「Internet of Things」入門

実際に組込みアプリケーションを作成することで Internet of Things に関連する概念や問題を理解する

ERIC J. BRUNO

ソフトウェア・アーキテクチャや開発のトピックを扱うライター、編集者。IT コミュニティにおいて 20 年以上の実績がある。デスクトップからデータセンターに至るさまざまなトピックについて講演。世界中の企業でエンタープライズ・アーキテクト、開発者、業界アナリストも務める。

多くの企業は、身の回りのあらゆるスマート・コンピューティング・デバイスが相互に通信し、さらによく利用するエンタープライズ・アプリケーションとも通信するような未来を約束しています。この手法は、マシン・ツー・マシン（M2M）アーキテクチャとも呼ばれます。オラクルも、Internet of Things（モノのインターネット：IoT）を重視した計画を立てています。このような IoT の約束を実現することは、モバイル・デバイスで Bluetooth や Wi-Fi をオンにする程度の単純なことに思えるかもしれません。しかし、すべてが接続された未来の世界を最大限に活かすためには、もっと深い部分で統合する必要があります。

デバイスとは

IoT のアーキテクチャや戦略の検討における最初の手順は、デバイスとは何かを定義することです。ほとんどの人はまず、デバイスをスマートフォンやタブレットと定義しますが、より小さく低機能の携帯電話もデバイスとして適格です。センサー、電気制御装置、RFID リーダー、最新の家庭用電化製品なども、M2M や IoT の世界においてそれぞれの役割を果たします。基本的に、一意に識別でき、適切なリソースやロケーションに関するデータを持つすべての電子的なエンドポイントは、IoT アーキテクチャにおける有効なデバイスとして適格です。

デバイスの種類によってデータ量や実用性は異なりますが、大量のデータを組み合わせることで、非常に単純なデータからでも途方もない価値を引き出すことができます。重要なのは、デバイス同士の通信や、さらには協調動作までも可能になるように、すべてのデバイスを識別し、まとめて接続することです。この識別と接続こそ、インターネットが重要な役割を果たす分野です。

Internet of Things

1999 年、マサチューセッツ工科大学 Auto-ID センターの共同創設者である Kevin Ashton 氏は、Internet of Things という用語を考案し、あらゆる場所にあるセンサーを通じて、インターネットが実世界に接続されるシステムと定義しました。Ashton 氏の構想は、データ入力を人間だけに頼らずに、独自のデータを収集するコンピュータ、デバイス、センサーによる地球規模のシステムを構築することでした。結果として、この地球規模のシステムでは、何がどこにあるのか、どの場所でどの時点に何が起きているのかを把握できます。その先につながるのは、大幅な無駄の削減、コストの低減、人間が機械を操作することによる損失（定義は人が行うのですが）の排除が可能となる、相互に接続されたシステムの世界でしょう。

研究者は、アドレス指定能力、通信規格、電池残量、データ収集などの問題を懸命に解決しようとしています。一方、各企業も、ギャップを埋め、標準プラットフォームを定義し、すべての関係者がメリットを享受できるエコシステムの構築を急いでいます。IoT はごく初期段階にあり、現在は IoT の成立に関わることのできる面白い時期です。

成功に不可欠
おそらく、IOT 戦略を推進する今日のテクノロジーの中では、**Java がもっとも良い位置に**あるでしょう。

Java と IoT：ハイパーコネクティビティ向けのプラットフォーム

Java は組込みデバイス上で誕生しました。当初、Star7 携帯端末に搭載される Green Project の一部であった Java は、その後、Web と組込みの双方のドメインにおいて大きな成功を収めました。おそらく、IoT 戦略を推進する今日のテクノロジーの中では、Java がもっとも良い位置にあるでしょう。

