



Struts を使用した Web アプリケーションの作成

目次

Struts を使用した Web アプリケーション の作成 1

概要: Struts を使用した Web アプリケーションの作 成	1
演習 1: Struts プロジェクトの作成	2
演習のチェックポイント	4
演習 2: Web ダイアグラム・ノードの作成	4
演習のチェックポイント	5
演習 3: フォーム Bean の作成	6
演習のチェックポイント	8
演習 4: Web ダイアグラムでのノードの接続	8
演習のチェックポイント	11
演習 5: フォーム Bean の編集	11
演習のチェックポイント	12

演習 6: アクション・マッピングの編集	12
演習のチェックポイント	14
演習 7: JavaServer Pages (JSP) ページへの要素の追 加	14
入力ページ	15
出力ページ	16
演習のチェックポイント	17
演習 8: エラー・メッセージの表示	17
演習のチェックポイント	18
演習 9: Struts アプリケーションのテスト	19
演習のチェックポイント	19
要約: Struts を使用した Web アプリケーションの作 成	20

Struts を使用した Web アプリケーションの作成

このチュートリアルでは、2 つの Web ページ、1 つのアクション・マッピング、および 1 つのフォーム Bean を備えたシンプルな Struts アプリケーションの作成方法について学習します。Web ダイアグラム・エディターを使用して、アプリケーションのロジック・フローの視覚化を行います。アプリケーションの作成後に、ランタイム・テスト環境のセットアップ方法およびアプリケーションのテスト方法を学習します。

学習目標

以下のタスクの実行方法を学習します。

- Struts プロジェクトの作成
- Web ダイアグラムの編集
- フォーム Bean の作成と編集
- 入力と出力の JavaServer Pages (JSP) ファイルの作成
- アクションおよびアクション・マッピングの作成
- 入力 JSP ページへの「ActionError」タグの追加
- Struts アプリケーションのテスト

60 分

関連情報



PDF バージョンを表示する

概要: Struts を使用した Web アプリケーションの作成

このチュートリアルでは、2 つの Web ページ、1 つのアクション・マッピング、および 1 つのフォーム Bean を備えたシンプルな Struts アプリケーションの作成方法について学習します。Web ダイアグラム・エディターを使用して、アプリケーションのロジック・フローの視覚化を行います。その結果、アプリケーションで、ユーザーが年、月、日付を入力できる入力用 Web ページが表示されます。入力された値が有効な場合、それに対応した曜日が計算され、出力 Web ページに表示されます。入力された値が無効の場合、入力用 Web ページにエラー・メッセージが表示されます。また、ランタイム・テスト環境のセットアップ方法およびアプリケーションのテスト方法についても学習します。

学習目標

以下のタスクの実行方法を学習します。

- Struts プロジェクトの作成
- Web ダイアグラムの編集
- フォーム Bean の作成と編集
- 入力と出力の JavaServer Pages (JSP) ファイルの作成
- アクションおよびアクション・マッピングの作成
- 入力 JSP ページへの「ActionError」タグの追加
- Struts アプリケーションのテスト

所要時間

このチュートリアルを終了するには、およそ **60 分間** かかります。 このチュートリアルに関連した他の概念を検討する場合、完了するのにさらに長い時間がかかります。

スキル・レベル

上級

対象読者

Web アプリケーション開発者および Web ユーザー・インターフェース設計担当者

システム要件

このチュートリアルを完了するには、以下のいずれかのサーバーがローカル・マシンで構成されている必要があります。

- WebSphere® Application Server 5.1 テスト環境またはリモート・サーバー
- WebSphere Application Server 6.x ランタイム・サーバー
- Apache Tomcat 5.x ランタイム・サーバー

前提条件

このチュートリアルを完了するには、以下の分野に精通している必要があります。

- サイト、ページ、ブラウザー、およびサーバーといった、Web の基本的な設計概念
- テーブル、ハイパーリンク、フォーム、およびイメージなどの Web ページ要素。
- Web ページ用に HTML コードを編集する方法
- Eclipse ベース製品のワークベンチのパーспекティブとビュー

演習 1: Struts プロジェクトの作成

Struts で使用可能な動的 Web プロジェクトを作成します。

Struts は、Java™ Web アプリケーションを作成するためのオープン・ソースのソフトウェア・フレームワークです。 Struts は、アプリケーション開発に *model-view-controller (MVC)* デザイン・パターンを使用します。MVC デザイン・パターンは、アプリケーションを 3 つのパーツに分割することで、複雑な Web アプリケーションを作成および保守できるようにします。

- 「モデル」はアプリケーション内のデータとビジネス・ロジックを保守します。このチュートリアルではデータベースを使用せず、ユーザーの入力するデータが唯一のデータであるため、このチュートリアルにはモデルが存在しないことになります。
- 「ビュー」はユーザーに対して情報を表示するインターフェースです。このチュートリアルでは、ビューは、ユーザーからデータを収集して、結果をユーザーに対して表示する、2 つの Web ページで構成されます。
- 「コントローラー」はアプリケーション内のイベントを処理します。このチュートリアルでは、コントローラーは、ユーザーが入力する情報を連動して処理するアクション・マッピングとフォーム Bean で構成されます。

ヒント: Struts テクノロジーについてさらに学習したい場合は、<http://struts.apache.org/> を参照してください。

Struts ベースの Web プロジェクトを新規作成する手順は、次のとおりです。

1. メニュー・バーで、「ファイル」→「新規」→「プロジェクト」を選択して、「新規プロジェクト」ウィンドウを開きます。
2. 「Web」→「動的 Web プロジェクト」を選択して、「次へ」をクリックして「新規動的 Web プロジェクト」ウィザードを開きます。
3. 「プロジェクト名」フィールドに「DayOfWeek」と入力します。
4. 「ターゲット・ランタイム」フィールドで、構成済みのランタイムを選択します。「次へ」をクリックします。

注: ランタイム環境を構成していない場合は、「新規」をクリックして「新規サーバー・ランタイム」ウィザードを開きます。このチュートリアルでは、以下のいずれかのサーバーを使用することができます。

