 ページの「環境変数の追加」を参照してください。
- **ツールコレクションラッパーに基づいたビルドアナライザ** - 詳細は、34 ページの「ビルドアナライザ」を参照してください
- **C/C++ キャッシュの消去** - 詳細は、36 ページの「C/C++ キャッシュの消去」を参照してください
- **検索の使用における読み取り/書き込みアクセスの区別** - 詳細は、36 ページの「読み取り/書き込みアクセスの区別」を参照してください
- **ターミナルの拡張機能** - 詳細は、37 ページの「ターミナル機能の拡張機能」を参照してください。
- **新しい固定可能ウォッチ** - 詳細は、39 ページの「固定可能なウォッチ」を参照してください

実行/デバッグ起動ツールの UI

新しい UI を使用して実行/デバッグコマンドの構成を管理できます。構成ウィンドウを開くには、プロジェクトのコンテキストメニューから「起動ツールの管理」をクリックします。

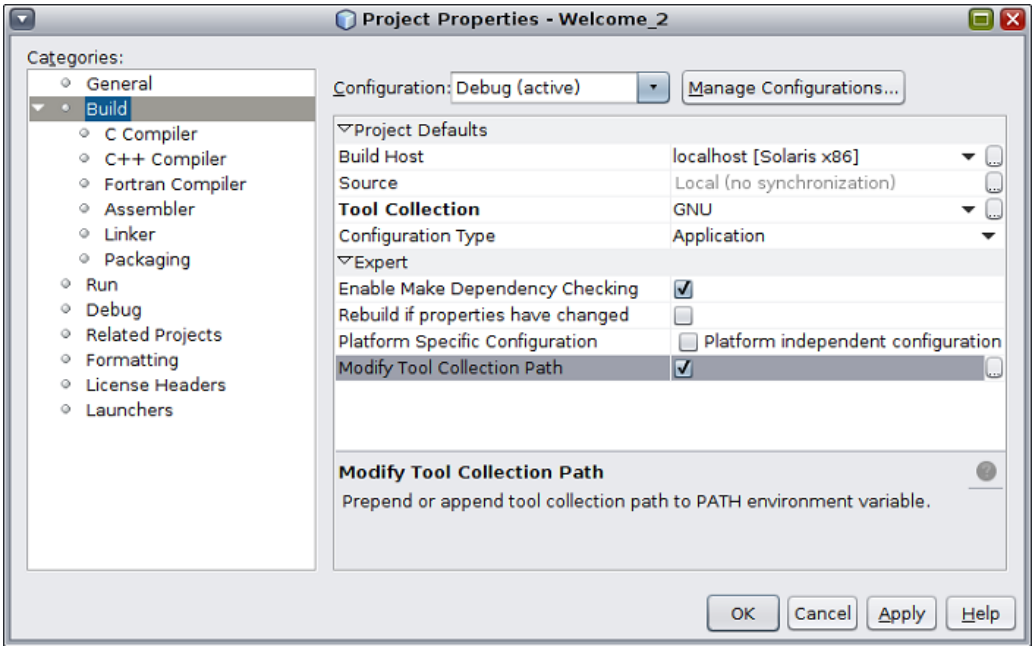
図 2 実行/デバッグ起動ツールの UI



ツールチェーンパスの前または後への付加

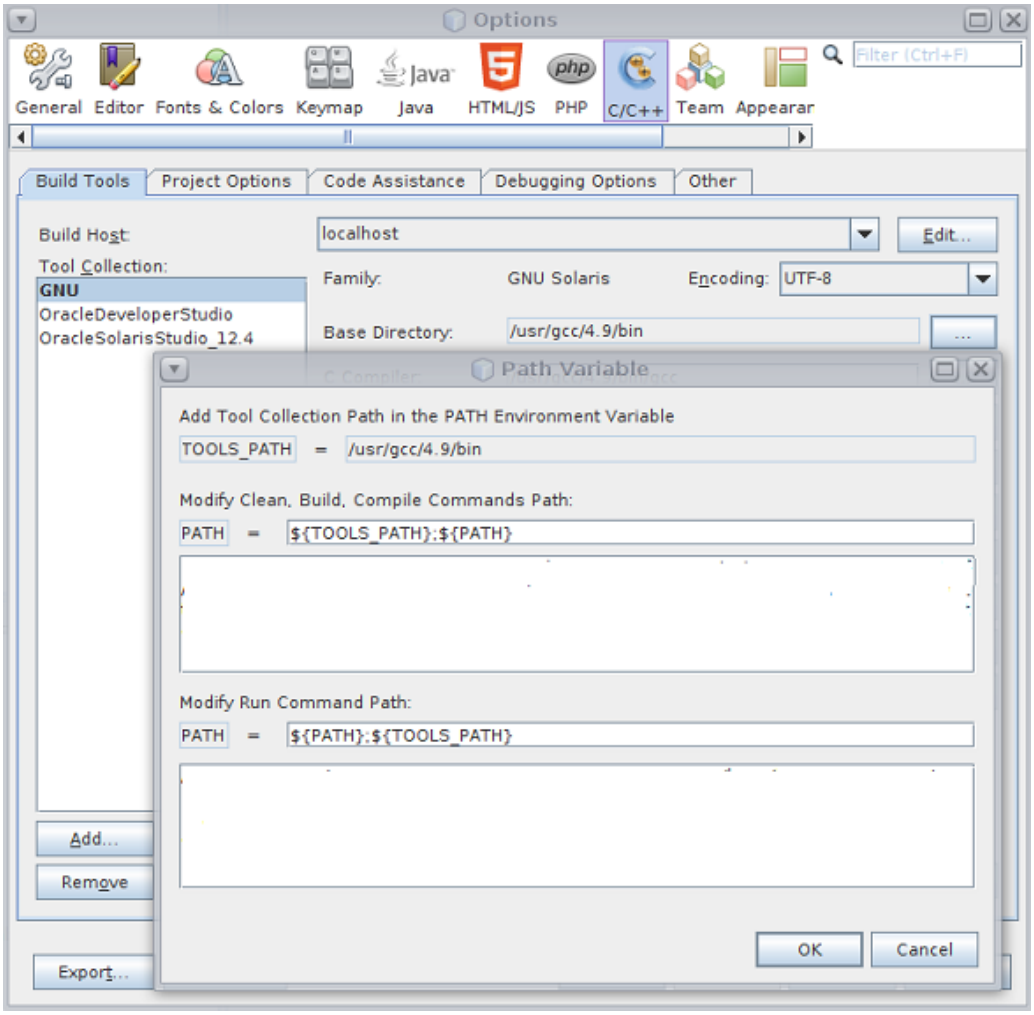
「プロジェクトのプロパティー」の新しいオプションを使用すると、ツールチェーンディレクトリを PATH 変数に付加するかどうかを指定できます。

