JSF 2 の魅力: 後から追加する Ajax 複合コンポーネント

実装済みの複合コンポーネントに、ページ作成者が Ajax を追加できるようにする

David Geary
President
Clarity Training, Inc.

2010年 6月 01日

JSF (Java™Server Faces) 2 Expert Group のメンバー、David Geary が、JSF 2 技術の詳細を紹介する連載を続けます。今回の記事では、実装済みの複合コンポーネントにページ作成者が Ajax を追加できるようにする方法を説明し、強力ながらもまったくドキュメントがない JSF 2.0 のタグについても詳しく見ていきます。さまざまな目的で再利用できる Ajax 対応の icon コンポーネントを、25 行にも満たない XML で実装する方法を学んでください。

このシリーズの他の記事を見る

この連載について

連載「JSF 2 の魅力」では、David Geary が JSF 2 について紹介する 3 回からなる同名の連載記事の続編として、皆さんが JSF 2 フレームワークのスキルを開発して磨きをかけ、カンフー・マスター並みにこのフレームワークの達人になるためのお手伝いをします。この連載では JSF 2 フレームワークとそれを取り巻くエコシステムの詳細を探るとともに、このフレームワークの外部にも目を向け、CDI (Contexts and Dependency Injection) などの Java EE 技術を JSF に統合する方法についても紹介します。

連載「JSF 2 の魅力」の前回の記事では、Ajax を組み込んだ自動補完複合コンポーネントを実装する方法を説明しました。ページ作成者がこのコンポーネントを Facelets で使用すると、このコンポーネントが Ajax の詳細をすべて引き受けてくれます。このように、Ajax を組み込んでおくと非常に便利ですが、開発者が実装した複合コンポーネントに (おそらくだいぶ時間が経ってから) ページ作成者が Ajax を追加できるようにするのも便利です。そこで、この記事では、複合コンポーネントに後から Ajax を追加できるようにする方法を説明します。

「JSF 2 の魅力: 第 3 回 イベント処理、JavaScript、そして Ajax」で説明したように、JSF 2 の `<f:ajax>` タグを使えば、ページ作成者が JSF 2 の組み込みコンポーネントに後から Ajax を追加することができます。例えば、`<f:ajax>` を使用すると、以下のように簡単にサブミット・ボタンを Ajax のボタンに変えることができます。

```
<h:commandButton value="Click me">
<f:ajax>
</h:commandButton>
```

© Copyright IBM Corporation 2010

けれども `<f:ajax>` タグは、うまく扱わない限り、複合コンポーネントでは機能しません。複合コンポーネントは、単なるコンポーネントのコンテナーでしかないからです。

複合コンポーネントの例として、「JSF 2 の魅力: 第 2 回 テンプレート機能と複合コンポーネント」では、画像として表現された 1 つのリンクからなる単純な icon 複合コンポーネントを紹介しました。ユーザーがこのアイコンをクリックすると、アイコンの構成要素となっているリンクが対応するフォームをサブミットし、それによってアイコンのリンクと関連付けられたサーバー・サイドのアクション・リスナーがトリガーされます。アイコンを使用する方法は、以下のように簡単です。

```
<util:icon image="...">
  <f:actionListener for="link" type="...">
</util:icon>
```

`<f:ajax>` タグでサブミット・ボタンを Ajax ボタンに変えられるのなら、アイコンにも同じ方法を適用できるのではないかと考えるかもしれません。

```
<util:icon image="...">
<f:ajax>
  <f:actionListener for="link" type="...">
</util:icon>
```

しかし上記のコード・フラグメントは機能しません。それは、`<f:ajax>` タグが icon コンポーネントにアタッチされているからです。本当は何がしたいのかと言えば、このタグをアイコンの内側にあるリンクにアタッチしたいのです。

