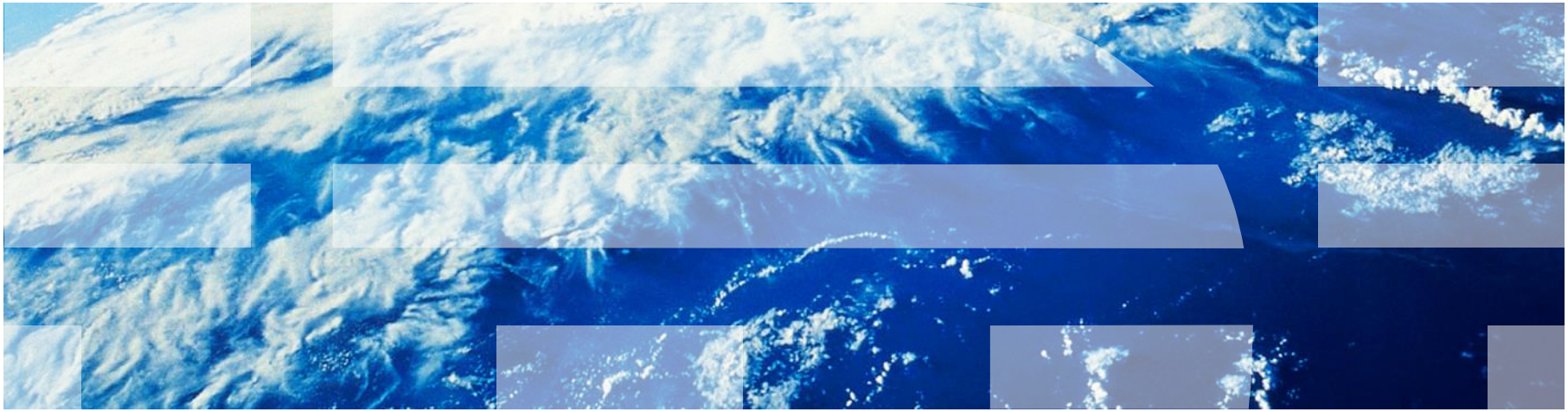
IBM Worklight Foundation V6.2.0 入門

JSONStore – Java API



2014 年 9 月 5 日

商標

- IBM、IBM ロゴ、ibm.com および Worklight は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- この資料は、事前に IBM の書面による許可を得ずにその一部または全部を複製することは禁じられています。

IBM® について

- http://www.ibm.com/ibm/us/en/ を参照してください。

アジェンダ

- JSONStore とは
- コード・サンプルの理解
- コード・サンプルのウォークスルー
- 予期される出力
- 詳細情報

JSONStore とは

- クライアント・サイドのオプションの API で、以下の主要な機能があります。
  - 効率的な検索のためのデータ索引付け。
  - 実稼働環境でのデータ暗号化。
  - 保管データに対するローカルのみの変更を追跡するためのメカニズム。
  - 複数ユーザーのサポート。
- 使用可能な環境:
  - ネイティブ: Android および iOS
  - ハイブリッド: Android、iOS、Windows Phone 8、および Windows 8
  - Preview Common Resources / Mobile Browser Simulator (実動使用の場合以外)

注: このモジュールは、JSONStore API から始める方法を示しています。一部の機能 (データ暗号化など) は、このモジュールでは扱いません。すべての機能についての詳しい記述は、IBM Worklight® Foundation ユーザー文書にあります。