図 3 「ツールコレクションパスの変更」オプション



デフォルトでは、$TOOLS_PATH が前に付加されます。ただし、「オプション」ウィンドウで変更できます。

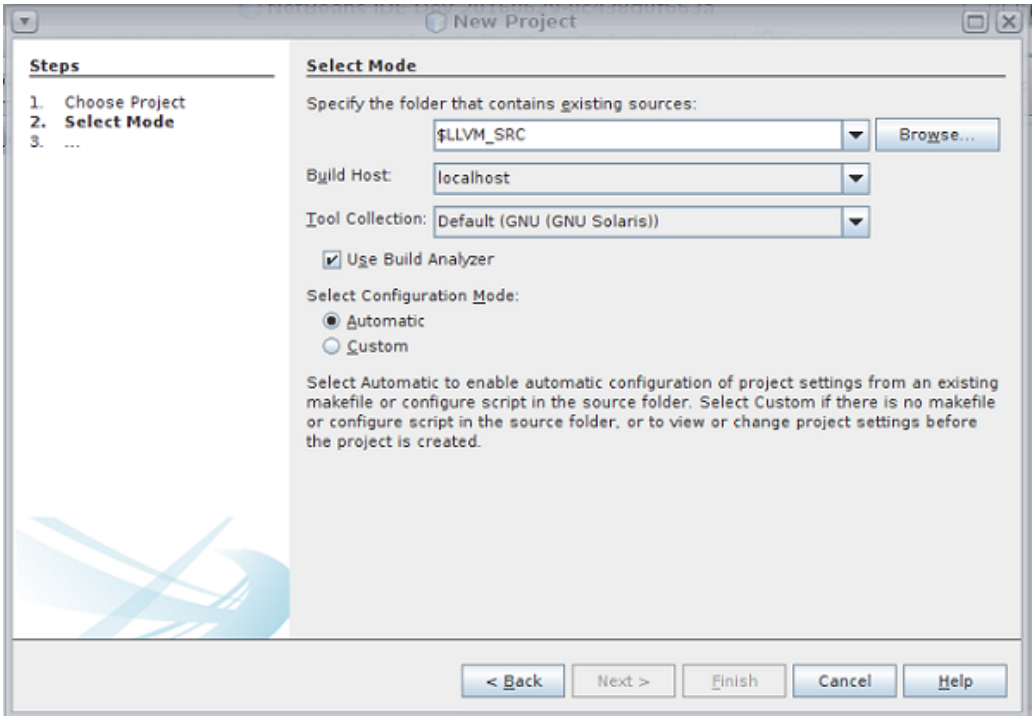
図 4 ツールコレクションパスの変更



環境変数の追加

環境変数は、既存のソースまたはライブラリからプロジェクトを作成するときに使用できます。

図 5 環境変数の追加



ビルドアナライザ

IDE では、Windows および Mac 上にツールコレクションラッパーを作成します。このラッパーはプロジェクトを構築するために使用されます。その結果、コンパイルのすべてのコマンド行が保存され、コード支援を構成するために使用されます。

図 6 ビルドアナライザ

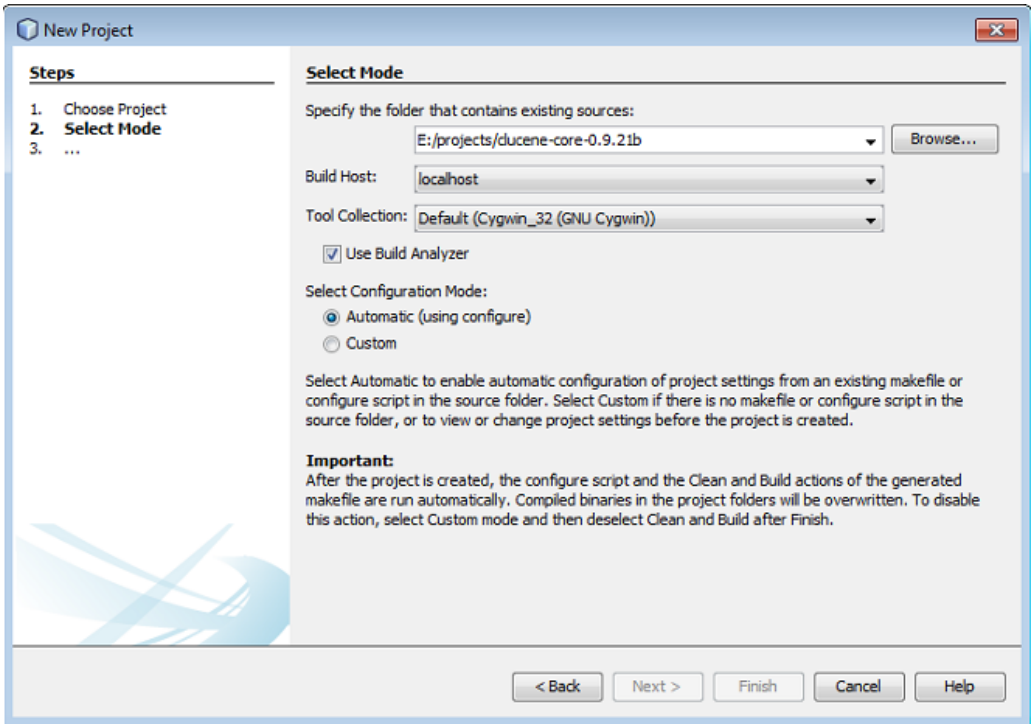
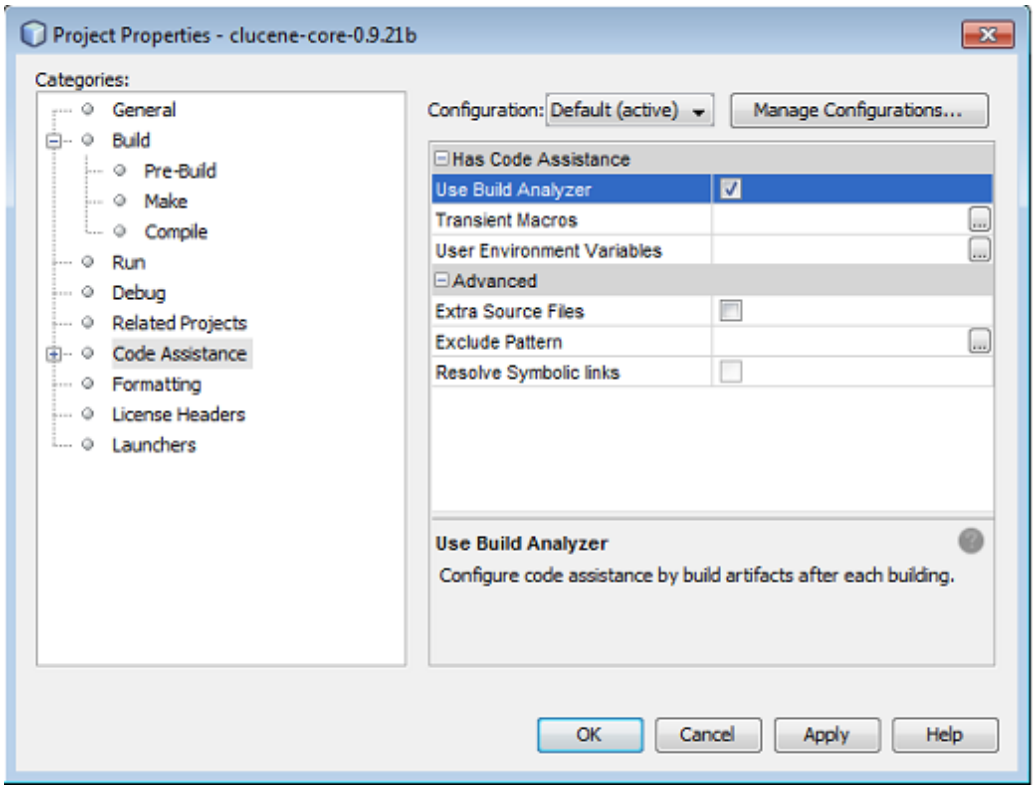