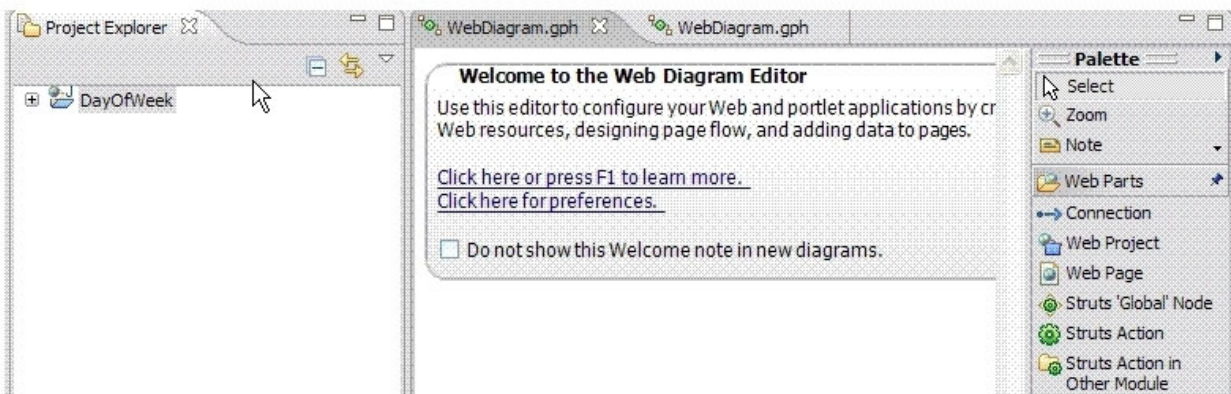
- WebSphere Application Server 5.1 テスト環境またはリモート・サーバー
- WebSphere Application Server 6.x ランタイム・サーバー
- Apache Tomcat 5.x ランタイム・サーバー

(Apache Tomcat サーバーの構成について詳しくは、<http://tomcat.apache.org/download-55.cgi> を参照してください。)

5. 「プロジェクト・ファセット」ページで、「Struts」をクリックしてから「次へ」をクリックします。
6. 「Web モジュール」ページで「次へ」をクリックします。
7. 「Struts 設定」ページで「終了」をクリックします。
8. J2EE パースペクティブになっていない場合は、「はい」をクリックしてこのパースペクティブを開きます。
9. これが 1 回目の Web プロジェクトの作成である場合は、「OK」をクリックして、必要な製品機能を使用可能にします。

プロジェクト・エクスプローラーに新規 Struts Web プロジェクトが表示されます。対応する Web ダイアグラムが開きます。

ヒント: Web ダイアグラムで「Web ダイアグラム・エディターへようこそ」オブジェクトを選択して、Delete (削除) キーをクリックしてオブジェクトを除去します。



演習のチェックポイント

Struts Web プロジェクトの作成、ランタイム環境の選択、および新規 Web ダイアグラムのオープンについて学習しました。

演習 2: Web ダイアグラム・ノードの作成

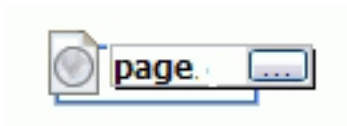
新規プロジェクトの Web ダイアグラムにノードを追加する方法を学習するには、この演習を使用します。

Web ダイアグラムは、Web アプリケーションのフローを視覚化および変更するのに役立ちます。Web ダイアグラム・エディターを使用する前に、以下の用語を理解する必要があります。

- 「ノード」とは、Web ページ、Java Bean、Struts のアクション、または Web アプリケーションなどのリソースを表します。
- 「接続」は、これらのリソースのうちの 2 つの間のロジック・フローを表します。

2 つの Web ページと Struts アクションを表す Web ダイアグラム上にノードを作成する手順は、次のとおりです。

1. パレットの「Web パーツ」ドロワーを開きます。Web ページ・ノードをダイアグラムにドラッグします。ダイアグラムには、デフォルト名 `page.jsp` を持つ新規 Web ページ・ノードが表示されます。



ヒント: Web ダイアグラムで新規ノードを作成する場合は、編集のためにノードの名前が自動的に選択されます。新規名を入力する前に、ダイアグラム内の任意の場所をクリックすると、新規 Web ページ名は同じままになります。ノードを名前変更するには、名前をクリックして強調表示してから、新規名を入力します。

2. Web ページ・ノードの新規名として「`input.jsp`」を入力します。



注: Web ページ・ノードに名前を付けるとすぐに、このノードが表す JavaServer Pages (JSP) ファイルが `DayOfWeek\WebContent` Web プロジェクト・フォルダーに作成されます。

3. 別の Web ページ・ノードをダイアグラムにドラッグします。
4. 2 つ目の Web ページの名前として、「`output.jsp`」と入力します。



5. Struts アクションをダイアグラムにドラッグして、「`action1`」というノードを作成します。

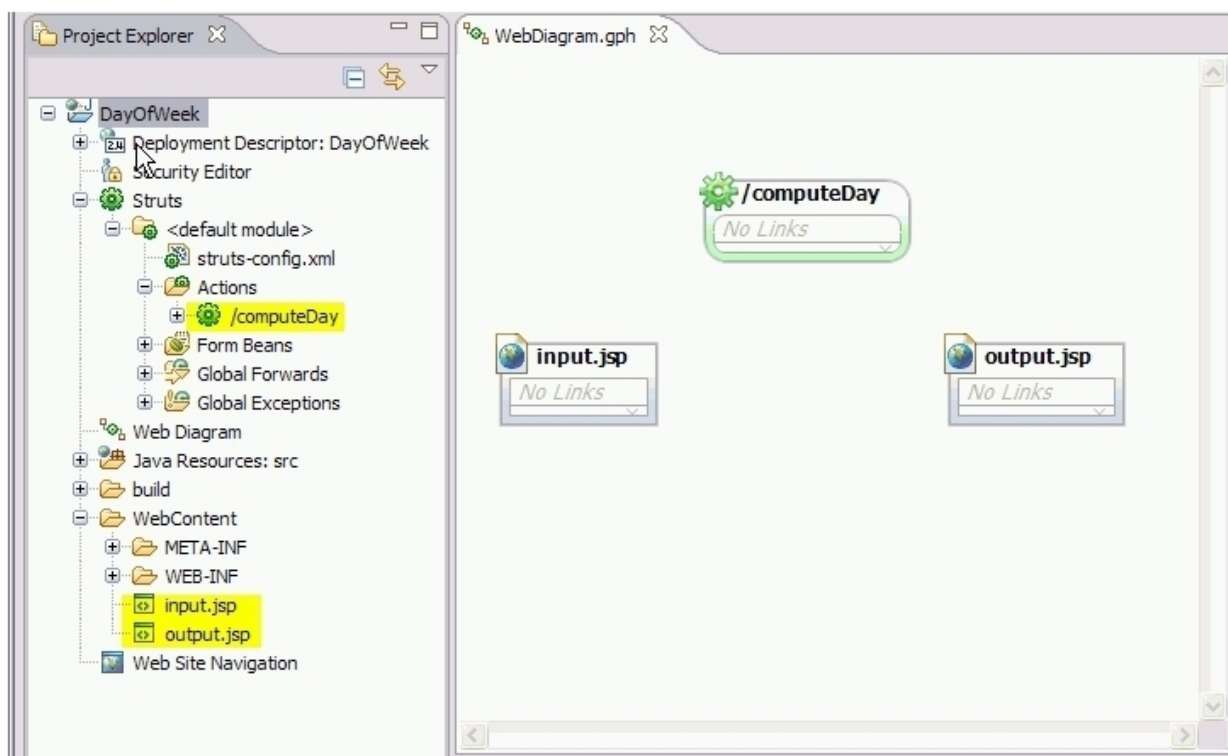


6. このアクションの名前として「computeDay」と入力し、「Enter」を押します。Web ダイアグラムに、「/computeDay」という名前のアクション・ノードが表示されます。プロジェクト・エクスプローラーで /computeDay アクションを表示するには、Web プロジェクト・フォルダーにナビゲートして、次に DayOfWeek¥Struts¥<default module>¥Actions にナビゲートします。



