Vaadin を使用したフルスタック Java の Web 開発

100 パーセント Java による Web アプリに HTML5 と JavaScript の最新の進化を取り入れる

Sing Li
Consultant
Makawave

2015年 11月 26日

このチュートリアルでは、Java 開発者が Vaadin フレームワークを使用することで、最新型の Web 開発への近道をする方法を紹介します。Vaadin を使用すれば、魅力的なユーザー・エクスペリエンスを創り出すために、HTML5、CSS、JavaScript をマスターする必要はありません。Vaadin フレームワークが開発者に代わってこれらの技術を取り入れてくれるので、Java 開発者は使い慣れた IDE で 100 パーセント Java だけを使ってプログラミングすることができます。実際のサンプルで確かめてみましょう。

Web アプリで魅力的な対話型のユーザー・エクスペリエンスを創り出すために、Java 開発者が最新のブラウザーの基礎となっている JavaScript、CSS、HTML5 といった Java 以外の技術を学ばなければならないことはよくあります。HTML5/JavaScript のフロントエンドと Java のバックエンドとの間で (通常は Ajax を使用して) インターフェースを取るには、保守するのが難しい複雑なアセンブリーが必要となります。しかし今は、オープンソースの Vaadin フレームワークのおかげで、Java 開発者は Java だけを使用したプログラミング環境で最新型の Web アプリのフルスタックの制御をできるようになっています。

Vaadin は成熟したフレームワークとして、よく使われている Java 開発環境のすべてをサポートしています (サポートしている開発環境には、Eclipse、IntelliJ、NetBeans、Maven などがありますが、このチュートリアルのサンプルでは Eclipse を使用します)。さらに、コミュニティーが貢献したものと市販されているものを含め、Vaadin には数百もの UI コンポーネントがあります。これらのコンポーネントは最近のあらゆるブラウザーで動作するように作成および保守されていて、HTML5、CSS、JavaScript の最新の機能を使用しています。

Vaadin ではサーバー・サイドの UI を作成するサポートをしていることから、Vaadin を使用すれば、再利用可能なコンポーネントを Java 技術で作成してデプロイすることができます。このフレームワークは、クライアント・サイドの UI を作成および管理する複雑さにも対処します。Java 開発者にお馴染みの中間層とバックエンドの Java 技術 (EJB (Enterprise Java Beans) 技術や、JPA (Java Persistence API)、Spring など) は、すべて簡単に統合することができます。Vaadin はサーバー・セントリックな性質を持つフレームワークであるため、IBM Bluemix をはじめと

する最近のクラウド・ホスト型ソリューションのすべてに対応します。したがって、極めてインタラクティブな Web アプリをクラウドにそのままデプロイすることができるのです。Vaadin は、JVM で駆動されているエコシステムに含まれるその他多くのプログラミング言語 (Python (Jython)、Groovy、Scala など) にも対応します。

Vaadin のスキルを活用してください

最大のハッカーソンで IBM と Vaadin を結び付けるべく、このチュートリアルと Bluemix ベースの関連チュートリアル「Vaadin を使用してクラウド内でフルスタック Java のアプリを開発する」で身に付けたスキルを活用して素晴らしいアプリを作成し、賞を獲得してください。このハッカーソンでの挑戦は、2015年 10月 15日から 11月 30日まで開催されています。このリンク先から今すぐ参加してください。

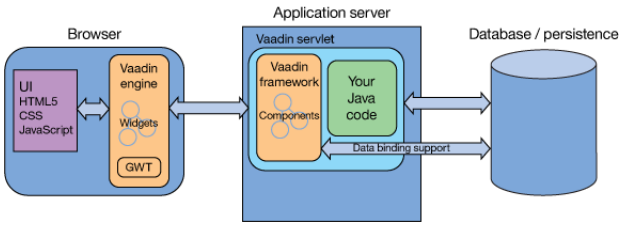
このチュートリアルでは Vaadin を紹介し、そのアーキテクチャーと仕組みを説明した上で、Eclipse 環境での Vaadin 開発に親しむための実践的なサンプルをいくつか記載します。そして最後の Jython サンプルでは、JVM でサポートしている多くのプログラミング言語に対応する Vaadin の柔軟性に注目します。

Vaadin の内部構造

Vaadin は、アーキテクチャーに関しては AWT、Swing、および SWT と同様です。そのまますぐに使えるサーバー・サイドのコンポーネントをインスタンス化して関連付けると、このフレームワークが自動的に UI 要素を生成してデプロイします。Vaadin と従来の Java ツールキットとの間の主な違いは、Vaadin では Java (またはサポート OS の特定のグラフィカル API) によってレンダリングされるデスクトップ GUI コンポーネントではなく、HTML5/CSS/JavaScript コンポーネントが Ajax を介してサーバーと通信し、このフレームワークによって自動的に管理されるという点です。