この例で必要なのは、Ajax の振る舞いをアイコン内部のリンクにアタッチするために使えるメカニズムです。もっと一般的に言うと、Ajax の振る舞いを、複合コンポーネントの内部にあるコンポーネントにアタッチできるようにするメカニズムです。そのメカニズム (Mojarra および Apache MyFaces に実装されていますが、JSF 2.0 にはまったくドキュメントがありません) が、この記事の焦点となります (注: MyFaces のサポートは、この記事を執筆している時点で追加されました)。このメカニズムがどのように機能するのかを説明する前に、説明で使用するための新しい icon 複合コンポーネントを作成します。

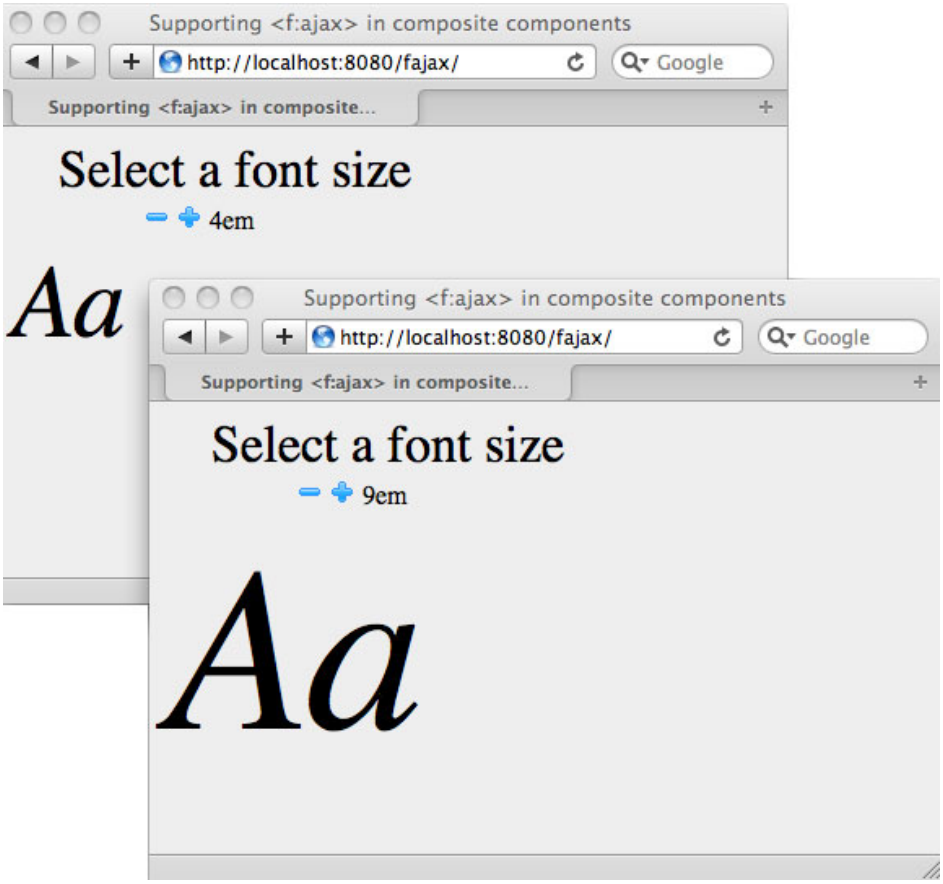
サンプル・コードの実行

この連載で使用するコードは、GlassFish や Resin などのエンタープライズ・コンテナー内で実行される JSF 2 をベースとしています。GlassFish でコードをインストールおよび実行する方法については、ステップバイステップのチュートリアルになっている「JSF 2 fu: Ajax components」を参照してください。また、この記事のサンプル・コードを入手するには、「ダウンロード」を参照してください。

再利用可能な icon コンポーネント

例えば、あなたは今、世界で最も素晴らしい仕事をしているとします。その仕事とは次世代 World of Warcraft のグラフィックス・エンジンの実装ということにしたいところですが、ここではその設定では無理があるので、図 1 に示すフォント・セレクターを実装していることにします。