CPU やメモリに制限のあるフィーチャーフォンから、処理能力や容量が豊富なラックマウント型サーバーに至るまで、Java はさまざまなデバイス上で稼働できます。そのことを考慮すれば、Java は、ユビキタスな接続性、つまりハイパーコネクティビティを実現したコンピューティング・リソースの世界を推進することを定め

られていると言えます。このエコシステム内の Java 搭載デバイスは、デバイス同士で通信し、ユーザーと通信し、さらにセンサーや中継器を通じて実世界と通信することで、知識を収集し、究極的には周囲の世界について解釈した情報もリアルタイムで収集します。

医療、自動車（テレマティクス）、ビルディング / 工業オートメーション、スマート・メーター、交通、物流などの分野において、豊富な情報を発見してレポートできることを想像してみてください。そのほんの数例について紹介しましょう。

- 内容物を示すビンに入った薬物に対して患者がアレルギーを持っていないかを自動化されたデバイスでクロスチェックすることで、潜在的で命取りになるミスを予防可能
- 車両に搭載されたテレマティクス・システムによって、渋滞の発生前に運転手に対して自動的に警告して経路を変更することで、交通状況を改善可能
- 家庭やオフィス・ビル内のゲートウェイ・デバイスで、電力需要の時間的変動の山と谷をより正確にレポートして予測することで、より効率的な発電を実現可能

オラクルは、Java ベースのミドルウェア、データ・ストレージ、分析手法を使用し、そのすべてが適切なハードウェアで稼働する標準の Java ベース IoT プラットフォームの定義に貢献しています。この IoT プラットフォームを利用することで開発者はイノベーションやソリューションに集中できます（図 1）。センサーに Oracle Java ME Embedded を使用することで、エンド・ツー・エンドのソリューション全体で同じ言語を使用し、デバイスの移植性と継続的なサポートを維持できるため、システムを効率的に構築できます。さらに、Oracle Java SE Embedded や Oracle Java Embedded Suite により、高速かつ低コストの M2M ゲートウェイを配備するために必要となる機能（エンド・ツー・エンドでのデータ取得、イベント処理、分析）が利用できます。