図 1 に、Vaadin のアーキテクチャーを示します。

図 1. Vaadin のアーキテクチャー



クライアント・サイドのプログラミング・モデル

このチュートリアルでは、より広範に採用されている Vaadin のサーバー・サイドのプログラミング・モデルにフォーカスしますが、それほど知名度が高くないクライアント・サイドのプログラミング・モデルも Vaadin はサポートしています。ブラウザー内で実行されるコードでウィジェットにアクセスすることや、リモートからサーバー・サイドのメソッドを呼び出すこともできます。ただし、それにはまず、Java から JavaScript にコードをコンパイルする必要があります。

図 1 を見ると、Vaadin では UI レンダリング・テクノロジー・スタックの複雑さを、開発者から完全に隠していることがわかります。Vaadin 開発者は、あらかじめ作成されたコンポーネントを Java サーブレット(`VaadinServlet`) 内で Java を使って関連付けて、さまざまなインタラクションに対応するイベント・ハンドラーを組み込んだ UI を作成すればよいのです。後は Vaadin がすべて

引き受けてくれます。クライアント・サイドの Vaadin エンジン (HTML5/CSS/JavaScript で作成された、Google Web Toolkit をベースとするエンジン) とサーバー・サイドのフレームワークが連携して、開発者に代わって UI の「実現」(生成) および管理をサポートします。つまり、ブラウザー内のクライアント・サイド UI ウィジェットのネットワークによって、サーバー・サイドの UI コンポーネントのネットワークが「実現」されるのです。

サーバー・サイドの UI コンポーネントは、Vaadin におけるデータ・バインディングをサポートできることから、バックオフィス・データベースと連動させるのは簡単です。Java アプリのコードでは引き続き、Java EE (Java Platform, Enterprise Edition) や Web サービスを含め、サーバーが提供するあらゆる機能を利用することができます。

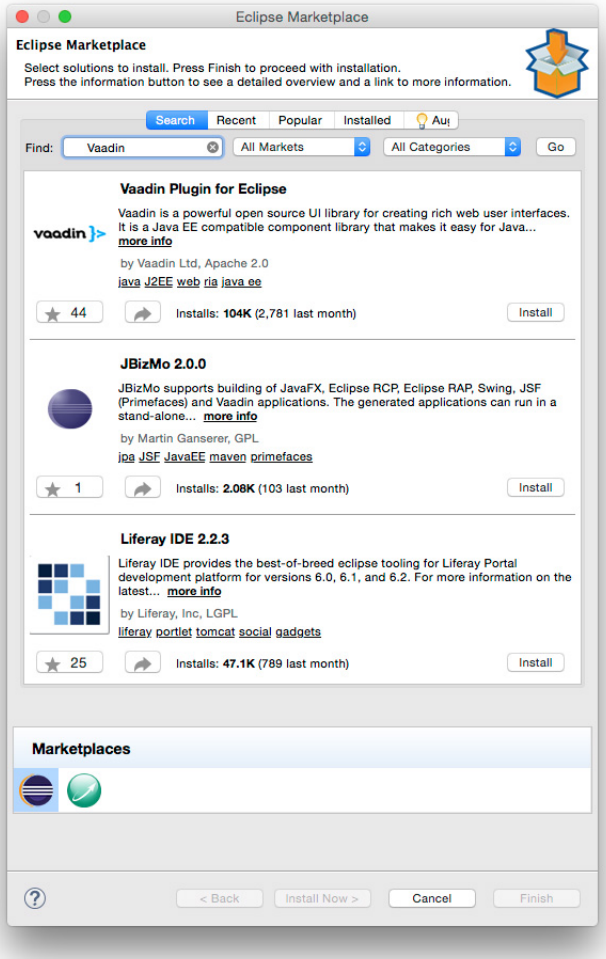
Vaadin を使用する場合、業界のベスト・プラクティスに従って、プレゼンテーションがビジネス・ロジックと切り離されます。レイアウトの作成は、Vaadin の HTML デザイン・ドキュメントを使って宣言によって行うことも、Java でプログラミングして行うこともできます。オプションで CSS または Sass のテーマを使用すれば、アプリの外観をさらにカスタマイズして調整することができます。

Vaadin を導入する

Vaadin を構成するすべての要素を理解したので、このフレームワークを使って何かの作成に取り掛かる準備はほぼできています。この記事のサンプルに取り組むには、以下のものをインストールする必要があります。