図 1. フォントの選択



あなたの上司たちが、このようなセレクターを実装するには、どれくらいの時間がかかるのかを聞いてきました。上司たちの要望は、ユーザーがプラス (+) またはマイナス (-) のアイコンをクリックすると、プレビューに表示された 2 つの文字のフォント・サイズが変更されるというものです。要望のなかには、もちろん Ajax も入っています。Ajax によって、ページの残りの部分に支障なく、プレビューとアイコンの隣にあるサイズ表示を動的に更新するためです。

上司たちが望んでいるのは単純なフォント・セレクター・コンポーネントですが、彼らよりも知識のあるあなたは、実行時に画像とアクションで構成することも、完全に Ajax 化することも可能な汎用の icon コンポーネントを実装することにしました。そして実装した icon コンポーネントをフォント・プレビューで使用するという計画です。そうすれば、将来も役立つ有益な icon コンポーネントになります。

これから、この icon コンポーネントを 25 行にも満たない XML で実装する方法を紹介します。

サンプル・フォント・セレクター

複合コンポーネント: 基礎知識

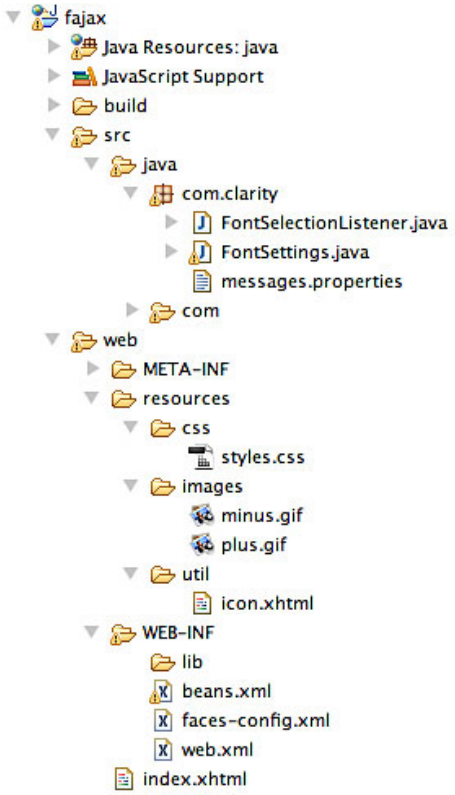
JSF 2 の複合コンポーネントの使い方や実装方法がよくわからない場合は、「JSF 2 の魅力: テンプレート機能と複合コンポーネント」でわかりやすく紹介しています。

サンプル・フォント・セレクターは、以下の 4 ファイルで構成されています。

- 図 1 のページ (index.xhtml で定義されています)
- icon コンポーネント (/resources/util/icon.xhtml に置かれています)
- リスナー (com.clarity.FontSelectionListener.java)
- Bean (com.clarity.FontSettings)

図 2 にディレクトリー構造を示します。

図 2. サンプル・フォント・セクレターのファイル



リスト 1 に、図 1 に示したページの Facelets (index.xhtml) を記載します。

リスト 1. Facelets (/web/index.xhtml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:corejsf="http://corejsf.com"
xmlns:util="http://java.sun.com/jsf/composite/util">

   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{msgs.windowTitle}</title>
   </h:head>

   <h:body>
      <h:outputStylesheet library="css" name="styles.css"/>
```

```
      <h:outputText value="#{msgs.fontSizeHeading}"
        style="padding-left: 30px; font-size: 2em;"/>

      <h:panelGrid columns="3" style="padding-left: 80px;">
        <util:icon id="minus" image="#{resource['images:minus.gif']}">
          <f:actionListener for="link" type="com.clarity.FontSelectionListener"/>
        </util:icon>

        <util:icon id="plus" image="#{resource['images:plus.gif']}">
          <f:actionListener for="link" type="com.clarity.FontSelectionListener"/>

        </util:icon>

        <h:outputText id="readout" value="#{fontSettings.size}em"/>
      </h:panelGrid>

      <h:outputText id="fontPreview" value="Aa"
                style="font-size: #{fontSettings.size}em; font-style: italic"/>