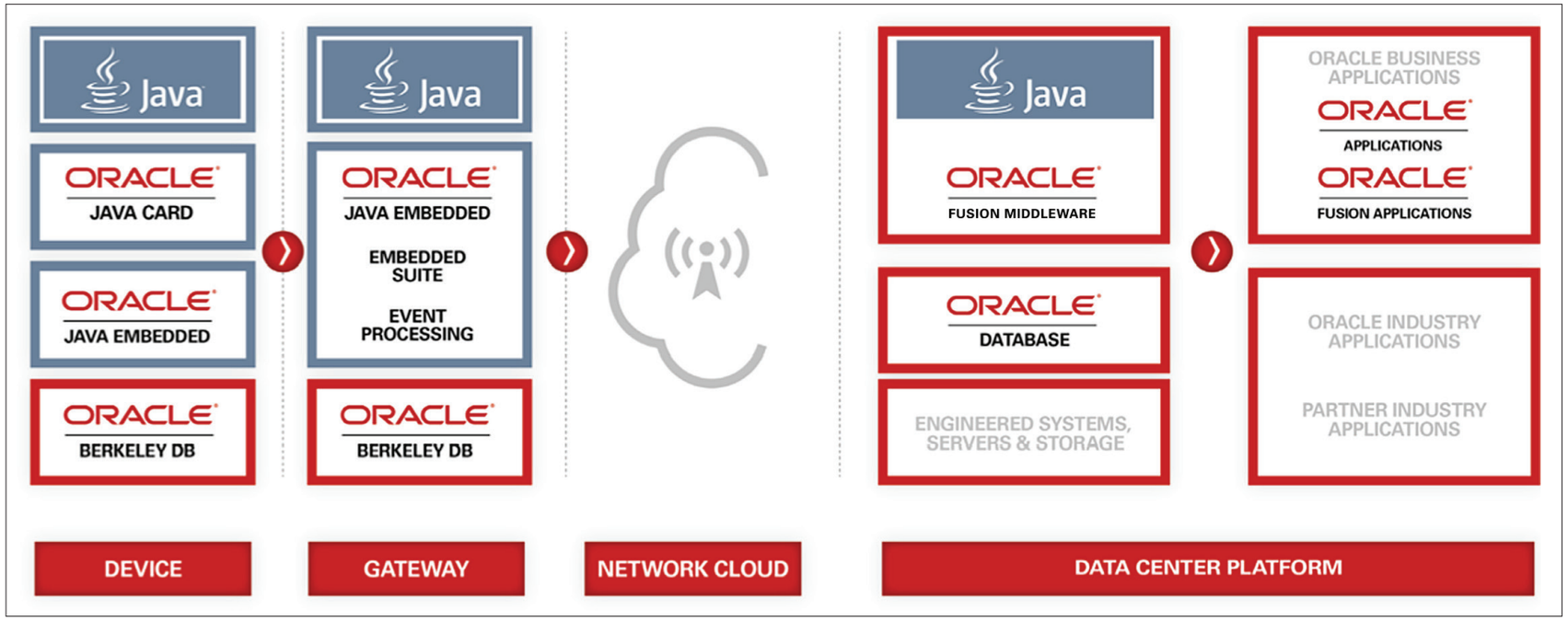

図 1

ただし、完全な IoT アーキテクチャは Java の範囲に収まらず、単純に特定のソフトウェア・スイートを使用すれば良いというものではありません。エンド・ツー・エンドのセキュリティ、将来のニーズに応じた拡張性やスケーラビリティ、業界における標準規格およびセキュリティに関する要件への対応、IT インフラストラクチャの観点からのコスト削減、既存のシステムとの統合、あらゆる層での十分なパフォーマンスなどをおもなメリットとする、マネージド・ソリューションを必要とするものです。これ以降、オラクルの IoT プラットフォームについて詳しく見ていきます。

プロビジョニングと管理：IoT ソリューションの各地点で、デバイス、データ、およびユーザーの識別、認証、および認可が必要になります。たとえば、ネットワーク内のセンサーから情報がレポートされたときには、何らかの判断を行う前に、そのセンサーが信頼でき、正しく識別できるかを確認することが不可欠です。さらに、現実のシナリオでは政府による規制や倫理的な制限があります。そのため、データが悪人の手に渡らず、そのデータの参照権限のある人（例：特定の患者について医学的診断を行う医師）だけが参照できるようにする必要があります。

新しいデバイスのプロビジョニング、ネットワーク内におけるデータの出所の特定、適切なユーザーへの適切なタイミングでの権限付与が、IoT プラットフォームで必要となる極めて重要な機能です。小さなセンサーや分散されたコンピューティング・ノードにより膨張するネットワークを効率的に（かつ一元的に）管理できることも必要です。

セキュリティ：前述のとおり、ユーザーやデバイスの識別はアプリケーションのセキュリティに不可欠です。オラクルの IoT アーキテクチャでは、顧客の LDAP システムへのエントリ・ポイント（ユーザー識別管理のため）、ネットワーク機器や業界標準プロトコルへのエントリ・ポイント（セキュアな通信のため）、安全なオフサイト・データ・ストレージへのエントリ・ポイント（必要となるシナ

JavaOne 2013 での IoT

JavaOne と Oracle OpenWorld では、開催期間全体を通じて、「IoT in Motion」によって参加者の移動状況を追跡しました。IoT in Motion は、オラクル、Eurotech、Hitachi Communication Technologies America（Hitachi CTA）、Hitachi Consulting による共同の「Internet of Things」であり、さまざまな会場のカンファレンス参加者をカウントして追跡しました。

