

JSP, Struts und JSF

UnFUG

BlueC0re

18. November 2010

Inhaltsverzeichnis

- 1 JSP
 - Einführung
 - Syntax
 - Arbeitsweise
 - TagLibs
- 2 Struts
 - Einführung
 - Arbeitsweise
- 3 JSF
 - Einführung
 - Arbeitsweise
- 4 Quellen

JSP

Einführung

Geschichte

- Januar 1999: Servlet Spezifikation 2.1
- Juni 1999: JSP 1.0
- Dezember 1999: JSP 1.1/Servlet 2.2
- 2001: JSP 1.2/Servlet 2.3
- 2003: JSP 2.0/Servlet 2.4
- 2006: JSP 2.1/Servlet 2.5

Einführung

Warum JSP?

- Servlets bieten nur eine Ausgabe über `out.print()`
- Entwickeln und Anpassen des Websitedesigns sehr schwierig und aufwändig
- Eine Trennung von Inhalt und Aussehen wäre sinnvoller

Einführung

Vorteile

- Mächtig - Sind im innern Servlets
- Effizient - Werden zu Byte-Code kompiliert
- Portabel - Sind standardisiert und funktionieren auf verschiedensten Servern
- Handlich - Verwenden JavaBeans
- Flexibel - Es können eigene Tags erstellt/verwendet werden
- Einfach - HTML/XML und Java

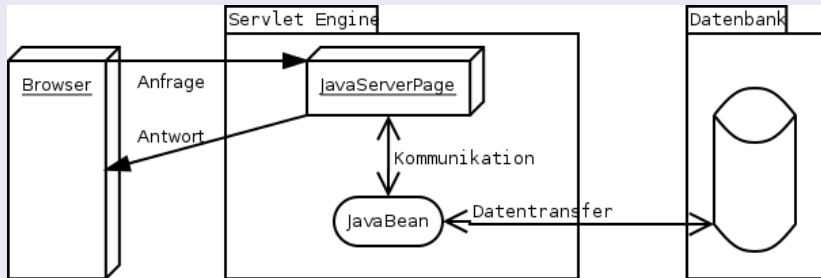
Einführung

Nachteile

- Trennen von Logik und Darstellung ist leichter gesagt als getan
- Ohne Framework sind viele Probleme der Webentwicklung noch vorhanden
 - Anwendungsfluss geht in der Darstellungsebene verloren
 - Logik auch in der Darstellungsebene
 - Designer müssen sich mit Java auskennen
- Debuggen ohne gute IDE kann schwerer sein

MVC

MVC 1



Servlet

```
1 public class HelloServlet2 extends HttpServlet {
2     protected void doGet(HttpServletRequest request ,
3     HttpServletResponse response )
4         throws ServletException , java.io.IOException {
5         response.setContentType("text/html");
6         java.io.PrintWriter out = response.getWriter();
7         out.println("<html>");
8         out.println("<head>");
9         out.println("<title>Hello_Servlet </title>");
10        out.println("</head>");
11        out.println("<body>");
12        out.println("<h1>Hello_World!</h1>");
13        out.println("The_time_is_<i>"+new Date()+"</i>");
14        out.println("</body>");
15        out.println("</html>");
16        out.close();
17    }
18 }
```

JSP

```
1 <html>
2   <head>
3     <title>Hello JSP</title>
4   </head>
5   <body>
6     <h1>Hello World</h1>
7     The time is <%= new java.util.Date() %>
8   </body>
9 </html>
```

Syntax - Elemente

Actions

`<action:actionName>...</action:actionName>`

- beeinflussen das Verhalten der Anwendung
- erstellen/verändern/löschen Objekte
- können über taglibs erstellt werden

Syntax - Elemente

Ausdrücke

`<%= Ausdruck %>`

- Wertet Ausdrücke aus und gibt deren Ergebnis aus
- equivalent zu `<% out.print(Ausdruck) %>`
- PHP pendant: `<?= Ausdruck ?>`

Syntax - Elemente

Scriptlets

`<% code %>`

- Können mit beliebigem Java-Code gefüllt werden (außer Methodendeklarationen)
- sogenannte Scriptlets
- PHP pendant: `<?php code ?>`