7. Web ダイアグラムを保管します。

Web プロジェクトには、/computeDay、input.jsp、および output.jsp の項目が含まれているはずです。



演習のチェックポイント

この演習では、入力 JSP ファイル、出力 JSP ファイル、および /computeDay アクションを追加しました。

以下のタスクの実行方法を学習しました。

- Web ダイアグラムに JSP ファイルと Struts のアクションを追加する方法。
- プロジェクト・エクスプローラーで JSP ファイルと Struts のアクションを見つける方法。

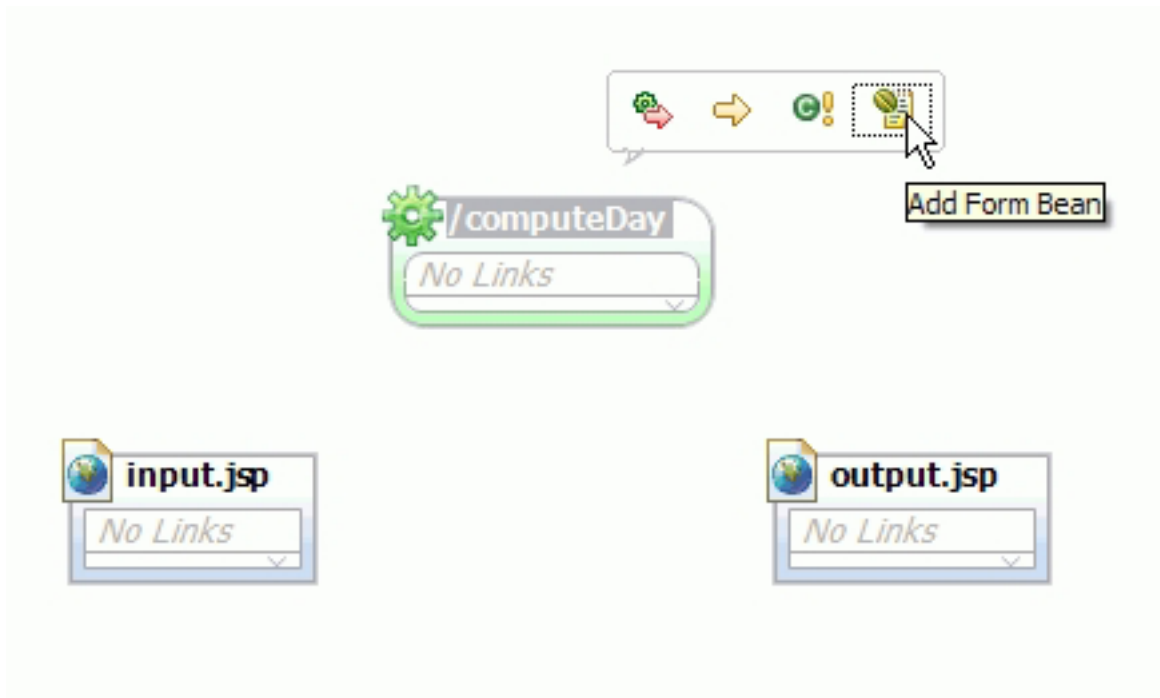
演習 3: フォーム Bean の作成

フォーム Bean の作成方法を学習します。

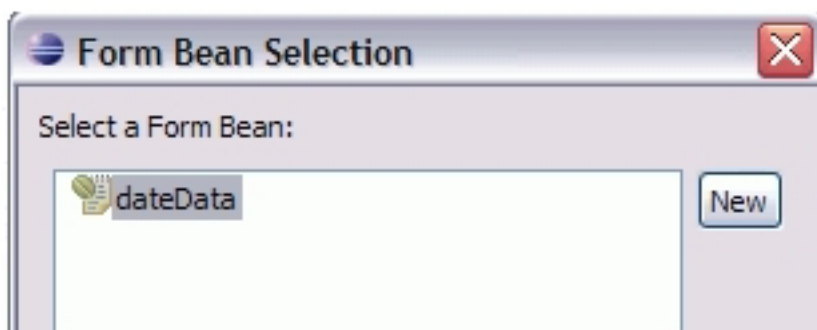
「フォーム Bean」とは、Java Bean の一種です。フォーム Bean は、実行されたクライアント要求からの HTML フォームのデータを保管したり、ユーザーがクリックした Struts アクション・リンクからの入力データを保管したりします。

フォーム Bean を作成する手順は、次のとおりです。

1. ダイアグラムで、/computeDay アクションの上にマウス・ポインターを置いて、「フォーム Bean の追加」をクリックします。



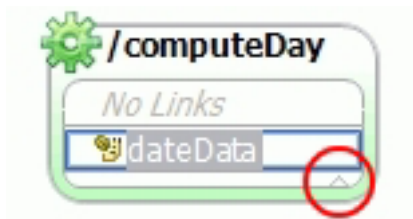
2. 「フォーム Bean 選択」ウィンドウで「新規」をクリックして、「新規フォーム Bean (New Form Bean)」ウィザードを開きます。
3. 「フォーム Bean 名」フィールドに、「dateData」と入力します。「次へ」をクリックします。



4. 「次へ」をクリックして、次のウィザード・ページにスキップします。新規アプリケーションであるため、プロジェクトのページにはまだフォーム情報が含まれていません。含まれている場合は、フィールドのこの情報を後で使用することができます。
5. 「ActionForm クラスに新規のフィールドを作成します」ページで、「追加」ボタンを使用して以下のフィールドを追加して、「次へ」をクリックします。