   </h:body>
</html>
```

リスト 1 の Facelets は icon コンポーネントの名前空間を宣言し、このコンポーネントをページ内で使用しています。これは JSF 2.0 複合コンポーネントの基本的な使い方であり、「JSF 2 の魅力: テンプレート機能と複合コンポーネント」で詳しく説明しています。

マイナスとプラスの両方のアイコンに、アイコンの `link` コンポーネントを対象としたアクション・リスナーが設定されていることに注目してください。ユーザーがどちらかのアイコンのリンクをクリックすると、JSF が該当するリスナー (リスト 2 を参照) をサーバー上で呼び出します。

リスト 2. リスナー (com/clarity/FontSelectionListener.java)

```
package com.clarity;

import javax.el.ELResolver;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.event.AbortProcessingException;
import javax.faces.event.ActionEvent;
import javax.faces.event.ActionListener;

public class FontSelectionListener implements ActionListener {
   @Override
   public void processAction(ActionEvent event)
         throws AbortProcessingException {
    FacesContext c = FacesContext.getCurrentInstance();
    ELResolver elResolver = c.getApplication().getELResolver();
    FontSettings fs = (FontSettings)
elResolver.getValue(c.getELContext(), null, "fontSettings");

    if (((UIComponent)event.getSource()).getClientId().startsWith("minus"))
fs.decrement();
    else
fs.increment();
   }
}
```

リスト 2 では、イベントをトリガーしたコンポーネントのクライアント ID が `minus` であるかどうかを確認します。この ID であれば、ユーザーはマイナスのアイコンをクリックしたことになるので、フォント・サイズを小さくします。そうでなければ、フォント・サイズを大きくします。

リスナーは、`fontSettings` 管理 Bean への参照を取得することに注目してください。その方法は、指定された名前を基に管理 Bean を見つける方法を把握している、式言語リゾルバーへの参照を取るというものです。リスト 3 に、`fontSettings` Bean を記載します。

リスト 3. fontSettings Bean (com/clarity/FontSettings.java)

```
package com.clarity;

import java.io.Serializable;

import javax.inject.Named;
import javax.enterprise.context.SessionScoped;

@Named
@SessionScoped
public class FontSettings implements Serializable {
   private static int INCREMENT = 1;
   private int size = 1;

   public int getSize() { return size; }
   public void setSize(int newValue) { size = newValue; }

   public void increment() { size += INCREMENT; }
   public void decrement() { size -= INCREMENT; }
}
```

icon 複合コンポーネントのコードを除き、アプリケーションのコードは以上の 3 つのリストにすべて示されています。次は、icon 複合コンポーネントに取り掛かります。

icon 複合コンポーネントの実装

アイコンには、次の 3 つの要件があります。

- 画像が構成可能であること
- ユーザーが画像をクリックしたときのアクションが構成可能であること
- アイコンが Ajax をサポートすること

リスト 4 は、最初の 2 つの要件を満たしています。

リスト 4. `<util:icon>` 複合コンポーネント、テイク 1 (/resources/util/icon.xhtml)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:composite="http://java.sun.com/jsf/composite">

    <composite:interface>
      <composite:attribute    name="image" required="true"/>
      <composite:actionSource name="link"  targets="#{cc.clientId}:iconForm:link"/>
    </composite:interface>

    <composite:implementation>
      <div id="#{cc.clientId}">
         <h:form id="iconForm">
            <h:commandLink id="link" immediate="true">
              <h:graphicImage value="#{cc.attrs.image}"/>
            </h:commandLink>
         </h:form>
```

```
      </div>
    </composite:implementation>
</html>
```

リスト 4 の icon コンポーネントが宣言しているのは、`image` 属性と `link` という名前の `actionSource` です。この `actionSource` は、リスト 1 の `<f:actionListener>` の `for` 属性の値として使用されています。この仕組みがよくわからない場合は、「JSF 2 の魅力: テンプレート機能と複合コンポーネント」を読んでください。リスト 4 と同じようなアイコン実装について概説しているこの記事では、アクション・ソースが複合コンポーネントと連動する方法を詳しく説明しています。