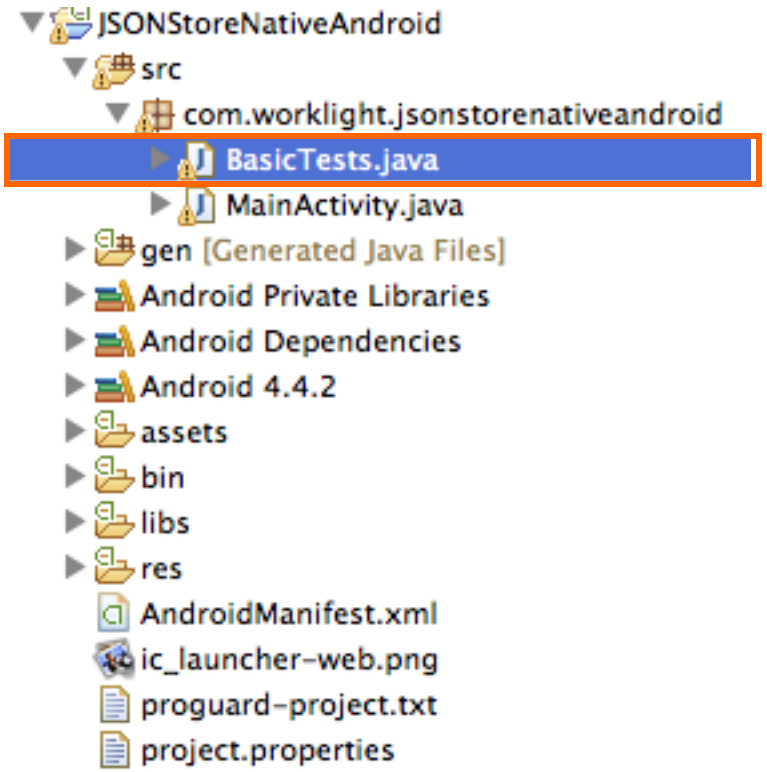
アジェンダ

- JSONStore とは
- コード・サンプルの理解
- コード・サンプルのウォークスルー
- 予期される出力
- 詳細情報

コード・サンプルの理解

1. このモジュールと関連するコード・サンプルを含む圧縮ファイルをダウンロードします。
2. `BasicTests.java` *ファイルを開きます。* コンテキストについてはサンプル・イメージを参照してください。
3. Android JUnit を使用してアプリケーションを実行します。アプリケーションを右クリックし、**「実行 (Run As)」**を選択してから **「Android JUnit Test」**をクリックします。

注: このコード・サンプルでは、Android の組み込み JUnit Test フレームワークが使用されます。フレームワークの動作については、このモジュールでは説明しません。



アジェンダ

- JSONStore とは
- コード・サンプルの理解
- コード・サンプルのウォークスルー
- 予期される出力
- 詳細情報

コード・サンプルのウォークスルー (1/9)

`destroy` API は、アプリケーションから JSONStore のすべての内容を削除します。ここでは、データのない状態で開始するために、これを使用します。こうすれば、コード・サンプル内で出力を予測することができます。

注: コンテキスト (`ctx`) については、このモジュールでは説明しません。このオブジェクトについて詳しくは、Android API の資料を参照してください。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

コード・サンプルのウォークスルー (2/9)

データをパーシストするには、少なくとも 1 つのコレクションを最初に定義する必要があります。これらのコレクションは、データを保持するエンティティーです。ここでは、`people` と呼ばれるコレクションの定義を確認することができます。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

コード・サンプルのウォークスルー (3/9)

検索フィールドは、コレクション内の索引付けされたフィールドです。コレクション内のデータを検索するとき、これらのフィールドを使用できます。

ここでは次の 2 つの検索フィールドの定義を確認することができます。

- `name` (ストリング)
- `age` (整数)

入力データをより適切に保管するために、string、integer、number、Boolean などのデータ型が使用されます。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

コード・サンプルのウォークスルー (4/9)

1 つ以上のコレクションを開くためには `open` API が使用されます。当該コレクションがこれまで一度も開かれたことがない場合は、そのコレクション内のデータをパーシストするためにファイル・システム上にファイルが作成されます。この操作が終了する前に、そのファイルへのアクセサーが作成されます。

このアクセサーにより、呼び出し元は `add` や `findAll` などのコレクション・レベル API を呼び出せるようになります。これらの API は、このコード・サンプル・ウォークスルーの後の方に出てきます。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

コード・サンプルのウォークスルー (5/9)

`people` コレクション内に保管されるデータがここで定義されます。このデータは `name` と `age` のキー値ペアを持つ 2 つの JSON オブジェクトから成るハードコーディングされた配列であることに注意してください。このデータは複数のソース (例: ネットワーク要求、ファイル入出力、ユーザー入力) から獲得することができます。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