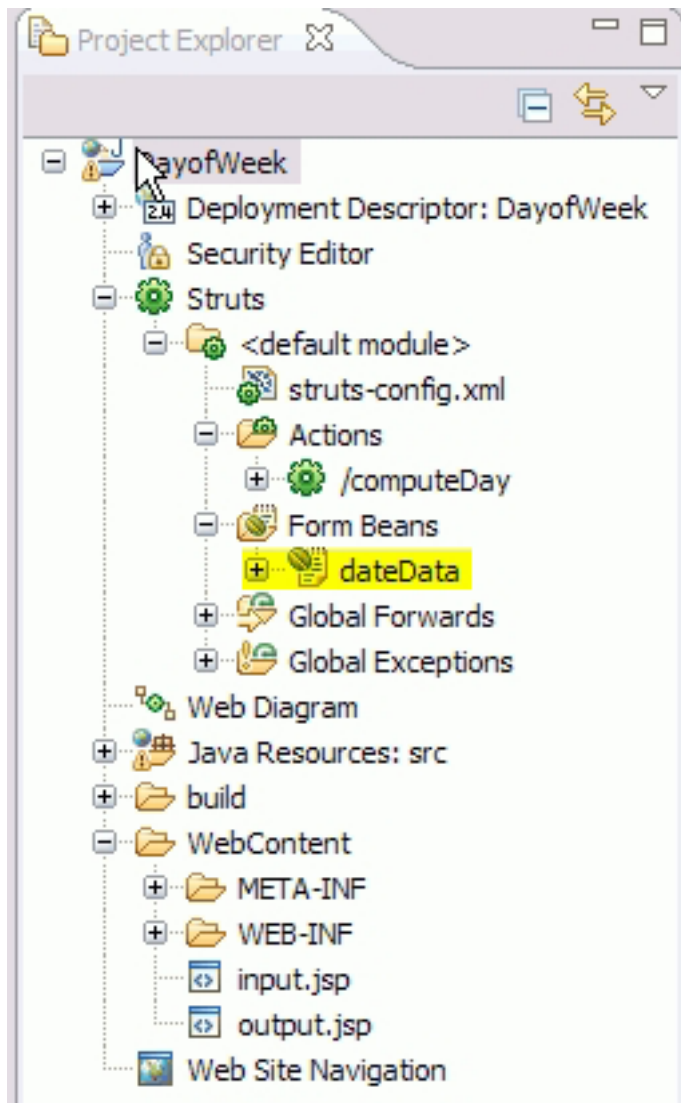
名前	タイプ
year	int
month	int
day	int
dayOfWeek	String

6. 「ActionForm クラスにマッピングを作成します」ページの「Java パッケージ」フィールドに、「com.ibm.dayofweek」と入力します。「終了」をクリックします。更新された「フォーム Bean 選択」ウィンドウが表示されます。
7. 「フォーム Bean の選択」ウィンドウで、先ほど作成した dateData フォーム Bean をクリックします。「OK」をクリックします。
8. /computeDay ノードで、ノードの右下隅の近くにある小さな三角形をダブルクリックします。データ・コンパートメントが展開して、ノードに関連付けられた dateData フォーム Bean が表示されます。



dateData フォーム Bean が /computeDay ノードに追加され、dateData.java ファイルが Web プロジェクト・フォルダーに作成されます。プロジェクト・エクスプローラーで、DayOfWeek\Struts\<default module>\Form Beans フォルダーにナビゲートして、ファイルを表示します。

演習の最後には、プロジェクト・エクスプローラーは次のようになるはずです。



演習のチェックポイント

この演習では、dateData フォーム Bean を作成して、ActionForm クラスを定義しました。

以下のタスクの実行方法を学習しました。

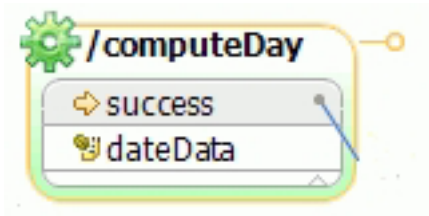
- 吹き出しメニューの表示方法。
- ActionForm クラスのパラメーターの定義方法。
- プロジェクト・エクスプローラーの Web プロジェクト・フォルダーで Java ファイルを見つける方法。

演習 4: Web ダイアグラムでのノードの接続

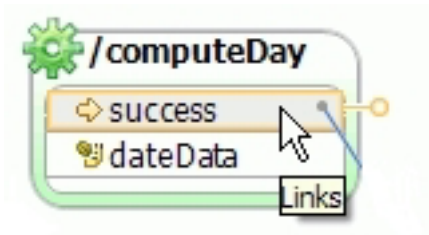
接続ハンドルを使用して、Web ダイアグラムのノードを接続する方法について学習します。

ノードを Web ダイアグラムに配置したので、次のステップではこれらのノードを接続します。このためには、接続ハンドルを使用します。接続ハンドルには、次の 2 つのタイプがあります。

- ノードから接続を描画するには、トップレベル・ハンドルを使用できます。



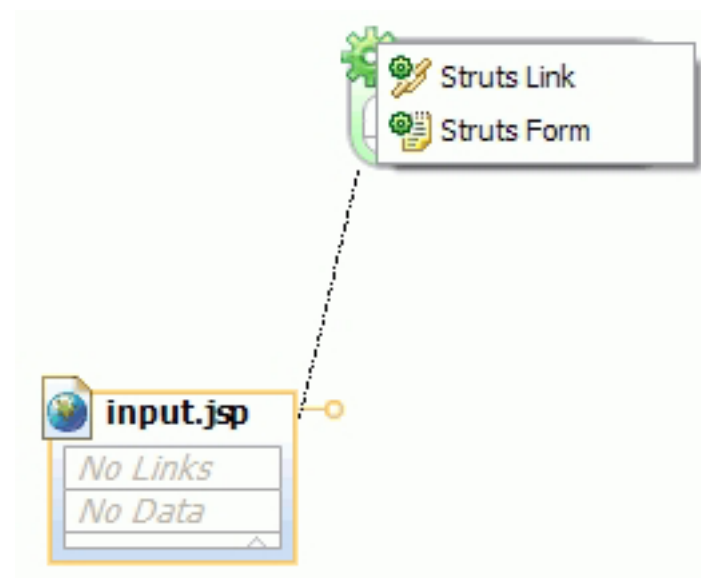
- ノード・コンパートメントにある項目から接続を描画するには、ネストされたハンドルを使用できます。



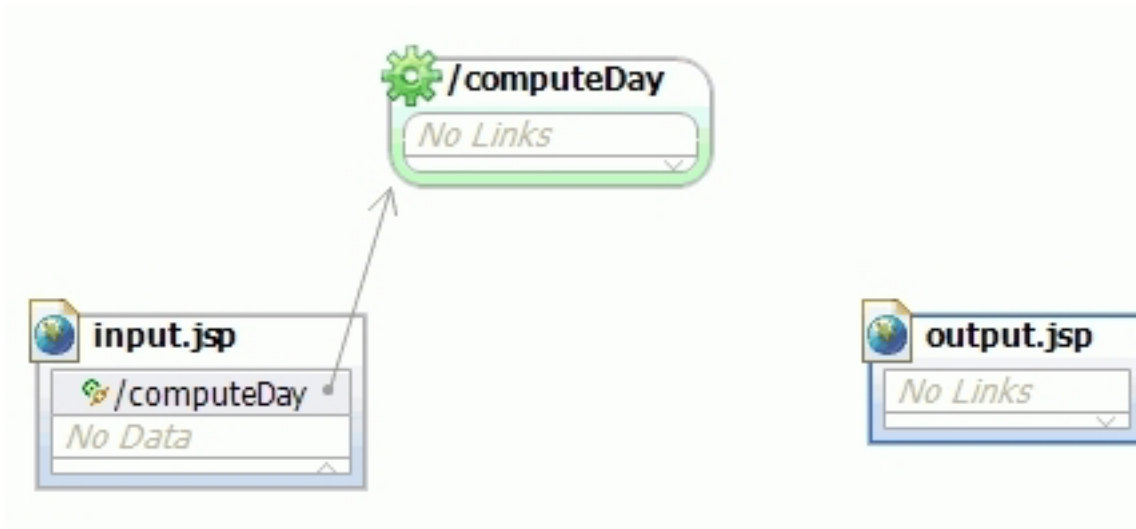
1. ダイアグラムで、input.jsp ノードの上にマウス・ポインターを置きます。ノードが強調表示され、接続ハンドルが表示されます。また、吹き出しアクション・メニューが表示されます。