オラクルの Jennifer Douglas がテクノロジーの概要について次のように説明しました。「データを運用する実際のアプリケーションの構築を手がけたのは Hitachi Consulting です。このデータ運用には、Oracle Exalytics と Oracle Business Intelligence ダッシュボードを使用しています。また、Hitachi CTA の SuperJ を Oracle Java SE Embedded と連結し、ゲートウェイを経由して Eurotech Everyware Cloud に接続して、このクラウドで生データを収集します。さらに、Oracle Exalytics がデータを集め、実際に利用できる形式に変換します」すべてのテクノロジーが Java 上で稼働しました。

IoT in Motion は犬や車両と人間とを区別できますが、立体カメラで実施することは空間を往来する人の登録とカウントのみであり、個人の特徴については一切監視されません。

—Janice J. Heiss

リオ向け）をそれぞれ定義しています。エンド・ツー・エンドのセキュリティは十分詳細に定義して適用する必要があり、かつ、システムの拡張と利便性を実現できるほど柔軟である必要もあります。

接続性：デバイス接続は、イーサネットなどの基本プロトコルから始まりますが、センサーや機密データ向けの通信規格もサポートする必要があります。業界プロトコルの例として、DASH7（ワイヤレス・センサー・ネットワーク向け）、ZIGBEE（電気制御装置向け）、6LoWPAN（ビルディング・オートメーション、スマート・グリッド・アプリケーション向け）、COSEM（スマート・メーター向け）、Bluetooth（テレマティクス・システムなどの広範な用途向け）などがあります。

プロトコルのほかにも、IoT プラットフォームは、利用中に接続と切断を繰り返す可能性のあるデバイスに対応し、必要に応じて確実にデータを保存して後で必ず転送する必要があります。

イベント処理：ネットワーク内にある数百、あるいは数千ものエンドポイントからデータを収集する際に、データから実用的な情報をリアルタイムで取得できる必要があります。IoT プラットフォームに組み込まれた複雑なイベント処理ソフトウェアにより、データ全体をデータセンターに送信する前に、データへのフィルタ適用やデータに関する判断を行うことが可能です。そのような事前の操作によって、バックエンド・システムで必要となる帯域幅や処理能力（ひいてはコスト）が削減されます。一元的に判断が行われることとは対照的に、自動化された決定はそれらのデータソースやそれに基づいて行動する必要があるシステムに近付られるため、結果として、より即応性に優れたシステムになります。

ストレージ：データ、特に実世界のセンサーから取得したデータには、さまざまな形式やサイズのものがあります。構造化されたリレーショナル・データから構造化されていないランダムなデータまで、IoT システムで取得するすべての情報について、収集、保管、取得、分析によって全価値を引き出す必要があります。ローカル・デバイスやゲートウェイ上の Oracle Berkeley DB 内のメモリにあるデータでも、アーキテクチャのミドル層やパブリック・クラウドの NoSQL データベース内のデータでも、データセンターやプライベート・クラウド内のリレーショナル・データベース内のデータでも、IoT システムのデータは各層で適切かつセキュアに保管する必要があります。

IoT プラットフォームでは、データ構造と取得のニーズに応じて、データを適切な場所に保管できます。迅速にデータを取得する必要性とセキュリティやデータ整合性の維持とのバランスを図ることは、IoT プラットフォームの役割であり、アプリケーションの関心事ではありません。

さらに深く

「Internet of Things」の約束を実現することは、モバイル・デバイスで Bluetooth や Wi-Fi をオンにする程度の単純なことに思えるかもしれません。しかし、**すべてが接続された未来の世界を最大限に活かすためには、もっと深い部分で統合する必要があります。**

ビッグ・データと分析：デバイスから収集され、ともすれば大量となるデータの分析によって、多くの隠れた価値が明らかになる可能性があります。一般的に、リアルタイム分析と履歴分析の 2 種類の分析があります。

