Webアプリケーション・モデルに基づくJSPの動的検証

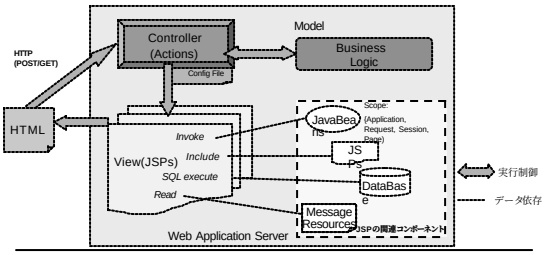
根路銘 崇, 小野 康一, 田井 秀樹, 安部 麻理
日本アイ・ビー・エム（株）東京基礎研究所

内容

- 背景：
  - Webアプリケーションの構成
  - 分業によるチーム開発
  - JSP開発における問題
- Webアプリケーション・モデル
- JSPの動的検証
- ツール・デモ
- 効果
- 今後の課題

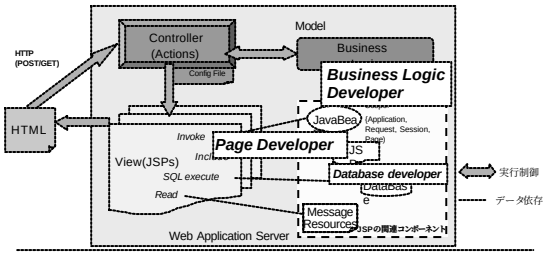
背景：Webアプリケーションの構成

- 近年, JSP Model2アーキテクチャに基づいた構成が主流である
  - 例 :Jakarta Struts
- *JSPは, コンパイルおよび実行時に関連コンポーネントとの依存関係を持つ*



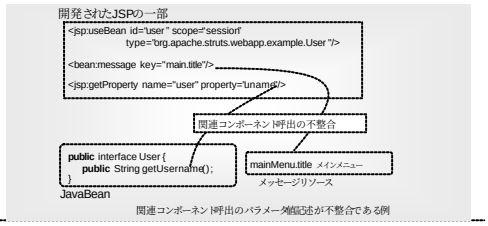
背景：分業によるチーム開発

- 大規模になるほどWebアプリケーション構成単位で, 役割分担による分業を行う場合が多い.
  - ページ数で数えると, 数十, 数百のレベル
- *設計者による設計後に, 平行して開発する場合が多い.*
  - 開発対象物間の動作確認や設計に不整合にないかのテストは重要！



背景：JSP開発における問題

- 受入れテストや統合テストで以下のような不具合が発生しやすい
  - 関連コンポーネントとの依存関係部分の不整合
  - 文法の誤り
  - 画面レイアウトの悪さ

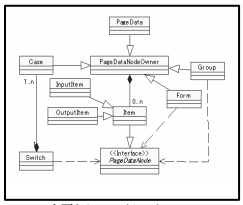


背景：JSP開発における問題

- 主要な原因
  - JSP開発に必要な仕様情報があいまい
    - 定義が不十分 / 仕様変更の対応方法に漏れ
  - JSP検証のための実行方法が複雑
    - 関連コンポーネントの用意の大変さ / 分岐ロジックの検証漏れ
- 解決策
  - JSP要素とその関連コンポーネントに関する正確かつ漏れのない仕様（モデル）に基づいた開発
    - 関連コンポーネントの依存関係＝動的な値の入出力項目　画面の観点より
  - 動的実行に基づく自動検証
    - 状態に依存したJSP要素の振る舞い検証
      - 関連コンポーネントの依存関係と仕様との整合性検証
    - 人間による実行および実行結果の確認
      - 開発したJSPに関して, 実行結果における画面レイアウトの検証