2. ハンドルをクリックして 入力 JSP ノードから /computeDay アクションまでドラッグします。ポップアップ・メニューが表示されます。



3. 「**Struts フォーム**」を選択して、フォームを処理するときに入力値がアクションに送信されるようにします。Struts フォームが、ノードを `/computeDay` アクションに接続するライン付きで入力 JSP ノードに追加されます。



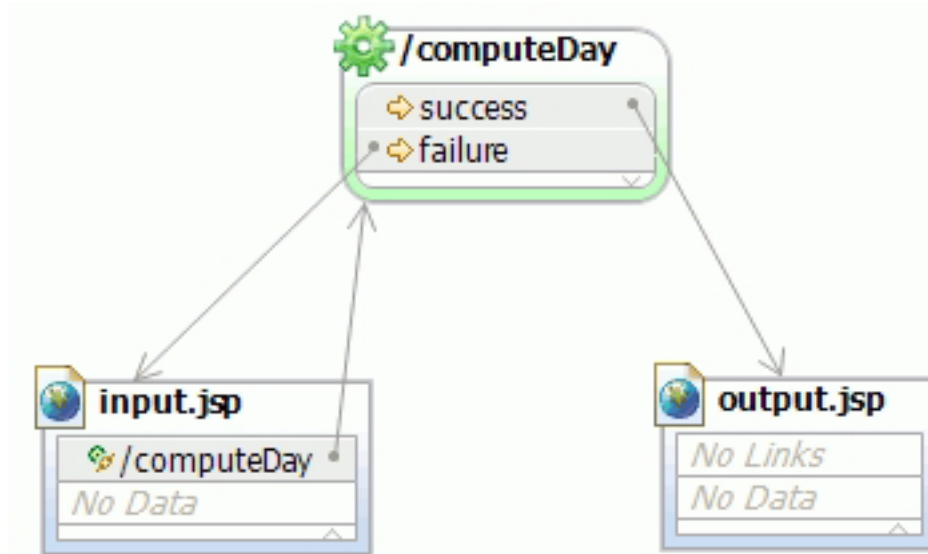
ヒント: あるいは、吹き出しメニューを使用して Struts リンクを入力 JSP ノードに追加し、そのリンク接続ハンドルを `/computeDay` アクションまでドラッグすることもできます。

4. ローカル転送を使用して、`/computeDay` アクションを `output.jsp` ノードに接続します。
- マウス・ポインターを `/computeDay` ノードの上に置いて、吹き出しメニューを表示します。
 - 「**ローカル転送の追加 (Add Local Forward)**」アイコンをクリックします。「success」という名前のローカル転送がノードに追加されます。(デフォルトでは、各ノードで作成した最初のローカル転送に「success」という名前が付いています。)
 - 「success」リンクから出力 JSP ノードまで接続ハンドルをドラッグします。その接続を示すラインがリンクからノードまで描画されます。

ヒント: ノードのリンク・コンパートメント内の各項目にも接続ハンドルがあります。ノードをクリックしてそのハンドルをつかむ際には、リンクのハンドルではなくノードのハンドルをドラッグしていることを確認してください。

5. ローカル転送を使用して、`/computeDay` ノードを入力 JSP ノードに接続します。
- マウス・ポインターを `/computeDay` ノードの上に置いて、その接続ハンドルを入力 JSP ノードにドラッグします。
 - 吹き出しメニューで「**ローカル転送**」を選択します。「failure」という名前のローカル転送がノードに追加され、ラインがローカル転送から入力 JSP ノードまで接続されます。(デフォルトでは、各ノードで作成した 2 番目のローカル転送に「failure」という名前が付いています。)
6. Web ダイアグラムを保管します。

Web ダイアグラム・エディターには、Struts リンクと 2 つのローカル転送が表示されているはずです。



演習のチェックポイント

お疲れ様でした! Web ダイアグラムのノードを接続しました。

以下のタスクの実行方法を学習しました。

- 接続を描画するためにトップレベル・ハンドルかネストされたハンドルを選択する方法。
- Struts リンクを入力 JSP ノードからアクション・ノードに追加する方法。
- ローカル転送をアクション・ノードから出力 JSP ノードに追加する方法。

演習 5: フォーム Bean の編集

フォーム Bean のソース・ファイルと Java リソース・ファイルを、Struts アプリケーションで正しく動作するように編集します。

1. プロジェクト・エクスプローラーで、/computeDay ノードの dateData フォーム Bean をダブルクリックして、Java エディターでフォーム Bean を開きます。
2. エディター内で、ファイルの下部付近にある次のコードの行を探します。

```
ActionErrors errors = new ActionErrors();
```
3. この行のすぐ下に、次のコードを挿入します。

```
if (year < 1582)
{
    errors.add("year",new org.apache.struts.action.ActionError("pre_gregorian"));
}
```
4. DateData.java ファイルを保管して閉じます。
5. プロジェクト・エクスプローラーで、DayOfWeekJava\Resources: src\com.ibm.dayofweek.resources フォルダにナビゲートします。
6. ApplicationResources.properties ファイルをダブルクリックします。
7. ApplicationResources.properties で、errors.header と errors.footer で始まる行から # 文字を除去します。
8. ファイルの下部に次のコードを追加します。

```
pre-gregorian=<li>Date is before 1582, the year the Gregorian calendar began</li>
```

```
# Optional header and footer for <errors/> tag.
errors.header=<ul>
errors.footer=</ul>
pre-gregorian=<li>Date is before 1582, the year the Gregorian calendar began</li>
```