New Project
Steps
1. Choose Project
2. Select Mode
3. ...
Select Mode
Specify the folder that contains existing sources:
E:/projects/clucene-core-0.9.21b
Browse...
Build Host:
localhost
Tool Collection:
Default (Cygwin_32 (GNU Cygwin))
Use Build Analyzer
Select Configuration Mode:
Automatic (using configure)
Custom
Select Automatic to enable automatic configuration of project settings from an existing makefile or configure script in the source folder. Select Custom if there is no makefile or configure script in the source folder, or to view or change project settings before the project is created.
Important:
After the project is created, the configure script and the Clean and Build actions of the generated makefile are run automatically. Compiled binaries in the project folders will be overwritten. To disable this action, select Custom mode and then deselect Clean and Build after Finish.
< Back
Next >
Finish
Cancel
Help

図 7 コード支援の構成



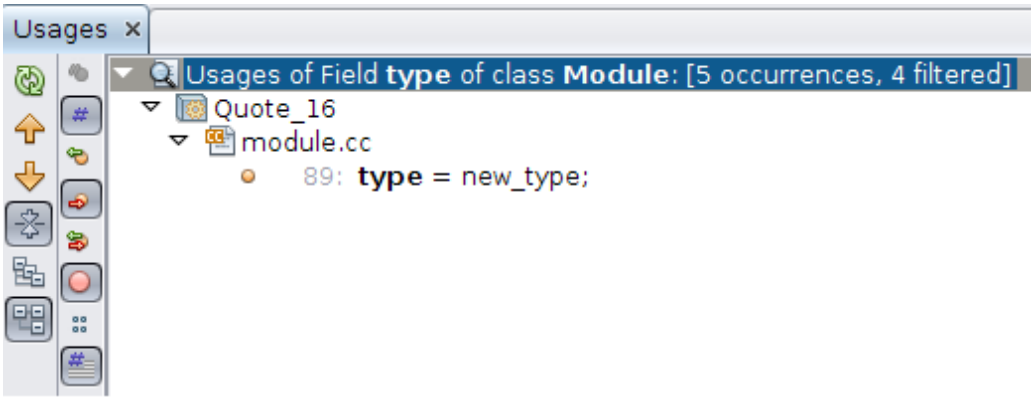
C/C++ キャッシュの消去

新しいオプション「*C/C++* キャッシュを消去して *IDE* を再起動」が「コード支援」の下に追加されました。このオプションを使用すると、キャッシュディレクトリを消去して、IDE を再起動できます。

読み取り/書き込みアクセスの区別

付与されているアクセス権に基づいて変数にマークを付けることができます。読み取りアクセス、書き込みアクセス、または読み取り/書き込みアクセスを持つ変数は別々に表示されます。

図 8 読み取り/書き込みアクセスの区別



ターミナル機能の拡張機能

ターミナル機能に次の拡張機能が追加されました。

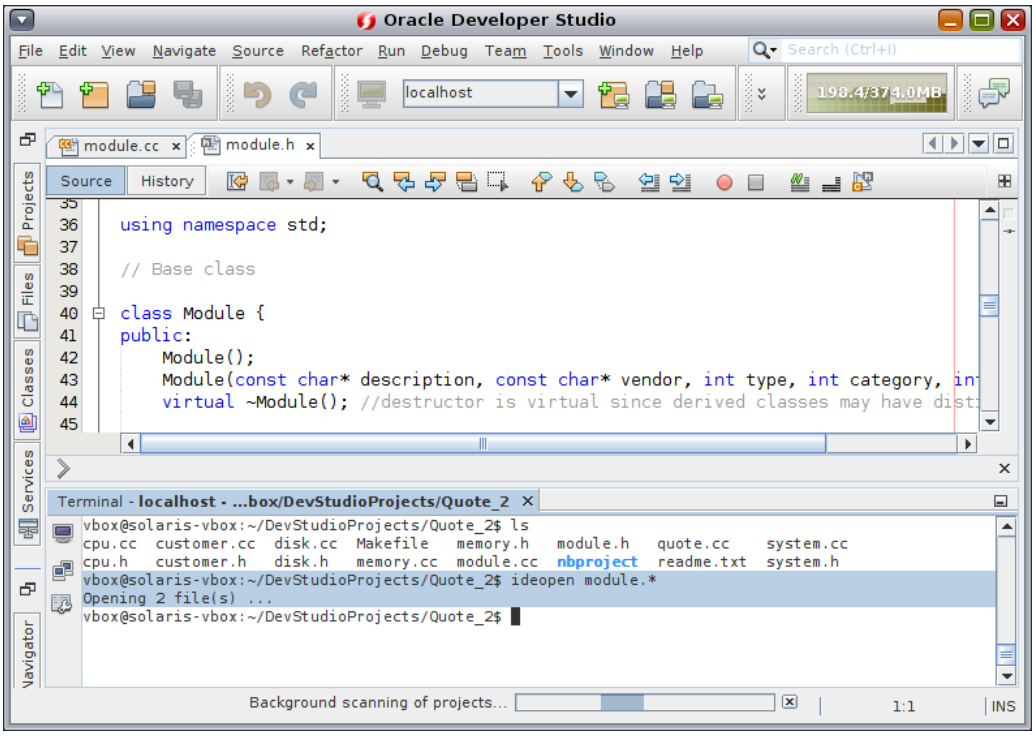
- **ターミナルからエディタでファイルを開く** - 「ウィンドウ」->「*IDE* ツール」->「ターミナル」の順に選択し、プロジェクトフォルダに移動します。次のコマンドを入力すると、指定されたファイルをエディタで開くことができます。
  - 1 つのファイルを開くには、次のように入力します。