Syntax

Deklarationen

`<%! code %>`

- dienen ausschliesslich zur Deklaration von Variablen und Methoden
- Variablen und Methoden werden zu Klassenvariablen und -methoden des Servlets

Syntax - Elemente

Direktiven

`<%@ ... %>`

- bietet angaben zur Seite
 - contentType
 - Encoding
 - cache
 - Vererbung
- bindet java packages ein
- bindet andere Dateien ein
- fügt Taglibs hinzu

Syntax - Elemente

Kommentare

HTML: `<!--... -->`

JSP: `<%--... --%>`

- HTML-Kommentare werden an den Client übertragen
- JSP-Kommentare werden nicht an den Client übertragen

Syntax - Elemente

Expression Language

Seit JSP2.0: $\$(\text{Ausdruck})$

Seit JSP2.1: $\#(\text{Ausdruck})$

- Erlaubt direkten Zugriff auf Bean elemente
- Verwendete Methode (get/set) hängt vom Kontext ab
- Standardoperatoren von Java

Syntax - Objekte

Objekte

Objekt	Beschreibung
request	Referenz auf die HTTP-Anfrage des Clients
response	Referenz auf die HTTP-Antwort des Servers
session	Referenz auf die HTTP-Session
application	Zugriff auf Konfigurationseinstellungen
out	Zum schreiben in den Outputstream
config	Zugriff auf Konfigurationseinstellungen des Servlets
page	Referenz auf die Instanz der Klasse
exception	Zur Fehlerbehandlung (nur in Fehlerseiten)
pageContext	kapselt alle Objekte über getter, Funktionen zur Anwendungssteuerung

Arbeitsweise - Lifecycle

Lifecycle einer JSP-Seite

- 1 JSP wird über einen Compiler in ein Servlet verwandelt (falls nicht schon geschehen)
- 2 Servlet wird in Bytecode kompiliert (falls nicht schon geschehen)
- 3 Bytecode wird geladen (falls nicht schon geschehen)
- 4 Objekt wird erstellt
- 5 init wird aufgerufen
- 6 service wird aufgerufen
- 7 destroy wird aufgerufen

Arbeitsweise - Anfragen verarbeiten

Verarbeitung von HTTP-Anfragen

- service-Funktion des Servlets wird mit dem Request aufgerufen
- Die umgewandelte JSP-Seite führt Code aus
- Führt ggf JavaBeans als Backend aus
- schreibt Ergebnisse in den OutputStream der Response
- Reponse wird an den Client versendet

TagLibs

Problem

- Für dynamische Seiten muss immer noch Java-Code in die Seite eingebaut werden
- Designer muss Java können/verstehen
- Programmierer muss Designen können/das Design anpassen

TagLibs

Lösung

- Auslagern von Funktionen in Taglibs
- Designer muss nur mit den Tags arbeiten (sollte er von HTML gewohnt sein)

TagLibs

Was wird benötigt?

- Ein Tag-Handler (implementiert `javax.servlet.jsp.tagext.SimpleTag`)
- Eine Tag-Library-Descriptor Datei (beschreibt das Tag mittels XML)

JSTL

Java Standard Tag Library

- Seit JSP2.0 vollständig vorhanden
- bietet allgemeine Tags zur Vereinfachung häufiger Aufgaben

JSTL

Java Standard Tag Library

Name	Prefix	Aufgabe
core	c	Basistags, vorallem Programmablauf wie Verzweigungen und Schleifen
fmt	fmt	Tags zur Formatierung und Internationalisierung
xml	x	Ähnlich wie core, nur für XML und mit xpath
sql	sql	Ansprechen von Datenbanken aus JSP
functions	fn	Funktionen zur String-Manipulation auf EL-Ausdrücke

Struts

Einführung

Geschichte

- 2000: Craig R. McClanahan startet das Apache Struts Projekt
- 2001: Version 1.0 wird released
- 2003: Struts 1.1
- 2004: Struts 1.2
- 2008: Struts wird mit WebWorks 2 zu Struts 2.0 verschmolzen

Einführung

Warum Struts?