リアルタイム分析では、デバイスからの収集時にデータを分析し、場合によっては別のデータと組み合わせて動的な判断を行います。できる限りデバイスに近い場所で分析が実行された場合に、最良の結果が得られる傾向にあります。一方、履歴分析では、長期間（数年間に及ぶこともある）取得したデータを使用して、アプリケーションとユーザー固有の傾向に相関関係があるかを調査します。履歴データ・セットは、時間の経過によって膨大なサイズになるため、分析は通常、データセンター内部のできる限りデータ・ストアに近い場所で実行されます。

ビッグ・データとは、非常に大量かつ複雑なために、従来のデータベース管理ツールやアルゴリズムでは処理が困難なデータ・セットの集合を示す用語です。分析ソリューションは、このような大規模なデータ・セットの保管、索引付け、検索、視覚化、およびデータ・セットからの推論や結論の導出という難問を解決します。IoT プラットフォームは、構造化データと非構造化

データを統合し、データに対してマップ/リデュース操作を実行し、Hadoop 分散ファイル・システム（HDFS）などのさまざまなファイル・システムをサポートする必要があります。

前述のイベント処理に関する説明のとおり、データの収集や整理を行う場所と物理的に近い周辺部で分析を実行することで、応答時間の短縮が可能な、より階層的なシステムを構築できる場合があります。

IoT の具現例

組込み、IoT というシナリオにおける Java の例として、512MB の Raspberry Pi を使用して家庭用ゲートウェイ・サーバーの基礎を構築します。このゲートウェイは、家庭において、電気機器を遠隔操作したり、食器洗い機などの電化製品をプログラミングし、電気使用料の単価（ワット時あたり）が特定のしきい値よりも低い場合に限り電源を入れるなどの目的で使用されることが想定できます。このゲートウェイを、電力会社のゲートウェイであるスマート・メーターと統合することで、消費者が電力コストの低減によるメリットを得て、電力会社は電力需要をより効率的に管理できるという意味で、すべての人にとって有益な真の IoT ソリューションを構築できます。

より現実的なソリューションを構築するために、ブラウザからアクセスして制御できるように、アプリケーションを Java サーブレットとして配置します。そうすることで、セキュリティ上の課題はあるものの、どこにいてもデスクトップ、ラップトップ、またはスマート・デバイスから家庭用ゲートウェイにアクセスできます。