    ```
    ideopen module.cc
    ```
  - 複数のファイルを開くには、次のように入力します。

    ```
    ideopen module.cc module.h
    ```
  - (現在のシェルでサポートされている) ワイルドカードを使用して複数のファイルを開くには、次のように入力します。

    ```
    ideopen module.*
    ```

図 9 エディタでファイルを開く



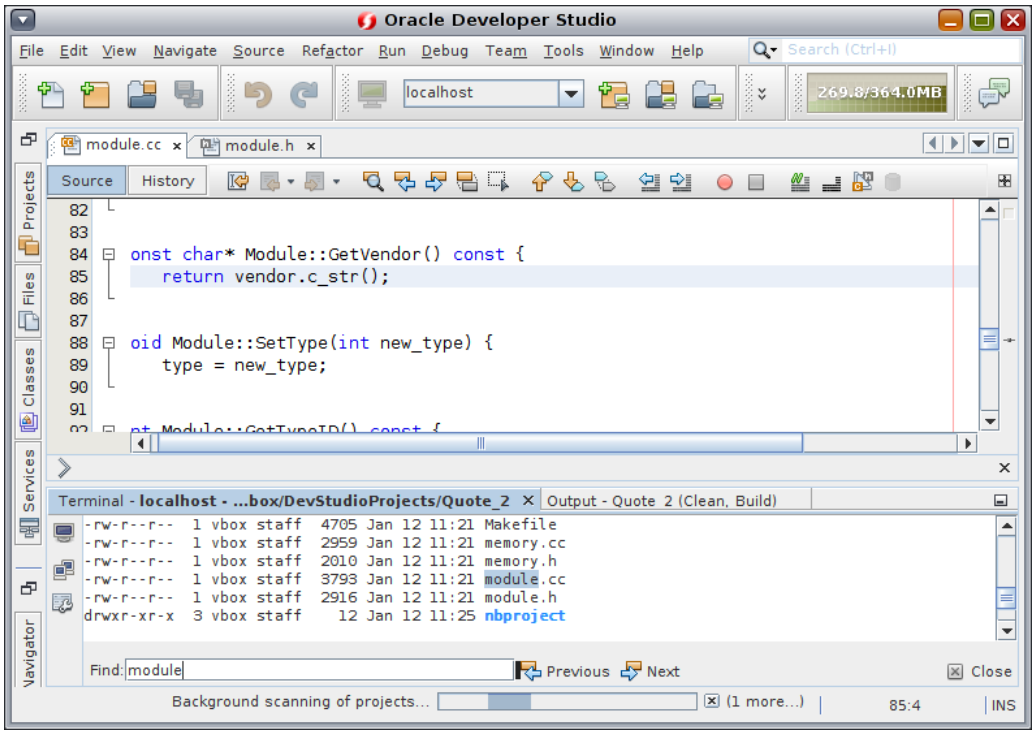
- **キーボードショートカットのサポート** - この機能により、ユーザーはキーボードを使用して新しいタブを開いたり、開いているタブを切り替えたりできます。新しいタブを開くには *Ctrl+Alt+Shift+T* を使用し、タブを切り替えるには *Alt+1*、*Alt+2* ... を使用します。
- **ターミナルからハイパーリンクへのアクセス** - プログラムの出力にターミナル内のファイルへのリンクが出力されている場合、ユーザーがターミナルでそのリンクをクリックすると、IDE によってそのファイルがエディタで開きます。エスケープシーケンスを使用してハイパーリンクをターミナルで出力できます。次に例を示します。

```
fprintf(stdout, "(\033]10;%s;%s\007)]\n", "/home/ilia/NetBeansProjects/
CppApplication_48/main.cpp:20", "main.cpp:20")
```

- **ターミナルでの検索** - 検索機能がターミナルに実装されました。検索するには、*Ctrl+Shift+F* を使用するか、右クリックして「ターミナルで検索」を選択します。見つかった出現箇所をナビゲートすることもできます。前の出現箇所に移動す

るには *Shift + F3* を使用し、次の出現箇所に移動するには *(F3、Enter)* を使用します。

図 10 ターミナルで検索

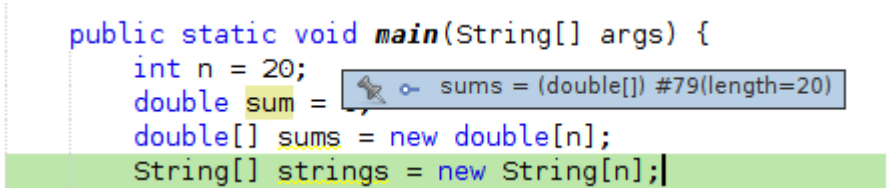


固定可能なウォッチ

ウォッチをエディタ領域に固定できるようになりました。変数または選択項目の上にマウスのポインタを置くと、その値を含むツールチップが表示されます。

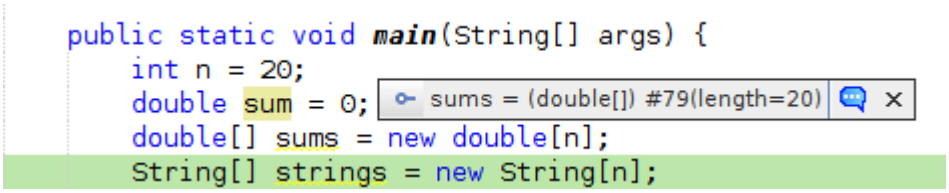
ツールチップにピンアイコンが含まれるようになりました。これをクリックすると、ウォッチが作成され、エディタに固定されます。