コード・サンプルのウォークスルー (6/9)

コレクション・アクセサーが、people コレクション内にデータを保管するためのアクセスを提供します。入力データは JSON 形式でなければなりません。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

コード・サンプルのウォークスルー (7/9)

JSONStore コレクション内のドキュメントを検索するいくつかの異なる方法があります (例: `find`、`findById`)。

最も簡単な方法 (ここで示す方法) は、`findAll` API を使用することです。この方法では、コレクション内に保管されているすべてのデータが返されます。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

コード・サンプルのウォークスルー (8/9)

コレクション内に保管されているデータはドキュメントと呼ばれます。

ドキュメントは _id と json のキー値ペアを持っています。_id ペア は、データ追加時に自動的に追加される内部 ID です。json ペアには、追加されたすべてのデータが含まれます。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

コード・サンプルのウォークスルー (9/9)

エラーが発生した場合は、`JSONStoreException` オブジェクトがスローされます。この例外には、発生したエラーに関する情報が含まれます。

```
try {
    Context ctx = getContext();

    WLJSONStore.getInstance(ctx).destroy();

    List<JSONStoreCollection> collections =
            new LinkedList<JSONStoreCollection>();
    JSONStoreCollection people = new JSONStoreCollection("people");
    people.setSearchField("name", SearchFieldType.STRING);
    people.setSearchField("age", SearchFieldType.INTEGER);
    collections.add(people);

    WLJSONStore.getInstance(ctx).openCollections(collections);

    JSONObject data1 = new JSONObject("{age: 20, name: 'carlos'}");
    JSONObject data2 = new JSONObject("{age: 30, name: 'mike'}");

    people.addData(data1);
    people.addData(data2);

    List<JSONObject> results = people.findAllDocuments();

    assertEquals(
        new JSONObject("{_id: 1, "
                + "json: {'name' : 'carlos', 'age' : 20}}")
            .toString(),
        results.get(0)
            .toString());

    assertEquals(
        new JSONObject("{_id: 2, "
                + "json: {name: 'mike', age: 30}}")
            .toString(),
        results.get(1)
            .toString());
}
catch (JSONStoreException ex) {
    throw ex;
}
```

アジェンダ

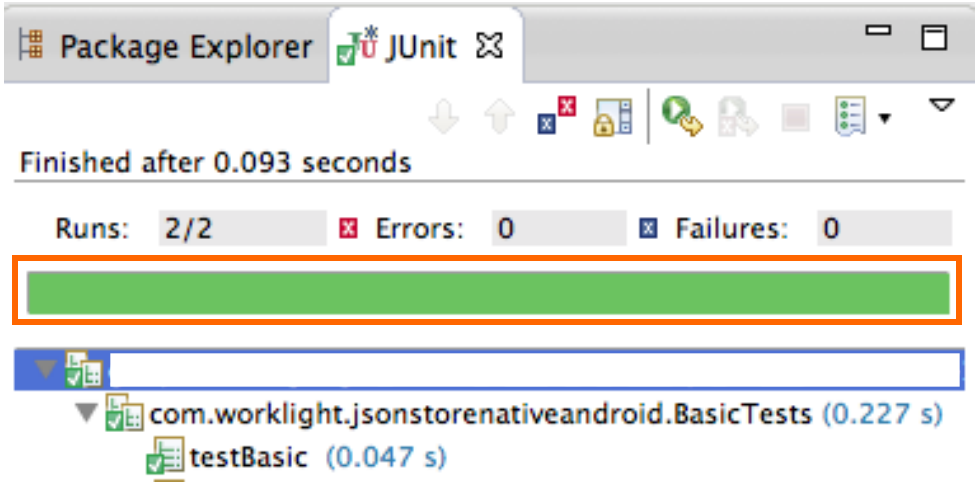
- JSONStore とは
- コード・サンプルの理解
- コード・サンプルのウォークスルー
- 予期される出力
- 詳細情報