9. ファイルを保管して閉じます。

演習のチェックポイント

この演習では、フォーム Bean のソース・ファイルを編集しました。

以下のタスクの実行方法を学習しました。

- フォーム Bean のソース・ファイルを見つける方法。
- ソース・コードの編集方法。

演習 6: アクション・マッピングの編集

アクション・マッピングは、実行されるアクションを表しています。また、実行時には、アプリケーションのフローを決定します。

アクション・マッピングは、通常はアクション・クラスへの参照を含む構成ファイル・エントリーです。このエントリーには、アクションが使用できるフォーム Bean への参照を含めることができます。また、このアクションのみに表示されるローカル転送と例外を定義することもできます。JavaServer Pages (JSP) ページのコードを作成する場合は、アクションをトリガーするアクション・マッピングを参照することができます。

アクション・クラスには、フォームが実行されたときに呼び出される `execute` メソッドが含まれます。このメソッドでは、算出された結果が「成功」(出力 JSP ページをポイントしている転送によって定義される)、または「失敗」(入力 JSP ページをポイントしている転送によって定義される) のどちらであるかを判別するロジックを指定することができます。

アクション・マッピングを編集するには、次のステップに従います。

1. プロジェクト・エクスプローラーで、DayOfWeek\Java Resources: src\dayofweek.actions フォルダーにナビゲートします。
2. `ComputeDayAction.java` ファイルをダブルクリックして開きます。
3. ファイルですべてのテキストを選択して、「Delete」を押して除去します。
4. 次のコードをコピーしてファイルに貼り付けます。

```
package dayofweek.actions;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import com.ibm.dayofweek.DateData;
```

```

/**
 * @version 1.0
 * @author
 */

public class ComputeDayAction extends Action {

    /**
     * Constructor
     */
    public ComputeDayAction() {

        super();

    }

    public ActionForward execute(
        ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {

        ActionErrors errors = new ActionErrors();
        ActionForward forward = new ActionForward();
        // 戻り値
        DateData dateData = (DateData) form;
        int year;
        int month;
        int day;
        int dayOfWeek;
        int valcen;
        int valleap;
        int valyear;
        int valmon;
        int valday;
        int[] centuries = new int[4];
        int[] months = new int[13];
        String[] daysOfWeek = new String[7];
        centuries[0] = 2;
        centuries[1] = 0;
        centuries[2] = 5;
        centuries[3] = 3;
        months[1] = 5;
        months[2] = 1;
        months[3] = 0;
        months[4] = 3;
        months[5] = 5;
        months[6] = 1;
        months[7] = 3;
        months[8] = 6;
        months[9] = 2;
        months[10] = 4;
        months[11] = 0;
        months[12] = 2;
        daysOfWeek[0] = "Sunday";
        daysOfWeek[1] = "Monday";
        daysOfWeek[2] = "Tuesday";
        daysOfWeek[3] = "Wednesday";
        daysOfWeek[4] = "Thursday";
        daysOfWeek[5] = "Friday";
        daysOfWeek[6] = "Saturday";

        try {
            day = dateData.getDay();
            month = dateData.getMonth();
            year = dateData.getYear();

```

```

        if (month < 3) {
            year--; // year から 1 を減算
        }
        valcen = centuries[year / 100 % 4];
        valleap = year % 100 / 4;
        valyear = year % 100 % 7;
        valmon = months[month];
        valday = day % 7;
        dayOfWeek = valcen + valleap + valyear + valmon + valday;
        dayOfWeek = dayOfWeek % 7;
        dateData.setDayOfWeek(daysOfWeek[dayOfWeek]);
        request.setAttribute("dateData", dateData);

    } catch (Exception e) {

        // 適切な名前と ID を使用してエラーを報告します。
        errors.add("name", new ActionError("id"));
        e.printStackTrace();
    }

    // メッセージが必要な場合は、タグで使用するために、
    // 指定されたキーを要求に保管します。

    if (!errors.isEmpty()) {
        saveErrors(request, errors);
        forward = mapping.findForward("failure");
    } else {
        forward = mapping.findForward("success");
    }

    // 次で終了
    return (forward);

}

}

```

ヒント: プロジェクト内のパッケージの名前は、`dayofweek.actions` と異なることがあります。そのため、先行するコードの最初の行の `dayofweek.actions` をパッケージの名前に置き換えます。

5. ファイルを保管して閉じます。

演習のチェックポイント

これで、アクション・クラスに、Web フォームが実行されたときに呼び出される `execute` メソッドが含まれました。算出された結果が「成功」(出力 JSP ページをポイントしている転送によって定義される)、または「失敗」(入力 JSP ページをポイントしている転送によって定義される) のどちらであるかを判別するロジックを作成しました。

以下のタスクの実行方法を学習しました。

- アクション・クラスに `execute` メソッドを追加する方法。
- アクションの成功または失敗を判別するロジックを作成する方法。

演習 7: JavaServer Pages (JSP) ページへの要素の追加

入力フィールド、プッシュ・ボタン、および出力フィールドを JSP ページに追加して、入出力の動的なユーザー・インターフェースを作成します。

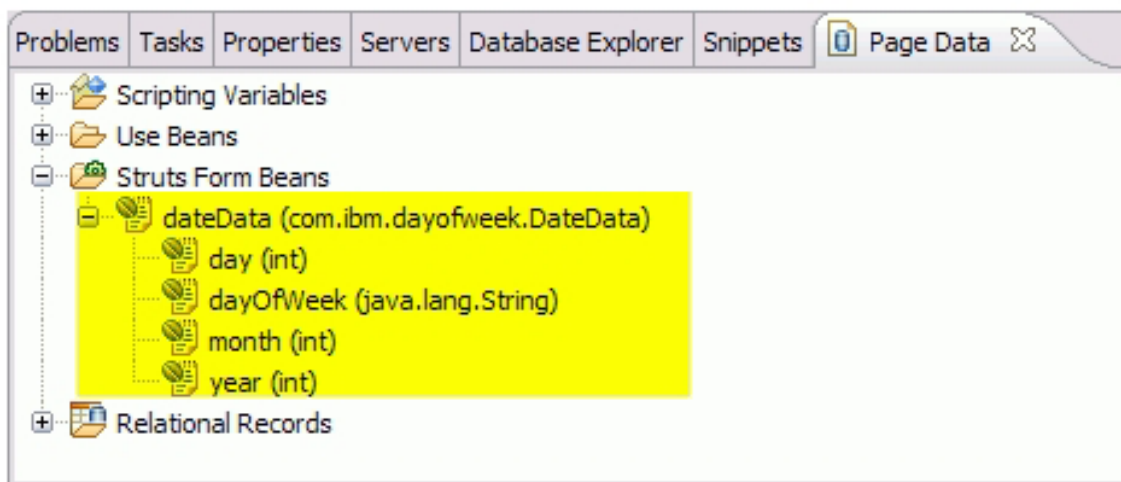
入力フィールドで、結果として作成されたアプリケーションのユーザーが興味のある日、月、および年を入力します。入力ページには、「実行」ボタンと「更新」ボタンがあります。出力フィールドには、対応する曜日が表示されます。これらのページを作成する手順は、次のとおりです。