リスト 4 の icon コンポーネント実装では、ページ作成者がアイコンの外観と振る舞いを構成することはできても、Ajax の振る舞いをコンポーネントにアタッチすることはできません。この実装では目下、ユーザーが画像をクリックすると JSF はページ全体をサブミットするため、ページが返ってくると、ページごと再描画されます。

そこで次に説明するのが、ページ作成者が icon コンポーネントに Ajax を追加できるようにする方法です。

`<composite:clientBehavior>` による Ajax サポートの追加

ページ作成者が `<util:icon>` コンポーネントの内部にあるリンクに Ajax を追加できるようにするために用いるのは、`<composite:clientBehavior>` タグです (リスト 5 を参照)。

リスト 5. `<util:icon>` 複合コンポーネント、テイク 2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:composite="http://java.sun.com/jsf/composite">

    <composite:interface>
      <composite:attribute      name="image" required="true"/>
      <composite:actionSource   name="link"  targets="#{cc.clientId}:iconForm:link"/>
     <composite:clientBehavior name="click"
                               event="action"
                             targets="#{cc.clientId}:iconForm:link"/>
    </composite:interface>

    <composite:implementation>
      <div id="#{cc.clientId}">
         <h:form id="iconForm">
            <h:commandLink id="link" immediate="true">
              <h:graphicImage value="#{cc.attrs.image}"/>
            </h:commandLink>
         </h:form>
       </div>
    </composite:implementation>
</html>
```

ドキュメント化されていない <composite:clientBehavior>

`<composite:clientBehavior>` タグは、JSF 2.0 Expert Group のサブグループによって具体化されたタグです。このタグは JSF リファレンス実装 (Mojarra) に実装されたものの、JSF

2.0 に際してドキュメント化されませんでした。そのため、Javadoc でも、JSF 仕様でも、このタグについてはまったく言及されていません。

ドキュメント化されていないことは、仕様バグとして知られています (詳細については、「参考文献」のリンクを参照)。JSF Expert Group では、仕様と Javadoc にこのタグに関する説明を追加することにより、このバグを確実に修正する予定です。

`<composite:clientBehavior>` タグは、複合コンポーネントに含まれるコンポーネントの Ajax イベントを公開します。リスト 5 では、`click` という論理名でクライアントの振る舞いを宣言しました。この振る舞いに、実際のイベント、つまりアイコンのリンクによって起動される `action` イベントを関連付けます。以下に、`<composite:clientBehavior>` タグに有効な属性について要約します。

- **`name`**: ページ作成者が使用するイベントの名前です。
- **`default`**: `true` または `false` のいずれかに設定されます。`true` の場合のデフォルト・イベントは、name 属性で指定されたイベントです。
- **`event`**: イベントの実際の名前です。
- **`targets`**: スペースで区切られたコンポーネント ID のリストです。JSF は振る舞いの対象をこれらのコンポーネントに再設定します。

これで、ページ作成者は Ajax の振る舞いを icon コンポーンネントにアタッチできるようになります (リスト 6 を参照)。

リスト 6. Facelets、テイク 2

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:corejsf="http://corejsf.com"
      xmlns:util="http://java.sun.com/jsf/composite/util">

   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{msgs.windowTitle}</title>
   </h:head>

   <h:body>
      <h:outputStylesheet library="css" name="styles.css"/>

      <h:outputText value="#{msgs.fontSizeHeading}"
        style="padding-left: 30px; font-size: 2em;"/>

      <h:panelGrid columns="3" style="padding-left: 80px;">
        <util:icon id="minus" image="#{resource['images:minus.gif']}">
<f:ajax event="click" render=":readout :fontPreview"/>
          <f:actionListener for="link" type="com.clarity.FontSelectionListener"/>
        </util:icon>

        <util:icon id="plus" image="#{resource['images:plus.gif']}">
<f:ajax event="click" render=":readout :fontPreview"/>
          <f:actionListener for="link" type="com.clarity.FontSelectionListener"/>
        </util:icon>

        <h:outputText id="readout" value="#{fontSettings.size}em"/>
```

```
      </h:panelGrid>

      <h:outputText id="fontPreview" value="Aa"
                style="font-size: #{fontSettings.size}em; font-style: italic"/>

   </h:body>
</html>
```