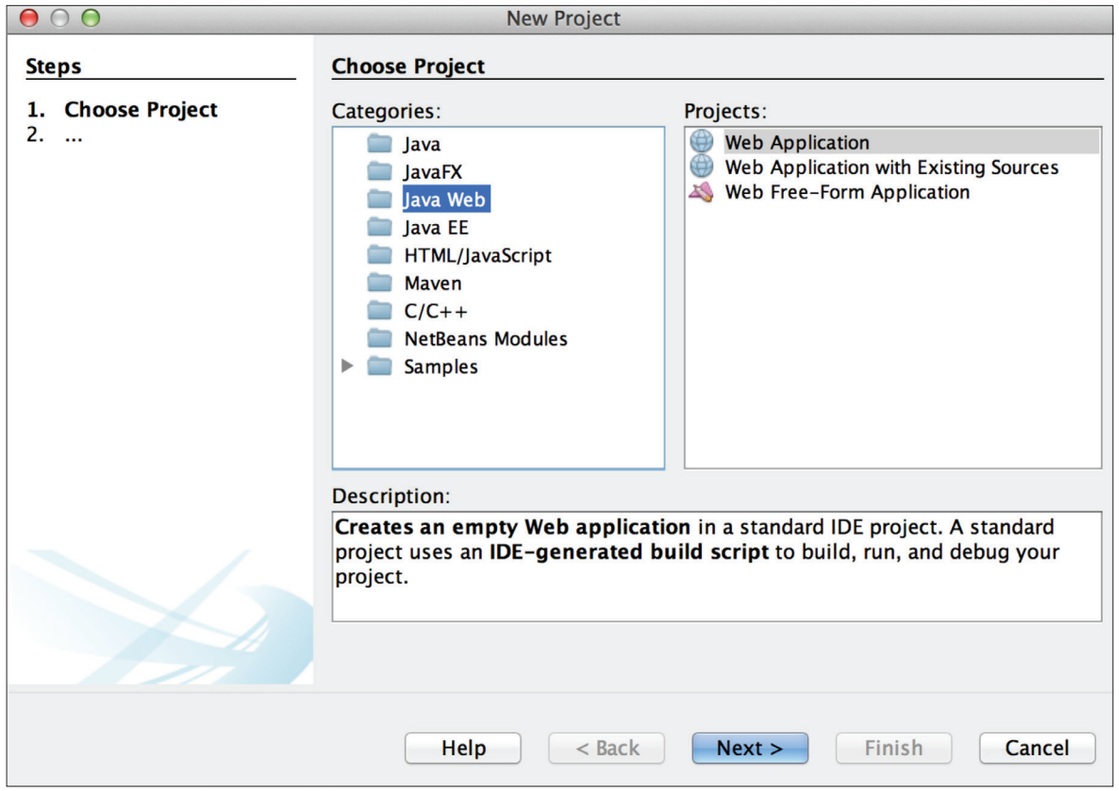

図 2

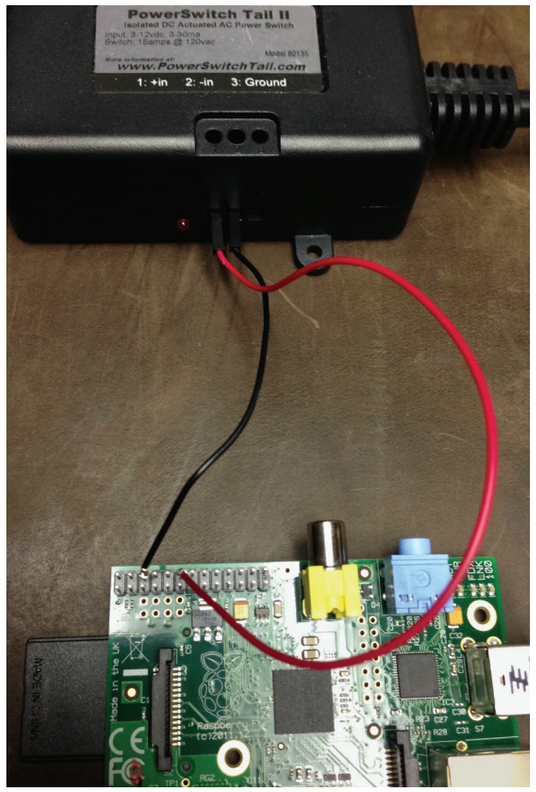

図 3

注：本記事で紹介するサンプルのソース・コードはこちらからダウンロードできます。

少し一服：組込みの環境においては、Java ME と Java SE から選択できます。ターゲット・デバイスの機能やアプリケーション、その他の要因から判断します。たとえば、Java ME では RAM などの必要なリソースが少なくなります。しかし、本記事の例では、Oracle Java SE Embedded を選択します。わずか 32MB のメモリで実行できる Oracle Java SE Embedded は、標準的な Java の要件と組込み開発の要件との絶妙なバランスをとっています。さらに、このサンプルの要件である、Java サーブレットを搭載した組込み Web サーバーを実行するという要件を満たしています。

まずは、Oracle Java SE Embedded をダウンロードしてインストールします。次のコマンドを使用して、ダウンロードした .tar ファイルを Raspberry Pi のホーム・ディレクトリに解凍します。

```
~ $ tar xvf <filename>.tar
```

~/ejre1.7.0_40/bin/java –version を実行してインストールをテストできます（ディレクトリ名は実際のインストール・ディレクトリの名前に置き換えてください）。