- Struts dient der Umsetzung einer MVC-Architektur
- Viele Tag-Libraries
- Vom Code getrennte Navigation

Einführung

Vorteile

- Zentrale Datei-basierte Konfiguration
- FormBeans
- Zusätzliche Bean-Tags
- HTML-Tags
- Validierung von Formularen
- Einheitliches Vorgehen

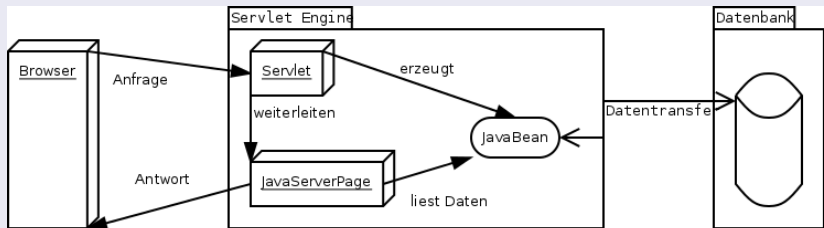
Einführung

Nachteile

- Große Lernkurve
- Schlechte Dokumentation
- Wenig transparent
- Strenges Vorgehen

Arbeitsweise - MVC

MVC 2



Arbeitsweise - Teile

Teile einer Strutsanwendung

- 1 Präsentation - JSP
 - Tiles - Templatesystem
 - Validatorframework - Elemente zur Validierung von Eingaben
- 2 Form-Bean - JavaBean Schnittstelle zwischen Formular und Action
- 3 Action - Schnittstelle zum Backend, schreibt und liest Daten
- 4 JavaBeans - Backend

Arbeitsweise - Form

Form-Bean

- JavaBean
- XML

Arbeitsweise - Validator

Validator

- Java-FormBeans von ValidatorForm ableiten \Leftarrow validate Methode zur Überprüfung der Parameter
- XML-FormBeans \Leftarrow festlegen von Parametern der vorhandenen Regeln
- Eigene Regeln können als Plugin erstellt werden

Arbeitsweise - View

Darstellung

Struts bietet einige Taglibs zur Erstellung der View, darunter:

- HTML: Formulare, Links und Standardhtml-elemente;
- BEAN: Dient der Internationalisierung und Zugriffen auf die Session
- LOGIC: Verzweigungen und Schleifen
- NESTED: Ausgabe und Verwendung von Objektorientierten Beziehungen

Arbeitsweise - Navigation

Navigation

- Eine Action Klasse gibt ein ActionForward Objekt zurück
- Dieses Objekt legt fest welche JSP seite verwendet werden soll
- Kann über die Verwendung von Namen in der Konfiguration geschehen

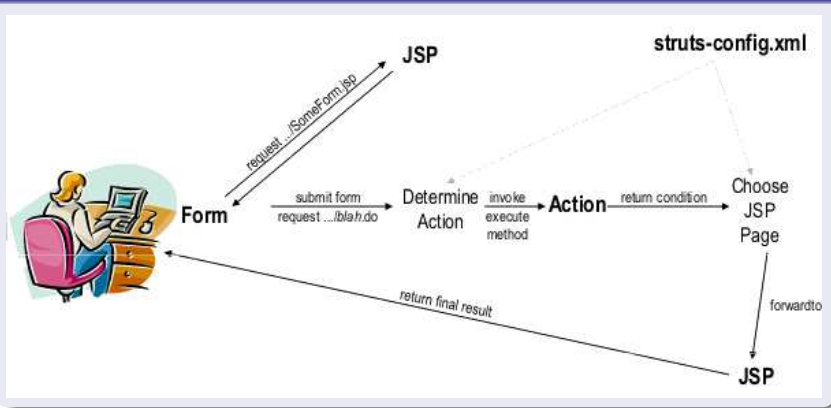
Arbeitsweise - Anfrage

Ablauf einer Anfrage

- 1 User startet eine Anfrage \Rightarrow bekommt ein Formular
- 2 Formular wird an eine Url mit der Endung *.do gesendet
- 3 Die verknüpfte Action wird ausgeführt
- 4 Das Ergebnis wird an die verknüpfte JSP weitergeleitet