複合コンポーネントの外部にあるコンポーネントのレンダリング

リスト 6 では、<f:ajax> タグの render 属性に readout コンポーネントと fontPreview コンポーネントを指定しています。この 2 つのコンポーネントの ID は、コロンで始まっていることに注意してください。

このコロンは、JSF にコンポーネント階層の先頭からコンポーネントを検索させるためのものです。こうしないと、JSF は Ajax がアタッチされるコンポーネントに最も近いネーミング・コンテナー (通常はフォーム) から検索することになります。このアイコンの場合、Ajax がアタッチされるコンポーネントはアイコンのリンクであり、それに最も近いフォームは、リンクが含まれるアイコンのフォームです。コロンを削除すると、JSF はアイコンのフォームから readout および fontPreview コンポーネントの検索を開始するため、コンポーネントを検出できません。したがって、エラーが発生します。

リスト 6 では、<f:ajax> タグを両方のアイコンに追加しています。ユーザーがアイコンのいずれかをクリックすると、JSF はサーバーに対する Ajax 呼び出しを行い、呼び出しから戻った時点で、readout および fontPreview コンポーネントをレンダリングします。

アイコンのインターフェースに含まれる <composite:clientBehavior> タグに default="true" 属性を含めるという選択肢もあります。そうすれば、ページ作成者が click イベントを指定する必要がなくなり、リスト 6 の <f:ajax> タグはさらに簡潔に、<f:ajax render=":readout :fontPreview"> となります。

まとめ

この記事では、ドキュメント化されてはいないタグではあるものの、このタグを使用することによって JSF 2 では簡単にページ作成者が複合コンポーネントに Ajax 機能を追加できるようにしていることを説明しました。この記事でまさに目にしたとおり、完全な HTTP リクエストを行う複合コンポーネントを、ページ作成者が Ajax リクエストを行うコンポーネントに変えることが可能です。そしてそのすべての仕組みは、複合コンポーネントのインターフェースでの 1 行の XML に凝縮されます。

連載「JSF 2 の魅力」の次回の記事では、複合コンポーネントの話題から離れ、JSF 2 で CDI (Contexts and Dependency Injection) を使用する方法について見ていきたいと思います。

ダウンロード

内容	ファイル名	サイズ
Sample code for this article	j-jsf2fu-0610-src.zip	11KB

著者について

David Geary

著者、講演者、コンサルタントとして活躍する David Geary は、Clarity Training, Inc. の社長です。彼はこの会社で、開発者に JSF および GWT (Google Web Toolkit) を使用した Web アプリケーションの実装を指導しています。JSTL 1.0 および JSF 1.0/2.0 Expert Group のメンバー、そして Sun の Web 開発者認定試験の共同制作者としての経験を持つ彼は、Apache Struts や Apache Shale などのオープンソース・プロジェクトにも貢献しています。彼の著書『グラフィック Java2 〈Vol.2〉 Swing 編』は Java 関連の本のなかでは史上に残るベスト・セラーの 1 つで、『Core JSF』(Cay Horstman との共著) は JSF 関連のベストセラー本となっています。コンファレンスやユーザー・グループで定期的に講演を行っている他、2003年以来 NFJS ツアーの常連で、これまでに 3 回 Java University の講師になり、JavaOne ロック・スターに 2 回選ばれました。

© Copyright IBM Corporation 2010
(www.ibm.com/legal/copytrade.shtml)
商標
(www.ibm.com/developerworks/jp/ibm/trademarks/)