図 11 ツールチップ内のピンアイコン



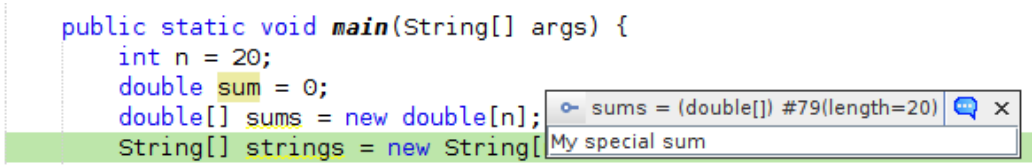
ピンウォッチウィンドウはツールチップの代わりに表示されます。このウィンドウの位置は、マウスでドラッグすることで調整できます。

図 12 ピンウォッチウィンドウ



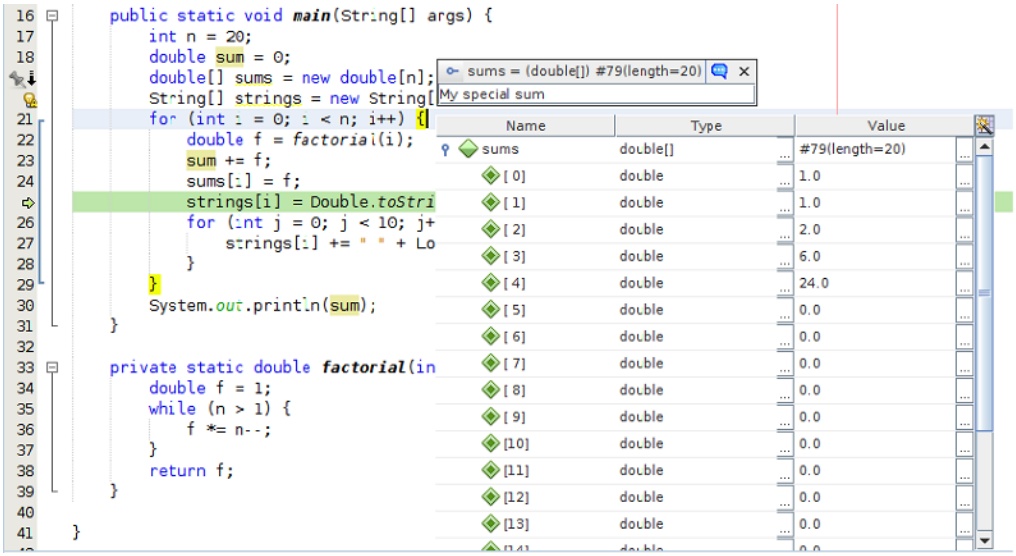
ピンウォッチウィンドウには、右側に「コメント」アイコンと「閉じる」アイコンの 2 つのアイコンが含まれています。「コメント」アイコンをクリックすると、ウォッチに関するコメントを追加できるテキストフィールドが表示されます。

図 13 「コメント」アイコンと「閉じる」アイコン



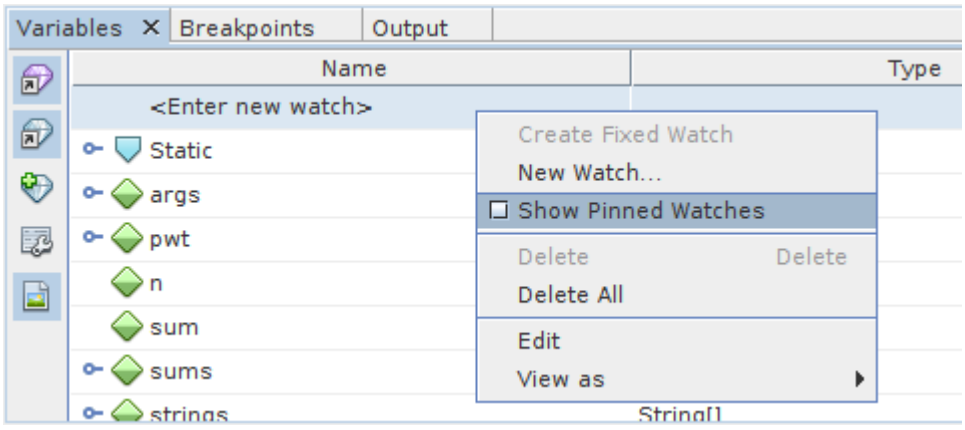
ピンウォッチに構造化された値が表示されるときは、左側に「展開」アイコンが表示されます。展開されると、その子が含まれるビューが表示されます。

図 14 「展開」アイコン



デフォルトでは、「変数」/「ウォッチ」 ウィンドウには固定されたウォッチは表示されません。ただし、「固定されたウォッチを表示」コンテキストアクションを使用することで、固定されたウォッチを表示できます。

図 15 「固定されたウォッチを表示」オプション



Docker

Oracle Developer Studio では、ユーザー向けに Docker をサポートしています。Oracle Developer Studio IDE には Docker ファイルの強調表示とオートコンプリート機能が備わっているため、ユーザーは Docker と容易に対話できます。ユーザーは、自分のマシンまたはリモートマシンで実行されている Docker デーモンに接続できます。ユーザーは、IDE を使用して Docker Hub からイメージをダウンロードし、独自のイメージをビルドして、Docker コンテナを実行および管理できます。

Docker について

Docker とは、軽量アプリケーションコンテナの作成、実行、および管理を提供するオープンソースのコンテナエンジンです。Docker では、ユーザーはパブリックリポジトリを通じて使用できる公開されたイメージからインスタンスを実行できます。Dockerfile と呼ばれるドメイン固有の言語を使用して、新しいイメージを作成することもできます。オーサリングを行うと、任意の Docker 環境でイメージを実行できます。