次に、サーブレット・アプリケーションを実行するためには、Apache Tomcat、IBM の WebSphere Liberty Profile、Oracle Java Embedded Suite などの、組込みの環境で実行するように設計されたサーバーを使用できます。サンプルのサーブレットのコードは標準の Servlet 仕様に準拠しているため、各サーバーで同様に動作します。ただし、配置方法のみがやや異なります。本記事では、コード自体について説明します。

まず、NetBeans で「Java Web」タイプの新規アプリケーションを作成します（図 2）。

筆者が試した際には、GlassFish をターゲット・ホストとして選択し、Raspberry Pi でサーブレットをホストするサーバーとして Oracle Java Embedded Suite を使用しました。この構成によって、ローカルで開発し Raspberry Pi と同様の配

置環境を使用してデバッグすることが簡単に実現できました。

サーブレットには Gateway という名前を付けました。このサーブレットは、Pi4J ライブラリを使用して、Java から Raspberry Pi の汎用入出力（GPIO）ピンに容易にアクセスできます。**リスト 1** に示すコードでは、Raspberry Pi の Pin 1 を表す変数を定義しています。

Pin 1 を「high」に設定することで、接続するデバイスに 3.3 ボルトが出力されます。サンプルでは、このピンを PowerSwitch Tail デバイスに接続します。PowerSwitch Tail デバイスは低電力の接続を受け入れ、その電力により内部の AC リレー・スイッチのオン / オフを切り替えることで、接続された電気機器（照明など）を制御します（図 3）。

PowerSwitch Tail の一方の先端を AC 電源に差し込みます。また、もう一方の先端に差し込まれた AC 電源機器が制御されます。サーブレットの残りのコードでは、Pin 1 の high/low（および接続されている機器の電源のオン / オフ）を、URL 経由で切り替えているだけです（**リスト 2**）。

筆者によるサーブレットのテストでは、Raspberry Pi 上で稼働するサーブレット・エンジンにこのサーブレットを配置し、ブラウザから「http://<Raspberry Pi の IP アドレス>/SmartGrid/Gateway?power=on」という URL を入力して Pin 1 を「high」に設定しました。

PowerSwitch Tail を Raspberry Pi に適切に接続し、この PowerSwitch Tail を AC 電源に差し込み、さらにデスク照明などの機器を PowerSwitch Tail のもう一方の先端に差し込んだ後に前述の操作を行うことで、リモートから機器の電源を入れることができます。電源を切るためには、単純に power パラメータの値を off に変更します（http://<Raspberry Pi の IP アドレス>/SmartGrid/Gateway?power=off と入力）。

まとめ

優れた機能と万能性があり、Raspberry Pi などの小さな組込みデバイス上で稼働でき、広範なツールやライブラリを利用できる Java のおかげで、開発者はまったく新しいスキル・セットを身に付けなくても、IoT の世界に足を踏み入れることができます。複数のプログラミング言語に熟達することが求められるクロスコンパイラやその他の特殊なツールを利用して、独自の一連のツールを構築する時代は終わりました。Java は、現在および未来の新しい IoT プロジェクトで開発者を成功に導く準備をすでに整えています。今日の開発者は、1 つのツール・セット、1 つの言語、1 つの統合された高性能 Java 仮想マシン（JVM）のセットを利用することで、新たな可能性を得ることができます。**</article>**

リスト1　リスト2

```
import com.pi4j.io.gpio.GpioController;
...

@WebServlet(urlPatterns = {"/Gateway"})
public class Gateway extends HttpServlet {
   private static final long serialVersionUID = 1L;

   final GpioController gpio =
         GpioFactory.getInstance();

   final GpioPinDigitalOutput pin =
         gpio.provisionDigitalOutputPin( RaspiPin.GPIO_01,
                                         "SmartGrid",
                                         PinState.LOW);
   ...
```

すべてのリストのテキストをダウンロード

LEARN MORE

- Oracle Java Embedded