- JDK (Java Development Kit) 1.7 またはそれ以降のバージョン (サンプル・コードの開発で使用したのは 1.7.0_60-b19 です)。
- Eclipse EE エディション (Luna 以降のバージョン)。
- Tomcat 8。Eclipse の中で Tomcat 8 サーバーがまだ構成されていない場合は、このリンク先の手順に従って構成してください。
- Vaadin の Eclipse 用プラグイン。Eclipse マーケットプレースで Vaadin Plugin for Eclipse を見つけてください (図 2 を参照)。

図 2. Eclipse マーケットプレースの Vaadin Plugin for Eclipse



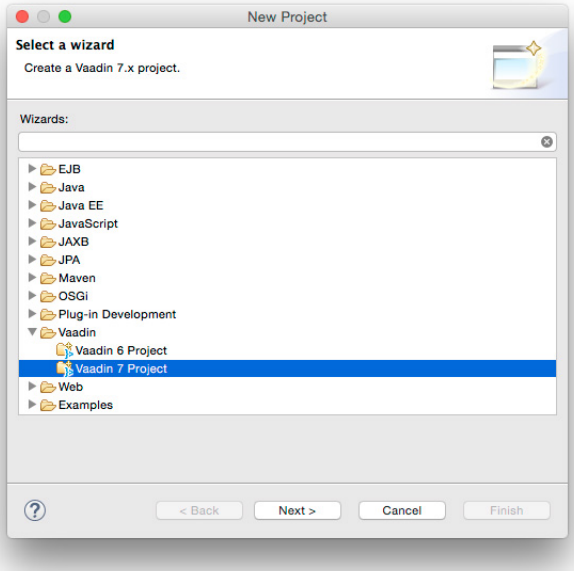
このチュートリアルの後のほうで取り上げる宣言型の UI サンプルでは、Vaadin Designer を利用します。デザインを保存するには、Vaadin Designer の試用版ライセンスが必要です。Vaadin の Eclipse 用プラグインをインストールした環境に、この UI デザイナーのベータ版を組み込むこともできます。その場合は、使用条件に同意して、Eclipse を再起動してください。

初めての Vaadin アプリを作成して実行する

Eclipse で以下の手順に従って、Vaadin スターター・アプリを作成してください。

1. 「File (ファイル)」 > 「New (新規)」 > 「Other (その他)」 > 「Vaadin」 > 「Vaadin 7 Project」の順に選択します (図 3 を参照)。

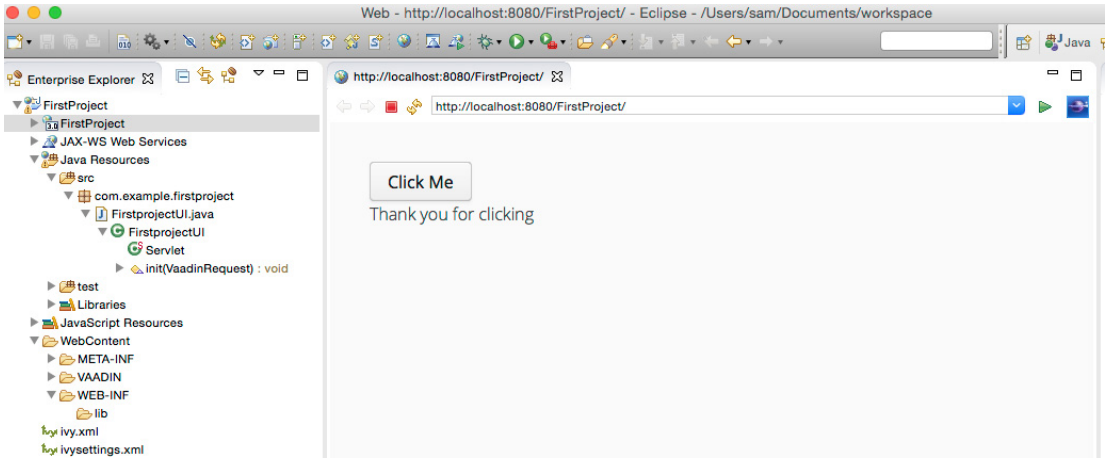
図 3. Eclipse でVaadin プロジェクトを作成する



2. プロジェクトに「`FirstProject`」という名前を付けてから、「Project (プロジェクト)」 > 「Build Project (プロジェクトのビルド)」の順に選択します。JDK 1.7 を使用している場合は、Java 1.8 ではなく Java 1.7 を組み込むように Eclipse プロジェクトの設定を調整する必要があります。
3. 作成したプロジェクトを「Enterprise Explorer (エンタープライズ・エクスプローラー)」ペインで強調表示 (選択) し、右クリックして表示されるメニューで「Run As (実行)」 > 「Run on server (サーバーで実行)」の順に選択し、Tomcat 8 サーバーを選択してアプリを実行します。