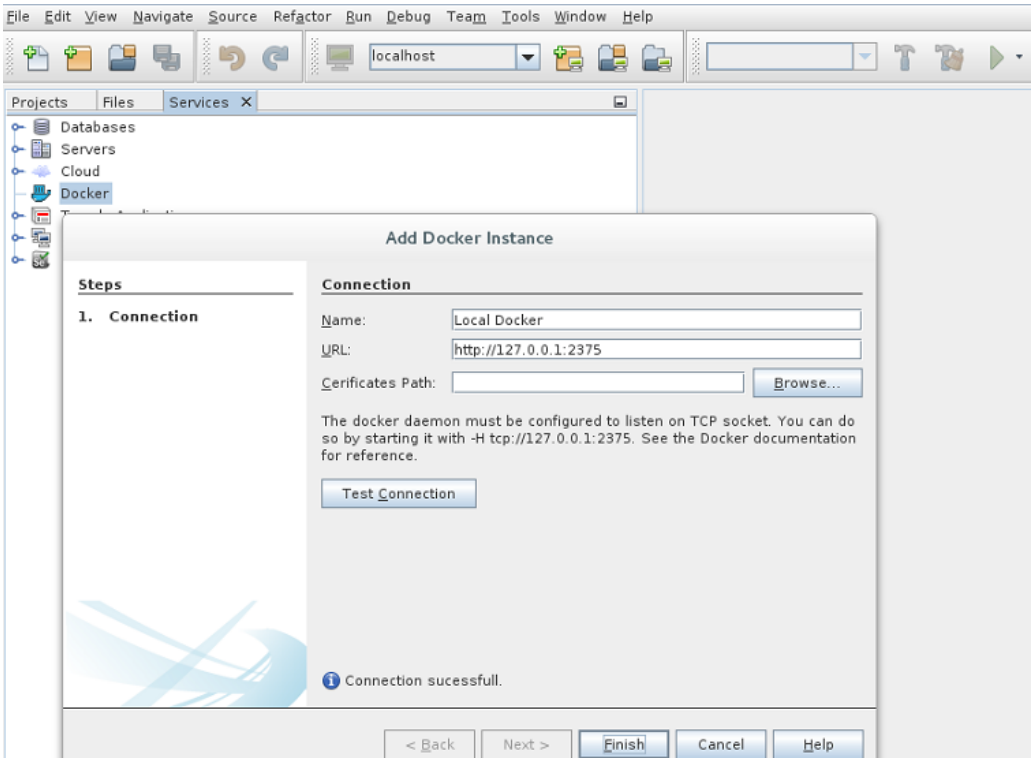
Docker との対話

このセクションでは、Oracle Developer Studio IDE を使用して Docker と対話するステップについて説明します。

▼ Oracle Developer Studio IDE を使用して Docker と対話する方法。

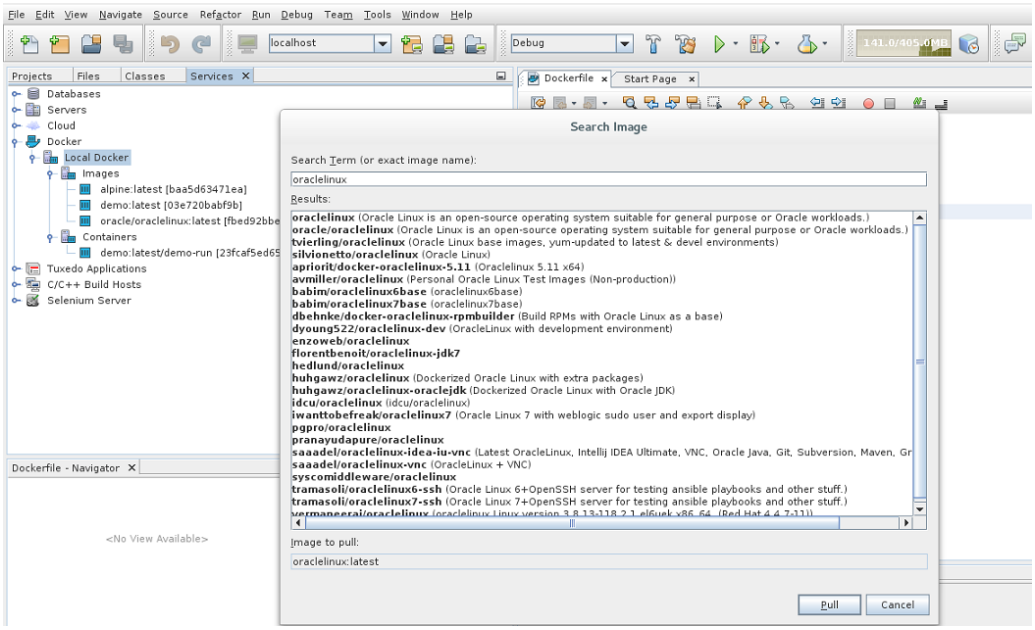
ユーザーは、ローカルマシンまたはリモートマシンで実行されている Docker デーモンに対して次のタスクを実行できます。ただし、リモートマシンで実行されている Docker デーモンと対話すると、クラウド環境で Oracle Developer Studio IDE を使用する機会も得られます。