1. 入力 JSP ページに要素を追加します。
2. 出力 JSP ページに要素を追加します。

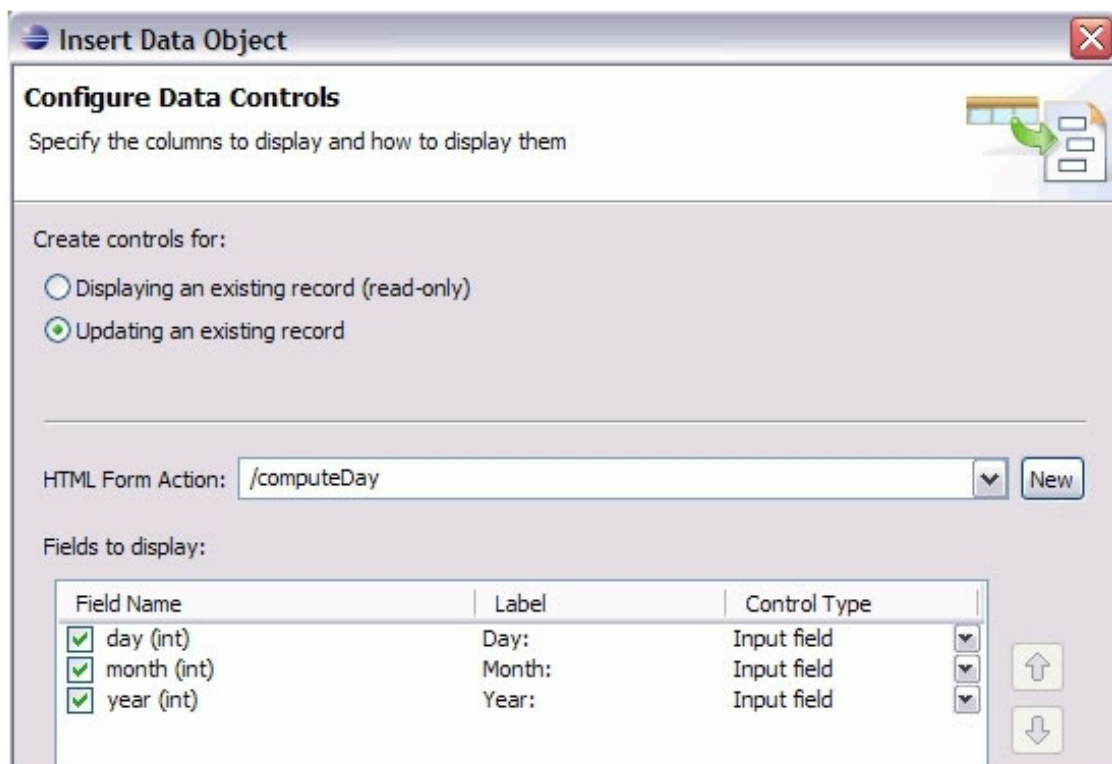
入力ページ

フィールドとプッシュ・ボタンを入力 JSP ページに追加する手順は、次のとおりです。

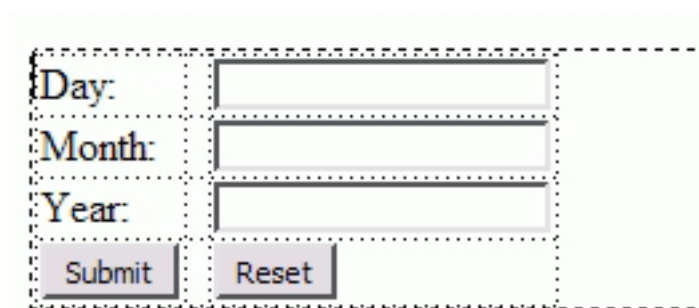
1. Web ダイアグラムで、input.jsp ページをダブルクリックして、Page Designer でページを開きます。
2. Page Designer で、「デザイン」タブをクリックします。
3. メインメニュー・バーで「ウィンドウ」→「ビューの表示」→「その他」を選択します。
4. ウィンドウで「Web」→「ページ・データ」を選択して、「OK」をクリックしてページ・データ・ビューを開きます。
5. ページ・データ・ビューで、Struts フォーム Bean フォルダーを展開します。 dateData フォーム Bean および定義したフィールドを見つけます。



6. Page Designer でページ・データ・ビューから入力 JSP ページに「day」、「month」、および「year」フィールドをドラッグします。「データ・オブジェクトの挿入」ウィンドウが開きます。



7. 上部にある「既存レコードの更新」を選択します。
8. 「オプション」をクリックして、「実行ボタン」と「削除」ボタン」チェック・ボックスが選択されていることを確認します。次に、「OK」をクリックします。
9. 「終了」をクリックして、フォームにフィールドとボタンを作成します。



10. 入力 JSP ファイルを保管して閉じます。

出力ページ

出力 JSP ページにフィールドを追加する手順は、次のとおりです。

1. Web ダイアグラムで、output.jsp ページをダブルクリックして、Page Designer でページを開きます。

2. Page Designer で、「デザイン」タブをクリックします。
3. ページ・データ・ビューで、Struts フォーム Bean フォルダを展開します。dateData フォーム Bean および定義したフィールドを見つけます。
4. Page Designer でページ・データ・ビューから出力 JSP ページに「dayOfWeek」フィールドをドラッグします。「データ・オブジェクトの挿入」ウィンドウが表示されます。
5. 上部にある「既存レコードの表示」を選択して、「終了」をクリックしてページにフィールドを作成します。
6. 出力 JSP ファイルを保管して閉じます。

演習のチェックポイント

この演習では、入力フィールドとプッシュ・ボタンを入力 JSP ページに追加しました。また、出力 JSP ページに出力フィールドを追加しました。

以下のタスクの実行方法を学習しました。

- JSP ページを開いて変更する方法。
- 入力フィールド、出力フィールド、およびボタンを JSP ページに追加する方法。

演習 8: エラー・メッセージの表示

アクションが入力 JavaServer Pages (JSP) ページにエラー・メッセージを戻す場合に、エラー・メッセージを表示します。

ActionError を使用するには、入力 JSP ページでエラー・メッセージを表示するために Struts HTML タグを追加する必要があります。また、入力フィールドをアクション・マッピングに追加する必要があります。

Bean 検証エラーを入力 JSP ページに戻すようアプリケーションを変更する手順は、次のとおりです。

1. DateData クラスの validate メソッドで、入力の年が 1582 年より前であると、このメソッドにより、「グレゴリオ暦以前 (pre-gregorian)」のメッセージとともにエラーが戻されることを確認します。

```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {  
  
    ActionErrors errors = new ActionErrors();  
    if (year < 1582)  
    {  
        errors.add("year",new org.apache.struts.action.ActionError("pre_gregorian"));  
    }  
}
```