プロジェクトを作成すると、`VerticalLayout` と `Button` という 2 つの Vaadin コンポーネントが含まれたアプリが生成されます。このアプリが動いているところを図 4 に示します。上記の UI に示されているボタンをクリックするたびに、Vaadin の `Label` コンポーネントが新しく作成されます。この単純なアプリは、Vaadin 開発の基礎を学ぶには申し分ないアプリです。

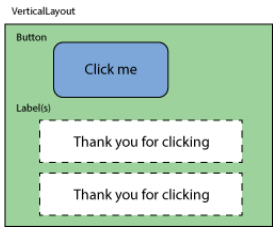
図 4. ウィザードによって生成された Vaadin スターター・アプリ



コンポーネントの構成

図 5 に、コンポーネントの構成を示します。`VerticalLayout` の中に、`Button` が含まれています。「Click Me (ここをクリック)」ボタンをクリックして、1 つまたは複数の `Label` を作成して VerticalLayout に追加することができます。

図 5. コンポーネントの構成



リスト 1 に、図 5 のコンポーネント同士を結び付けるコードを記載します。

リスト 1. 生成されたスケルトン・アプリのコード

```
public class FirstprojectUI extends UI {

@WebServlet(value = "/*", asyncSupported = true)
@VaadinServletConfiguration(productionMode = false, ui = FirstprojectUI.class)
public static class Servlet extends VaadinServlet {
 }

protected void init(VaadinRequest request) {
   final VerticalLayout layout = new VerticalLayout();
   layout.setMargin(true);
   setContent(layout);

   Button button = new Button("Click Me");
   button.addClickListener(new Button.ClickListener() {
      public void buttonClick(ClickEvent event) {
         layout.addComponent(new Label("Thank you for clicking"));
      }
   });
   layout.addComponent(button);
}
```

抽象クラス `com.vaadin.ui.UI` を継承する 1 つまたは複数のクラスの中で UI を作成します。UI が作成される場所は、`init()` メソッドです。`Servlet` クラス (`com.vaadin.server.VaadinServlet` から派生) には `@WebServlet` アノテーションが付けられており、作成する UI を提供および管理することになるのはこのクラスであることに注意してください。

Vaadin でのイベント処理

リスト 1 では、`button.addClickListener()` がボタン・クリック用のコールバックを追加しています。この場合、ボタンのクリック・リスナーは、「Thank you for clicking (クリックしてくれてありがとう)」というメッセージを示す `Label` コンポーネントを新規に作成して、その新しいコンポーネントを `VerticalLayout` コンポーネントに追加しているだけです。これにより、「Thank you for clicking」という新しい行のそれぞれが、既存のラベルの下に表示されます。

次は、さらに踏み込んだサンプルを取り上げます。このサンプルにはより複雑な Vaadin コンポーネントが必要になってきますが、その動作は基本的に最初のサンプルと同じです。

より複雑なコンポーネントを作成する

次のサンプルのために、サンプル・コード (「ダウンロード」を参照) から ArticleViewer.zip プロジェクト・ファイルをロードしてください。図 6 に、このサンプルで使用する Vaadin コンポーネントの構成を示します。

図 6. article-viewer アプリの Vaadin コンポーネントの構成

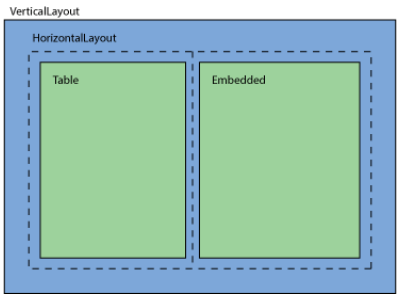
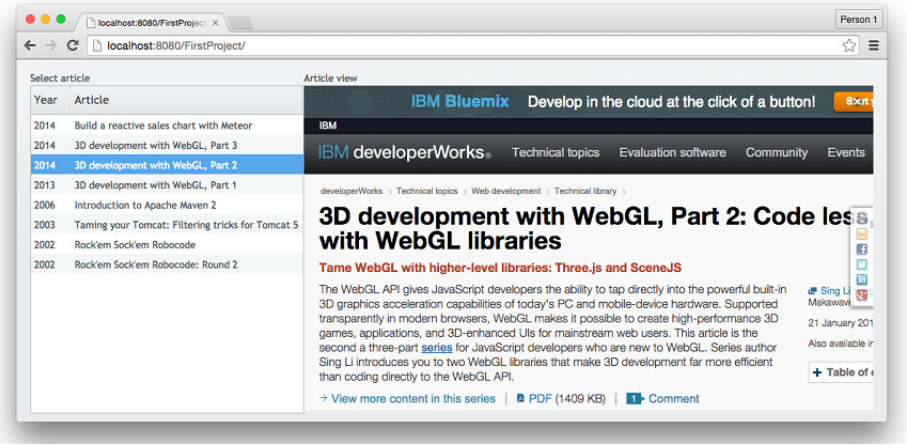