予期される出力

テストを実行するには、「**プロジェクト (Project)**」を右クリックし、>「**指定して実行 (Run As)**」>「**Android JUnit テスト (Android JUnit Test)**」を選択します。

テストが実行されると、出力はサンプル・イメージのように表示されます。

テストの上にある緑のバーは、すべてが予想どおりに進んでいることを示しています。



アジェンダ

- JSONStore とは
- コード・サンプルの理解
- コード・サンプルのウォークスルー
- 予期される出力
- 詳細情報

詳細情報

- JSONStore について詳しくは、製品ユーザー文書の『JSONStore』を参照してください。
- JSONStore のパフォーマンスおよびベスト・プラクティスについて詳しくは、製品ユーザー文書の『JSONStore パフォーマンス (JSONStore performance)』を参照してください。

特記事項

- これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。
- 本書は米国 IBM が提供する製品およびサービスについて作成したものです。
- 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作または サービスの 評価および検証は、お客様の責任で行っていただきます。
- IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。
  - 〒103-8510
    東京都中央区日本橋箱崎町19番21号
    日本アイ・ビー・エム株式会社
    法務・知的財産
    知的財産権ライセンス渉外
- **以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含む すべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。**
- この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。
- 本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。
- IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、 自ら適切と信ずる方法で、使用もしくは配布することができるものとします。
- 本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- 本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。
- 本書で説明されているライセンス・プログラムまたはその他のライセンス資 料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。
- IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

著作権使用許諾:

- 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。
- それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。
  - © (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

プライバシー・ポリシーの考慮事項

- サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。
- このソフトウェア・オファリングは、展開される構成に応じて、(アプリケーション・サーバーが生成する) セッション情報を収集するセッションごとの Cookie を使用する場合があります。これらの Cookie は個人情報を含まず、セッション管理のために要求されるものです。加えて、匿名ユーザーの認識および管理のために持続的な Cookie が無作為に生成される場合があります。これらの Cookie も個人情報を含まず、要求されるものです。
- この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

サポートおよびコメント

- IBM Worklight の一連の文書、トレーニング資料、および質問をポストできるオンライン・フォーラムはすべて、次の IBM Web サイトからご覧になれます。
  - http://www.ibm.com/mobile-docs
- **サポート**
  - ソフトウェア・サブスクリプション & サポート (ソフトウェア・メンテナンスと呼ばれる場合もあります) は、パスポート・アドバンテージおよびパスポート・アドバンテージ・エクスプレスから購入されたライセンスに含まれています。International Passport Advantage Agreement および IBM International Passport Advantage Express Agreement の追加情報については、次のパスポート・アドバンテージ Web サイトを参照してください。
    - http://www.ibm.com/software/passportadvantage
  - ソフトウェア・サブスクリプション & サポートが有効になっている場合、IBM は、インストールおよび使用法 (ハウツー) に関する短期間の FAQ に対するサポートや、コード関連の質問に対するサポートを提供します。詳しくは、次の IBM ソフトウェア・サポート・ハンドブックを参照してください。
    - http://www.ibm.com/support/handbook
- **ご意見**
  - 本資料に関するご意見をお寄せください。本資料の具体的な誤りや欠落、正確性、編成、題材、または完成度に関するご意見をお寄せください。お寄せいただくご意見は、本マニュアルまたは製品の情報、およびその情報の提示方法に関するもののみとしてください。
  - 製品の技術的な質問および情報、および価格については、担当の IBM 営業所、IBM ビジネス・パートナー、または認定リマーケターにお問い合わせください。
  - IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。IBM またはいかなる組織も、お客様から提示された問題についてご連絡を差し上げる場合にのみ、お客様が提供する個人情報を使用するものとします。
  - どうぞよろしくお願いいたします。
  - 次の IBM Worklight Developer Edition サポート・コミュニティーにご意見をお寄せください。
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - IBM からの回答を希望される場合は、以下の情報をご連絡ください。
    - 氏名
    - 住所
    - 企業または組織
    - 電話番号
    - E メール・アドレス

ありがとうございました