1. **`http` を使用して Docker デーモンに接続します。**

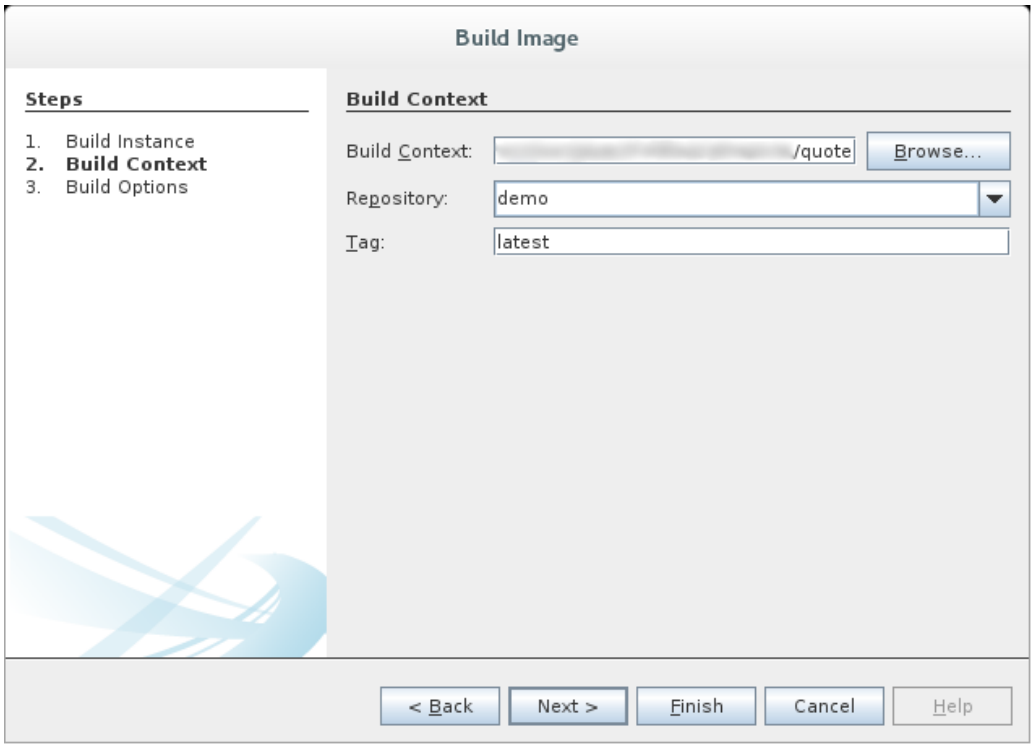


「名前」および「URL」フィールドに Docker の名前と URL をそれぞれ指定して、「完了」をクリックします。

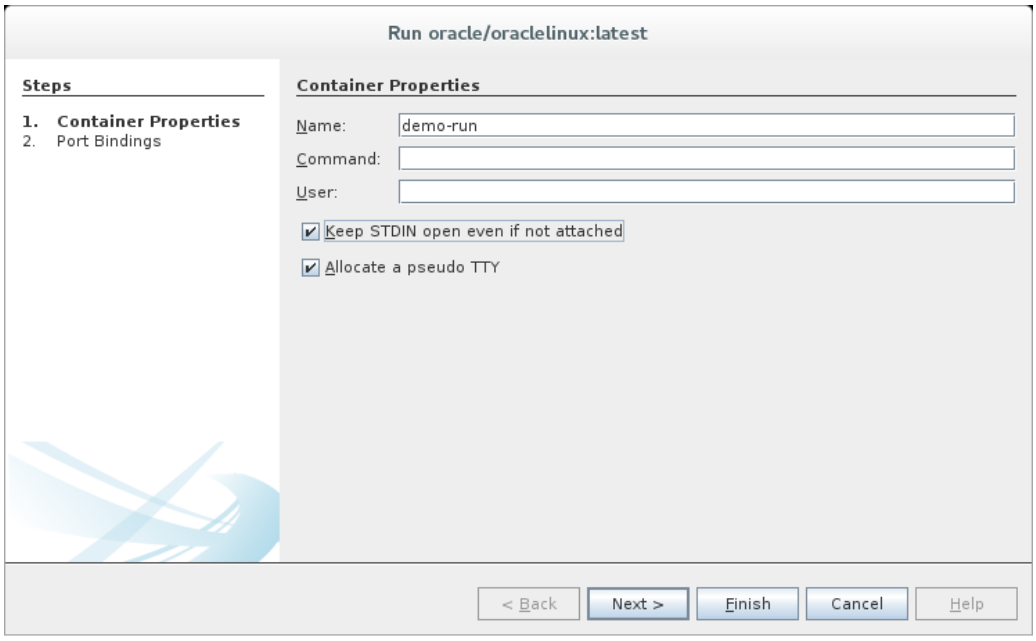
2. 「サービス」ビューで、**Docker Hub からのイメージの検索とダウンロードに使用できるものを使って、新しいノードが表示されます。**



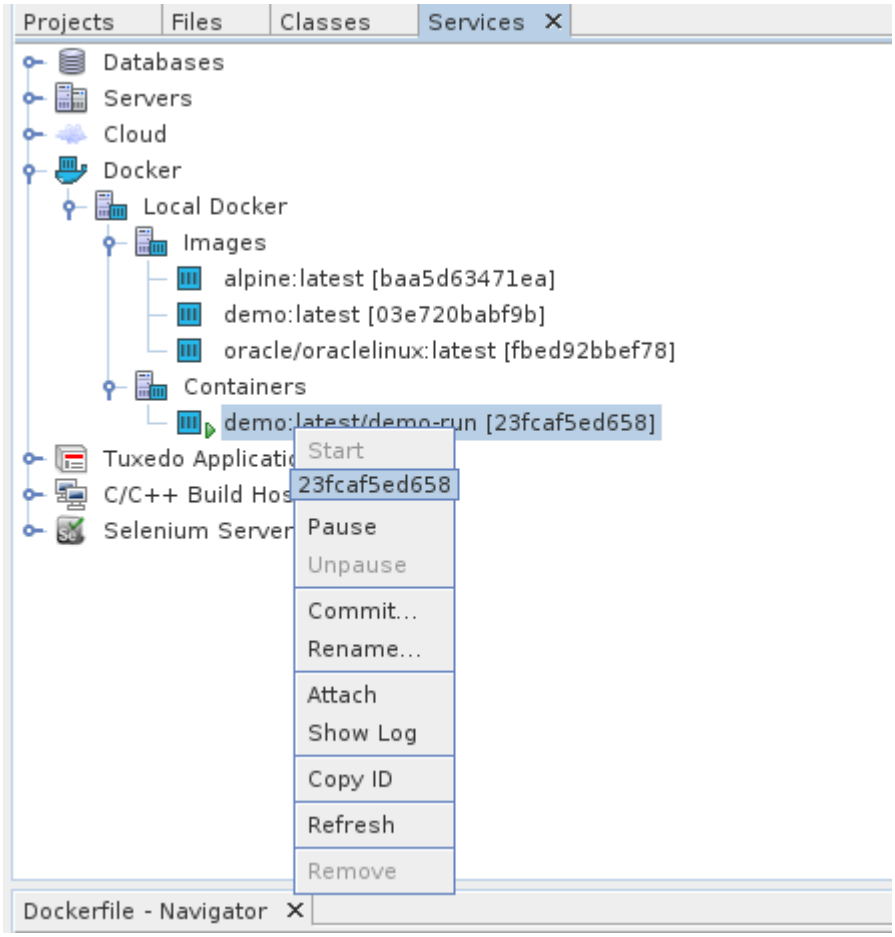
3. 「プロジェクト」ビューで、**Dockerfile を右クリックし、「ビルド」をクリックして、イメージをビルドします。**「イメージのビルド」ウィンドウで、**「ビルド用コンテキスト」を選択して詳細を入力します。**



4. **Docker コンテナを実行します。IDE ターミナルを使用して、コンテナと対話します。**



5. 「サービス」ビューで、コンテナを右クリックしてコンテナを管理します。



♦ ♦ ♦ 第 8 章

8 OpenMP API

この章では、Oracle Developer Studio のこのリリースにおける OpenMP API サポートの変更点について説明します。

OpenMP

このセクションでは、OpenMP API の新機能および更新について説明します。

- 動的ループスケジュールの改善 - OpenMP 4.5 仕様では、動的ループスケジュールによってチャンクがスレッドに割り当てられる方法を明確に示しています。ループ繰り返しのチャンクが単調な順序でスレッドに割り当てられる保証はありません。たとえば、次の OpenMP 動的ループでは、50 番目の繰り返しが 0 番目の繰り返しの前に実行されます。

  ```
  #pragma omp parallel for num_threads(2) schedule(dynamic)
  for (i=0;i<100;i++) A[i] = B[i];
  ```

  これにより、より効率的な動的ループスケジューリングを実装できます。
- OpenMP task プラグマの priority 節 - priority 節の機能には次のものがあります。
  - `OMP_MAX_TASK_PRIORITY` 環境変数
  - `omp_get_max_task_priority()` API ルーチン