図 6 では `VerticalLayout` コンポーネントに `HorizontalLayout` コンポーネントが含まれており、この `HorizontalLayout` コンポーネントの内部には、左側に `Table` コンポーネント、右側に `Embedded` コンポーネントがあります。

図 7 に、article-viewer アプリの UI を示します。画面左側には、このアプリを構成する Vaadin Table コンポーネント内に記事のタイトルと公開年のリストが表示されています。ユーザーがこのリストから任意の記事を選択すると、その記事が取得されて、画面右側の Vaadin `Embedded` (ブラウザー) コンポーネント内に表示されます。

図 7. article-viewer アプリが動作しているところ



このコードは `ArticleViewerUI` クラスの中にあります。コードの構造や、`VaadinServlet` によってコードが提供されている点、さらにはクラスの `init()` メソッドの中に UI を作成するコードがある点 (リスト 2 を参照) は、最初のサンプルと同様です。

リスト 2. article-viewer アプリのコード

```
@Override
protected void init(VaadinRequest request) {
   final VerticalLayout layout = new VerticalLayout();

    layout.setMargin(true);
   setContent(layout);

   HorizontalLayout horiz = new HorizontalLayout();
   horiz.addStyleName("outlined");

   Table table = new Table("Select article");

   table.addContainerProperty("Year", String.class, null);
   table.addContainerProperty("Article", String.class, null);

   int i = 1;
   for (Object[] row: listdata ) {
      table.addItem(getRowData(row), i++);

   }
   table.setSelectable(true);
   table.setImmediate(true);

   horiz.setSizeFull();      horiz.addComponent(table);

   final Embedded e = new Embedded("Article view", new ExternalResource(
          getURL(listdata[0])));
     e.setAlternateText("Article View");
     e.setType(Embedded.TYPE_BROWSER);
     e.setSizeFull();
     horiz.addComponent(e);
 horiz.setExpandRatio(e, 1);

   table.addItemClickListener(new ItemClickListener(){

      @Override
      public void itemClick(ItemClickEvent event) {
         e.setSource(new ExternalResource(getURL(listdata
```

```
            [Integer.parseInt(event.getItemId().toString()) - 1]));

      }

   });

   layout.setSizeFull();
     layout.addComponent(horiz);
     layout.setExpandRatio(horiz, 1);
}
```

デモのための簡略化

サンプル・コードを簡略化するために、データの設定と初期データに関する詳細は同じ UI クラス内に入れました。実際には、リファクタリングしてこの 2 つを別々のクラスに分けるのが賢明です。

リスト 2 のコードでは、`ItemClickEvent` リスナーがハンドラーとして行の選択を処理します。このハンドラーが、クリックされた記事に関連付けられている URL を見つけて、`Embedded` コンポーネントの `source` プロパティーを設定することで、記事が表示されることになります。

リスト 2 では、`listdata` 要素の配列を繰り返し処理することによって、テーブルにデータが取り込まれることにも注意してください。これにより、コードは短くシンプルになっています。

Vaadin でのデータ・バインディング

Vaadin `Table` 要素は、データ・バインディングもサポートします。サンプル・コードに含まれる ArticleViewerDatabinding.zip プロジェクト・ファイルをインポートすると、`Table` 要素との適切なデータ・バインディングを実行する、`ArticleViewer` の機能強化版が得られます。