Webアプリケーション・モデル

- JSPのコンポーネント呼出をモデル化
  - PageData
    - 要素：入力項目/出力項目
    - 関連：従属/繰り返し/分岐



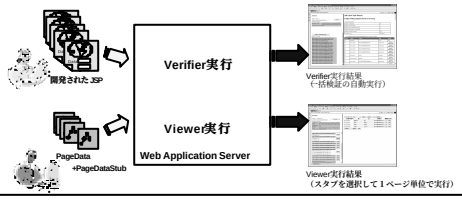
主要なクラス・ダイアグラム

```
<PageData targetPage="Login.jsp" >
  <Switch>
    <Case><OutputItem name=" welcome_msg " /></Case>
    <Case><OutputItem name=" error " /></Case>
  </Switch>
  <Group name="history_group" multiple="true" maxSize="-1">
    <OutputItem name="accesshisoty" required="true">
      <Presentation name=" bean:write">
        <Param name="output param">param value</Param>
      </Presentation>
    </OutputItem>
  </Group>
  <Form name="MainForm" method="post">
    <InputItem name="page_hidden_number" multiple="true" >
      <Presentation name=" html:hidden">
        <Param name="maxLength">20</Param>
      </Presentation>
    </InputItem>
    <InputItem name="request" multiple="true" >
      <Presentation name=" html:radio">
        <Param name="maxLength">20</Param>
      </Presentation>
    </InputItem>
  </Form>
</PageData>
```

XML記述の例

JSPの動的検証

- 開発したJSPおよびPageDataを入力とした2つの動的検証の自動化
  - Verifier実行
    - JSPの記述とPageDataとの整合性を比較検証
      - 主要な検証結果：OK / UNKNOWN ERROR（余計な記述） / MISSING ERROR（記述が不足）
    - 全てのJSPを一括で検証実行および実行結果ログ出力（HTML形式）
  - Viewer実行
    - 関連コンポーネントのスタブを用意し，実行結果を出力
    - 利用者は，実行確認，実行結果画面の表示を確認する



デモンストレーション

-PageDataエディタ
-JSPの動的検証ツール

効果

- JSP１０ページ（平均18個のJSP要素）に対し，ツールを利用する開発者Aとツールを利用しない開発者Bで検証結果を比較

検証担当者	不整合が検出されたJSP要素の記述数	検出とその修正に必要とした時間
開発者A	16個	9分
開発者B	13個	40分

表．JSPファイル10枚の検証結果

  - 検証にかかった時間は，開発者Bは開発者Aの4.4倍
    - 検証の高速化
  - 結合テストで，開発者B担当のJSPは不整合要素が2つ発見
    - 品質の向上(開発者A)
  - 開発者Aは，Viewer実行により3ページのレイアウトを修正
    - 受入れテスト/結合テスト時の修正箇所を削減

今後の課題

- PageDataへの更新
  - 不十分なPageData部分を，開発したJSPから更新するという利用形態への対応
- Verifier実行のJSP scriptletサポート
  - 任意のJSP要素記述の検証
- 更に多くの利用データに基づく不整合パターン抽出
  - 開発物の品質を更にあげるためのフィードバック

BACK UP

問題解決に向けた既存技術

- Cactus
  - JUnitの拡張.
  - 必要なテスト環境が用意されればJSPの単体テストには便利である. しかし, ページ開発者にとってテストのためのロジックをJavaで作成する手間が発生する. また, テスト環境が仕様に沿って作成される保証はない.
- TagUnit
  - JUnitのassertionをJSP内のJSP要素に対して行う.
  - こちらも検証のためのロジックをページ開発者が用意する必要があり, その作成において誤りが発生する可能性がある. また, 対象となるJSPに対して, 検証用のJSPタグを埋め込む事は好ましくない. そのJSPタグを解除する際に, 正しいJSP要素に影響を及ぼす可能性がある.

PageDataStub

- PageDataに基づいたスタブデータ
  - PageDataとPageDataStubの関係は, 1対N.

PageDataエディタの項目定義
型を指定して, スタブデータを自動生成)

```
<PageDataStub>
   <welcome_msg>Welcome</welcome_msg>
   <history_group>
      <accesshisoty>2003/08/10</accesshisoty>
   </history_group>
   <history_group>
      <accesshisoty>2003/08/19</accesshisoty>
   </history_group>
   <MainForm>
      <page_hidden_number>X00001</page_hidden_number>
      <request>true</request>
   </MainForm>
</PageDataStub>
```

XML表現の例

JSP検証環境の概要