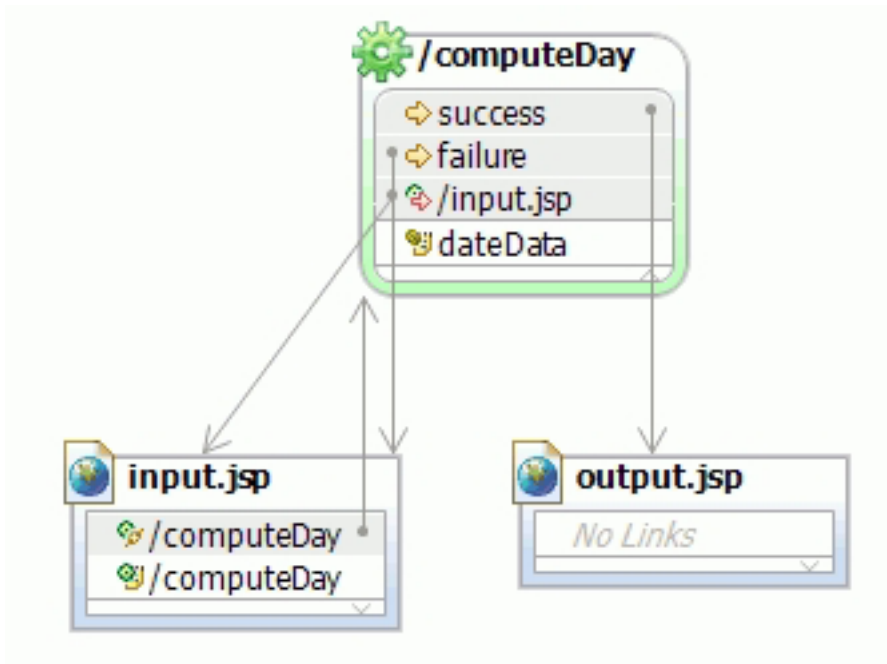
2. ApplicationResources.properties ファイルで「グレゴリオ暦以前 (pre-gregorian)」エラー・メッセージを見つけて、メッセージが次の定義と一致することを確認します。

```
pre-gregorian=<li>Date is before 1582, the year the Gregorian calendar began</li>
```

3. エラー・メッセージのターゲット JSP ページを指定するには、フォーム Bean を使用するアクションに入力を選択します。

- a. Web ダイアグラムを開きます。
- b. ダイアグラムで、マウス・ポインターを /computeDay ノードの上に置いて、ノード接続ハンドルを input.jsp ノードにドラッグします。
- c. ポップアップ・メニューで「アクション入力」を選択します。

Web ダイアグラムは、次のイメージのようになります。



4. エラー・メッセージを表示するには、入力 JSP ページを更新します。
 - a. input.jsp ページをダブルクリックして、Page Designer で開きます。
 - b. パレットの Struts HTML タグ・ドロワーから、エラー・メッセージを表示するページ内の場所にエラー・タグをドラッグします。

入力 JSP ページは、次のイメージのようになります。

- * First error message
- * Second error message
- * Third error message

Day:	
Month:	
Year:	
Submit	Reset

5. 入力 JSP ページを保管して、Web ダイアグラムを閉じます。

演習のチェックポイント

この演習では、入力 JSP ページでのエラー・メッセージの表示を作成しました。

以下のタスクの実行方法を学習しました。

- 入力 JSP ページでエラー・メッセージを表示するために HTML タグを追加する方法。
- アクション・マッピングを変更して、入力フィールドを含むようにする方法。

演習 9: Struts アプリケーションのテスト

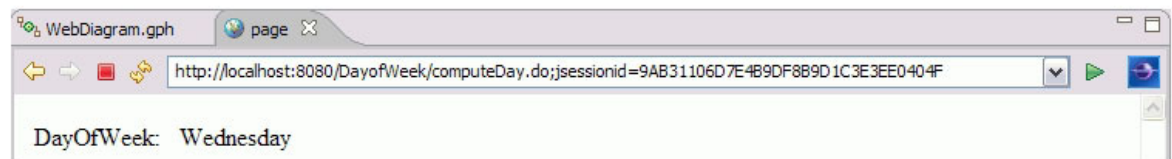
IBM® WebSphere または Apache Tomcat ランタイム環境で Struts アプリケーションをテストします。この環境で以前 JSP ページに追加した入力コントロールと出力コントロールを表示できるはずです。

Struts アプリケーションをテストする手順は、次のとおりです。

1. プロジェクト・エクスプローラーで、input.jsp ファイルを右クリックして、「実行」→「サーバーで実行」を選択します。
2. 「サーバーの選択」ウィンドウで、「WebSphere Application Server 6.x」または「Apache Tomcat Server」を選択して、「終了」をクリックします。
3. 「サーバーの操作」ウィンドウが表示される場合は、「はい」をクリックしてサーバーを始動します。
4. 「終了」をクリックします。入力 JSP ページが Web ブラウザー・ビューで開きます。



5. JSP ページのフィールドに、次のデータを入力します。
 - a. 「日」フィールドに、「10」と入力します。
 - b. 「月」フィールドに、「12」と入力します。
 - c. 「年」フィールドに、「1962」と入力します。
6. 「実行」ボタンをクリックします。出力 JSP ページが、Web ブラウザー・ビューで表示されます。「DayOfWeek」フィールドの値は Wednesday でなければなりません。



入力 JSP のこれらのフィールドに別の日付を入力して、対応する曜日を計算することができます。

演習のチェックポイント

お疲れ様でした! Struts アプリケーションのテストと、シミュレートされた Web ブラウザーでの出力の表示を正常に実行しました。

以下のタスクの実行方法を学習しました。

- テスト環境の選択方法。
- 入力 JSP ページのフィールドにデータを入力する方法。
- 出力 JSP ページで結果をストリング・フォーマットで表示する方法。

要約: Struts を使用した Web アプリケーションの作成

このチュートリアルでは、シンプルな Struts アプリケーションの作成方法を学習しました。

学習した演習

これらの演習を完了すると、以下のタスクを実行できるようになります。

- Struts プロジェクトの作成
- Web ダイアグラムの編集
- フォーム Bean の作成と編集
- 2 つの JavaServer Pages (JSP) ファイルの作成
- アクションおよびアクション・マッピングの作成
- 入力 JSP ページへの「ActionError」タグの追加
- Struts アプリケーションのテスト

追加のリソース

このチュートリアルで取り上げたトピックについてさらに学習したい場合は、次のリソースを検討してください。

- Apache Struts Web サイト (<http://struts.apache.org>) には、Struts フレームワークに関する情報があります。
- オンライン・ヘルプ (「ヘルプ」 → 「ヘルプ目次」) には、Struts アプリケーションと Web ダイアグラムの資料が含まれています。

関連情報

 ibm.com

 eclipse.org