元の ArticleViewer プロジェクトと比較すると、以下の違いに気付くはずです。

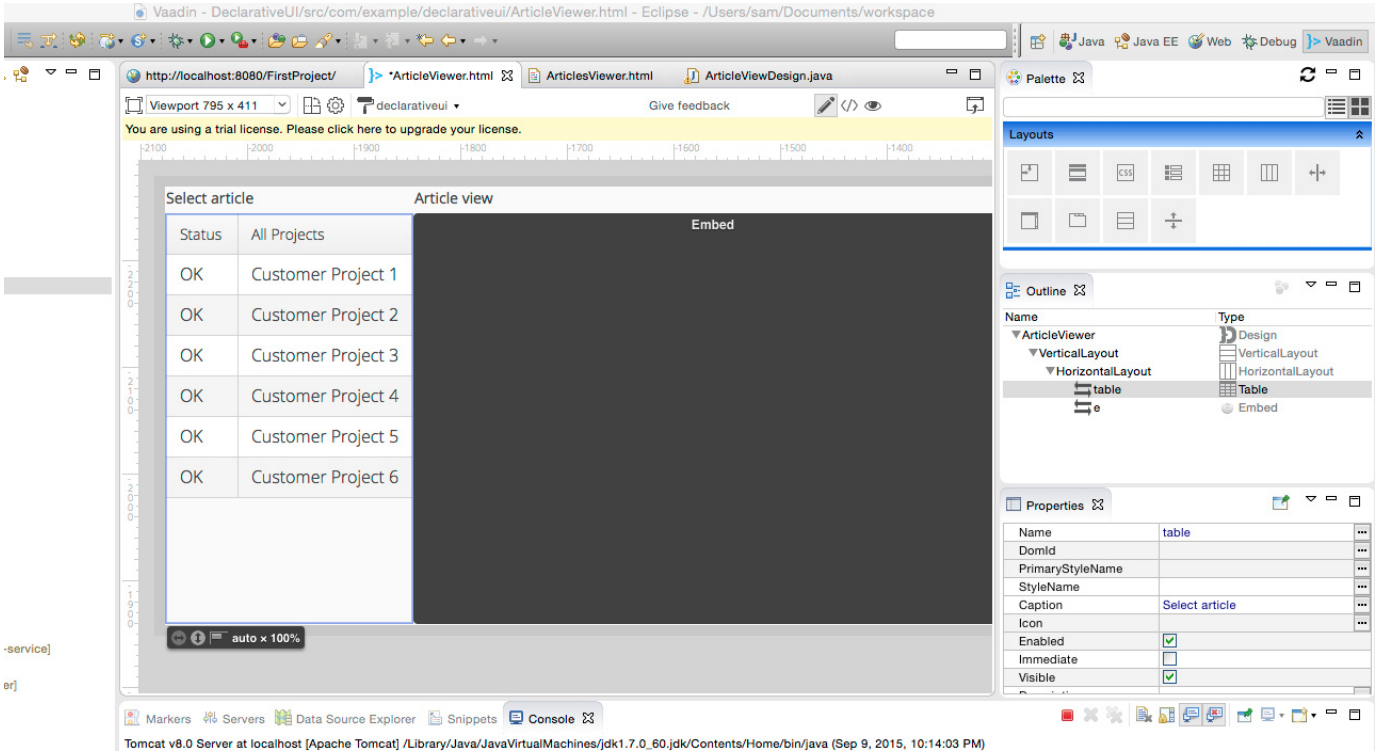
- `Article` は、単一の記事のデータを格納する POJO (Plain Old Java Object) クラスです。
- `ArticleService` は、静的データ・ソースを提供するサービス・クラスです。本番コードでは、このサービスはデータベースからデータをフェッチすることになります。
- キーボード・ナビゲーションをサポートするために、`Table` 要素では `ClickListener` の代わりに `ValueChangeListener` を使用しています。
- `BeanItemContainer` を使用して、`Article` POJO を直接 `Table` 要素にバインドしています。

Vaadin の宣言型 UI

Java プログラミング言語で 1 行ごとに UI を作成していくのではなく、特定の目的のためにフォーマット設定された HTML Vaadin デザイン・ドキュメントの中で、すべてのコンポーネントとそれぞれの包含関係を指定することができます。Vaadin はこの HTML ドキュメントを読み取って、UI 全体を一気に生成することができます。

さらに、Java コードや HTML を 1 行も書かずに UI デザイン・ドキュメントを作成できるよう、ドラッグ・アンド・ドロップで操作できる Vaadin ビジュアル・デザイナー・ツールも用意されています。図 8 に、このデザイナーを実行しているところを示します。

図 8. ドラッグ・アンド・ドロップ操作対応の Vaadin ビジュアル UI デザイナー



UI デザイナーとアプリの開発者がそれぞれ独立して作業している職場では、この機能は特に重宝します。

次のサンプルによって作成される UI は、article-viewer アプリとまったく同じですが、このサンプルでは Java コードではなく宣言型の HTML デザイン・ドキュメントで UI を指定しています。

DeclarativeUI.zip プロジェクトを Eclipse にインポートして、ArticleViewer.html デザイン・ドキュメントを調べてください (リスト 3 を参照)。この Vaadin HTML デザイン・ドキュメントで、UI およびレイアウトが指定されます。以下のコードをリスト 2 のコードと比較して、類似点を確認してください。