  詳細は、**OpenMP 4.5 仕様**のセクション `2.9.1` (task 構文の priority 節)、セクション `4.14` (`OMP_MAX_TASK_PRIORITY` 環境変数)、およびセクション `3.2.36` (`omp_get_max_task_priority` API ルーチン) を参照してください。
- スレッドのアフィニティー (プロセッサバインディング) 用のクエリー関数 - 次のクエリー API ルーチンが追加されました。
  - `omp_get_num_places()`
  - `omp_get_place_num_procs()`
  - `omp_get_place_proc_ids()`
  - `omp_get_place_num()`
  - `omp_get_partition_num_places()`

- `omp_get_partition_place_nums()`

  詳細は、**OpenMP 4.5 仕様**の 256 ページのセクション `3.2.23` から 261 ページのセクション `3.2.28` までを参照してください。

OpenMP の詳細は、『Oracle Developer Studio 12.6: OpenMP API User's Guide』を参照してください。

♦ ♦ ♦ 第 9 章

その他の変更点

この章では、Oracle Developer Studio ソフトウェアのその他のコンポーネントについて、新機能および変更された機能について説明します。

コンパイラに対する変更

次のセクションではコンパイラに対して行われた変更について説明し、内容は次のとおりです。

全コンパイラに共通の新機能および変更点

C、C++、および Fortran コンパイラに対して前のリリースから次の変更が行われました。詳細は、コンパイラのマニュアルページを参照してください。C++ コンパイラに固有の変更点は、第2章「C++ コンパイラ」に詳しく説明されています。C コンパイラに固有の変更点は、第3章「C コンパイラ」に詳しく説明されています。

GCC 互換オプション -fvisibility

`-fvisibility=v` オプションは、次のように `-xldscope` オプションと同等です。

`-fvisibility` Options	**同等の `-xldscope` オプション**
`default`	`global`

-fvisibility Options	同等の -xldscope オプション
internal	hidden
protected	symbolic
hidden	hidden

GCC 互換オプション -shared

Oracle Developer Studio 12.6 では、-shared オプションは GCC と互換性があります。-shared オプションは、-G オプションと同様に、共有ライブラリを作成します。デフォルトでは、-shared オプションは、実行可能プログラムの作成時にリンクされるのと同じライブラリにリンクしますが、-G オプションはデフォルトライブラリに自動的にリンクしません。以前のリリースでは、-shared オプションは -G オプションと同等でした。

新しいコマンドオプション

次のリストに、すべてのコンパイラに共通する新しいコマンドオプションを示します。

- **SPARC M8/T8 の新しいオプション** - -xchip および -xtarget 値が SPARC M8/T8 システムで使用できるようになりました。
- **x86 プロセッサ Skylake の新しいオプション** - -xchip=skylake、-xtarget=skylake、および -xarch=avx512 値が Intel Skylake プロセッサで使用できるようになりました。
- **Linux のデフォルトオプション** - Linux のデフォルトオプションは -xannotate です。
- コマンドオプション -fstrict-aliasing および -fno-strict-aliasing が使用できるようになりました。

Oracle Solaris の新しいマクロ

新しいマクロ __SUNOS_Release が Oracle Solaris でのみ使用できるようになりました。

Oracle Solaris の __SunOS_RELEASE

16 進値 0xRRrrmm は Oracle Solaris リリースを表します。ここで、RR.rr は sysinfo (SI_RELEASE) システムコールまたは uname -r コマンドの出力であり、必要に応じて先頭にゼロが追加されます。mm の数字は、考えられる将来のマイクロリリース用に予約されています。それらの数字はすべて 10 進数です。

たとえば、Oracle Solaris 11 (SunOS 5.11: __SunOS_RELEASE) の値は 0x051100 になります。

古い Oracle Solaris リリースの __SunOS_RELEASE の値は常に、以降のリリースの値より小さくなります。次に例を示します。

```
#if __SunOS_RELEASE >= 0x051100 // Solaris 11 or later
```

コンパイラのインライン化動作

Studio のコンパイラでは、本体が呼び出し側のオーバーヘッドより小さいルーチンを自動的に -O3 レベルでインライン化します。自動的にインライン化される関数を制御するには、`-xinline=list` オプションを使用します。

デバッグセクションの圧縮

`-xcompress=debug` オプションを使用すると、`-xcompress_format` オプションで指定した形式を使用してデバッグセクションを圧縮できます。同等の gcc オプション `-gz` のサポートも追加されました。

`-xdebugformat=stabs` オプション

`-xdebugformat=stabs` オプションが削除されました。

Fortran コンパイラ

Fortran コンパイラは、Fortran77、Fortran90、および Fortran95 規格のための、過去最高のランタイムパフォーマンスおよび互換性オプションにより、科学技術アプリケーション開発をサポートします。Fortran 2003 の機能と OpenMP 4.5 のサポートの大部分が含まれます。Fortran コンパイラでは C および C++ コンパイラと同様のハイパフォーマンスなコード生成テクノロジが使用され、最新の SPARC および x86 ベースの Oracle システムのための最大パフォーマンスの並列コードがアプリケーションで生成されることを保証します。

Fortran コンパイラの変更点には、51 ページの「全コンパイラに共通の新機能および変更点」に記載されている変更点と、次が含まれます。

- Fortran の大域プログラム検査 (GPC) のサポートが削除されました。
- Fortran 2003 標準で導入された、パラメータ化された派生型が現在サポートされています。

詳細は、f95(1) のマニュアルページおよび『Oracle Developer Studio 12.6: Fortran User's Guide』を参照してください。

パフォーマンスライブラリの変更点

Oracle Developer Studio パフォーマンスライブラリは、線形代数や大量の数値計算を伴う問題を解くために最適化された高速な数学サブルーチンのセットです。Oracle Developer Studio パフォーマンスライブラリは、http://www.netlib.org の Netlib からほとんど入手できるパブリックドメインサブルーチンのコレクションに基づいています。Oracle はこれらのパブリックドメインサブルーチンを強化し、Oracle Developer Studio パフォーマンスライブラリとしてバンドルしました。

このリリースでは、Oracle Developer Studio パフォーマンスライブラリの LAPACK がバージョン 3.6.1 にアップグレードされました。LAPACK 3.6.1 のすべての新機能が Oracle Developer Studio パフォーマンスライブラリに実装されています。変更点の詳細は、http://www.netlib.org/lapack/lapack-3.6.1.html を参照してください。

詳細は、『Oracle Developer Studio 12.6: Performance Library User's Guide』を参照してください。

索引