リスト 3. Vaadin HTML デザイン・ドキュメント

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8" name="design-properties" ...
 </head>
 <body>
  <v-vertical-layout size-full="true">
     <v-horizontal-layout size-full="true">
        <v-table _id="table" caption="Select article"
           width-auto height-full>

        </v-table>
        <v-embedded _id='e' caption="Article view"
          source="http://www.ibm.com/"
          size-full :expand>

     </v-horizontal-layout>
  </v-vertical-layout>
```

```
 </body>
</html>
```

カスタム・コンポーネント

ArticleViewer.html デザイン・ドキュメントを読み取って、そこに指定された内容に基づいて UI を実現するコードは、カスタム・コンポーネントである ArticleViewDesign.java 内にあります。このクラスは、テーブルにデータを取り込み、イベント・ハンドラーを関連付ける場所でもあります。リスト 4 にその方法を記載します。

リスト 4. 宣言型の UI デザイン・ドキュメントを使用した処理およびこのドキュメントとの連携

```
public ArticleViewDesign() {
       Design.read("ArticleViewer.html", this);

      table.addContainerProperty("Year", String.class, null);
      table.addContainerProperty("Article",  String.class, null);

      int i  = 1;
      for (Object[] row: listdata ) {
         table.addItem(getRowData(row), i++);

      }
      table.setSelectable(true);
      table.setImmediate(true);
      e.setType(Embedded.TYPE_BROWSER);
      e.setSource(new ExternalResource(getURL(listdata[0])));
      table.addItemClickListener(new ItemClickListener(){

         @Override
         public void itemClick(ItemClickEvent event) {
            // TODO Auto-generated method stub
            System.out.println("ID is " + event.getItemId());
            e.setSource(new ExternalResource(getURL(listdata
                [Integer.parseInt(event.getItemId().toString()) - 1])));

         }

      });
   }
```

(他のサンプルとまったく同じように、`com.vaadin.ui.UI` を継承し、`VaadinServlet` を含んでいる) `DeclarativeUI` Java クラスによって、カスタム `ArticleView` コンポーネントがインスタンス化されて UI に組み込まれます。リスト 5 に DeclarativeUI.java を記載します。

リスト 5. カスタム・コンポーネントを統合する UI 作成クラス

```
public class DeclarativeUI extends UI {

   @WebServlet(value = "/*", asyncSupported = true)
   @VaadinServletConfiguration(productionMode = false, ui =
     DeclarativeUI.class)
   public static class Servlet extends VaadinServlet {
   }

   @Override
   protected void init(VaadinRequest request) {

      setContent(new ArticleViewDesign());

   }
}
```

Vaadin のテーマを使用してアプリの外観を即座に変更する

プログラムによるバージョンと宣言型のバージョンでは、アプリの外観に違いがあることに注目してください。これは、この 2 つのバージョンでは「テーマ」を使用しているためです。テーマとは、アプリの外観をカスタマイズする CSS または Sass のコードであり、他のアプリのコードとは独立して適用することができます。

Vaadin には 4 つの組み込みテーマが付属しており、使用するテーマをこれらの中で簡単に変えることができます。組み込まれているテーマは、valo、reindeer、chameleon、および runo の 4 つです。WebContent/themes/declarativeui/declarativeui.scss を調べてください。このファイルの終わりのほうに指定されているテーマの値を変更すると、アプリで使用するテーマが変更されます。

Vaadin の Add-ons ディレクトリーにアクセスすると、すぐに適用できる、実に多種多様な Vaadin のテーマが見つかります。

Python で Vaadin をプログラミングする

意図的に、Vaadin は Java 以外のプログラミング言語に対応するように設計されていて、JVM エコシステムのプログラミング言語 (Wikipedia の「List of JVM languages」を参照) の多くをサポートしています。中でも代表的なのは、Python (Jython を使用)、Groovy、Scala です。

このチュートリアルの最後のサンプルでは、Jython を使用して Python で article-viewer アプリを作成し直します。

このサンプルを正常にロードして実行するには、その前に、以下のツールもインストールしておく必要があります。

- Jython 2.7 またはそれ以降のバージョン。
- Jython との連動をテスト済みの PyDev 4.3.0 またはそれ以降のバージョン (Eclipse にインストールする方法に関する情報は、PyDev のドキュメントを参照)。
- Vaadin 6.8.16 (または最新の 6.x) JAR。この JAR ファイルは WebContent/WEB-INF/lib フォルダー内に配置する必要があります。

このサンプルの Eclipse プロジェクトは、JythonVaadin.zip アーカイブに含まれています (「ダウンロード」を参照)。このアーカイブを Ecipse にインポートして、サンプルを実行します。

コードの構造が Java 版と似ていることに注目してください。この Web アプリの web.xml ファイル (リスト 6 を参照) を調べると、通常の `VaadinServlet` ではなく、特殊化された Python インタープリター・サーブレッドがロードされることがわかります。

リスト 6. JythonVaadin アプリの web.xml 記述子

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" ...>
   <display-name>JythonVaadin</display-name>
   ...
   <servlet>
    <servlet-name>Jythonvaadin Application</servlet-name>
    <servlet-class>com.example.jythonvaadin.PythonServlet</servlet-class>
    <init-param>
        <description>Vaadin application class to start</description>
        <param-name>application</param-name>
        <param-value>PythonArticleViewer</param-value>
    </init-param>
  ...
```

`PythonServlet` クラス内で、Jython インタープリターがロードされて PythonArticleViewer.py ソース・ファイルが提供されます (リスト 7 を参照)。

リスト 7. Jython インタープリターをインスタンス化する PythonServlet

```
...
PythonInterpreter interpreter = new PythonInterpreter();
        interpreter.exec("from "+applicationModuleName+" import "+applicationClassName);
        PyObject pyObj = interpreter.get(applicationClassName).__call__();
        Application pyApp = (Application)pyObj.__tojava__(Application.class);
        applicationClass = pyApp.getClass();
...
```

リスト 8 に、Vaadin を使用して article-viewer の UI を作成する Python コードから一部を抜粋します。Python の知識があれば、このコードと Java 版との類似点がわかるはずです。

リスト 8. Python による article-viewer Web アプリ

```
from com.vaadin import Application
from com.vaadin.event import ItemClickEvent
from java.lang import String
from com.vaadin.terminal import ExternalResource
...
class PythonArticleViewer(Application, ItemClickEvent.ItemClickListener):
    def __init__(self):
        pass

    def init(self):
      ...
      mainWindow = Window("Python Article View")
       vertical = VerticalLayout()

        vertical.setMargin(True)
        horiz = HorizontalLayout()
        horiz.addStyleName("outlined")
        self.table = Table("Select article")
        self.table.addContainerProperty("Year", String().getClass(), None)
        self.table.addContainerProperty("Article", String().getClass(), None)
```

```
        for idx, row in enumerate(self.listdata):
            self.table.addItem(self.getData(row) , idx)

        ...
        vertical.setSizeFull()
        vertical.addComponent(horiz)
        vertical.setExpandRatio(horiz,1)

        mainWindow.addComponent(vertical)
        self.setMainWindow(mainWindow)

    def itemClick(self, event):
       ...
```

まとめ

フルスタック Java の開発ソリューションを提供する Vaadin は、Java Web 開発者の生産性を向上させます。Vaadin により、使い慣れた Java IDE からまったく離れることなく、HTML5、CSS、Ajax、JavaScript を含め、ブラウザーの革新における最新の進化を取り入れた UI を作成できるようになります。Vaadin の UI は、Java コードをプログラミングして作成することも、HTML デザイン・ドキュメント内の宣言によって作成することもできます。Vaadin は、Python をはじめ、JVM をサポートする他のプログラミング言語で作成されたコードもサポートします。さらに、利用できるアドオンを豊富に集めたライブラリーを使用することで、苦もなくアプリに機能を追加できるようになります。

Vaadin と標準的な Java EE 技術を使用して作成したアプリは、簡単にクラウドにデプロイすることができます。Web アプリ全体を IBM Bluemix クラウドに迅速にデプロイする方法については、私のチュートリアル「Vaadin を使用してクラウド内でフルスタック Java のアプリを開発する」を参照してください。

謝辞

このチュートリアルのレビューとデータ・バインディングのサンプル提供に協力してくださった、Vaadin Inc. の Marcus Hellberg 氏に感謝いたします。

ダウンロード

内容	ファイル名	サイズ
Code for first three examples	vaadin_classic_code1_0928.zip	82KB
Code for Jython example	vaadin_classic_code2.zip	20MB

著者について

Sing Li

Sing Li は、developerWorks サイトの開設以来、Web や Java に関するさまざまなトピックを取り上げて記事とチュートリアルを書いている developerWorks の著者です。組み込みシステムからスケーラブルなエンタープライズ・システムに至るまで、20 年を超えるシステム・エンジニアリングの経験があり、現在は再び Web 規模のモバイル対応マイクロサービスと「モノのインターネット」エコシステムに取り組んでいます。