Arbeitsweise - Anfrage

Ablauf einer Anfrage



Arbeitsweise

Die 6 Standardaktionen

- 1 Konfigurationsdatei (struts-config.xml) anpassen
 - Url mappings eintragen
 - Navigationsvarianten eintragen
 - ggf. FormBeans eintragen
- 2 ggf. FormBean erstellen
- 3 ggf. Bean zur Verarbeitung erstellen
- 4 Action-Klasse zur Anfragebehandlung erstellen
- 5 Formular erstellen welches auf die Url verweist
- 6 Ergebnis in einer JSP-Seite anzeigen

JSF

Einführung

Geschichte

- März 2004: JavaServerFaces 1.0 (JSR-127)
- Mai 2004: JavaServerFaces 1.1 (JSR-127)
- Mai 2006: JavaServerFaces 1.2 (JSR-252)
- Juni 2008: JavaServerFaces 2.0 (JSR-314)

Einführung

Warum JavaServerFaces?

- Dient der Umsetzung einer MVC-Architektur
- Viele Tag-Libraries
- Vom Code getrennte Navigation
- GUI-Anwendung
- Standardisiert
- Hohe Abstraktion

Einführung

Vorteile

- Eigene GUI-Elemente
- Event-Handling
- Managed Beans
- Eingebauter Ajax-Support
- Eingabedatenkonvertierung und Validierung
- Zentrale Konfigurationsdatei
- Einheitliches Vorgehen
- Nicht auf HTML beschränkt

Einführung

Nachteile

- Große Lernkurve
- Schlechte Dokumentation
- Wenig Transparenz
- Wenig Tools (Netbeans als IDE)
- Strenges Vorgehen

Einführung

Vorteile gegenüber Struts

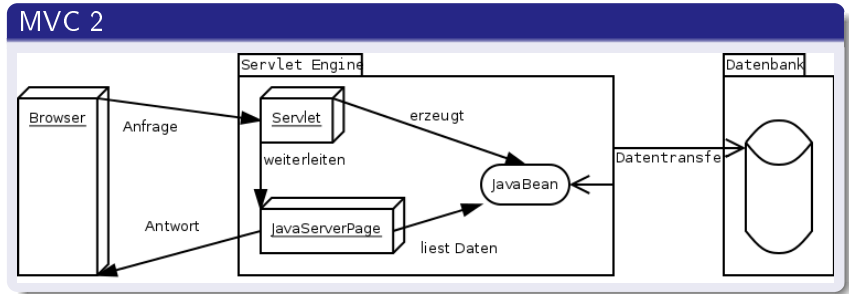
- Eigene Komponenten
- Bean Zugriff per Name
- Offizieller Teil von Java EE
- Expression Language
- Einfachere Controller und Beans
- Einfachere Konfigurationsdatei
- Besserer Potentieller Tool-Support (zb. Drag&Drop)

Einführung

Nachteile gegenüber Struts

- Verwirrung bei den Dateinamen
 - Dateien haben die endung xhtml oder jsp
 - Urls enden mit faces oder jsf
 - Schwer die xhtml Dateien vor externem Zugriff zu schützen
 - Schwer in der Konfigurationsdatei auf andere Dateien zuverweisen
- Formular und Handler haben dieselbe URL (foo.jsf vs foo.jsp und foo.do)
- Weniger eingebaute Validierungstechniken
- Fehlende Clientseitige Validierung

Arbeitsweise - MVC



Arbeitsweise - Teile

Teile einer JSF-Anwendung

- 1 Listener
- 2 Handler (Event-, Phase-, ...)
- 3 BackingBeans
- 4 Validierer/Konvertierer

Arbeitsweise - Form

Formulare

- Beliebige Java Klassen wählbar
- Beliebige Methode wählbar zur Verarbeitung
- Innerhalb der xhtml/jsp Datei über EL und Tags ansprechbar

Arbeitsweise - Validator

Validator

- Validierung wird in der View angegeben
- Fehlermeldungen können leicht übersetzt werden
- Es können wieder auch beliebige Methoden zur Validierung gewählt werden (Nur signatur muss stimmen)

Arbeitsweise - View

Darstellung

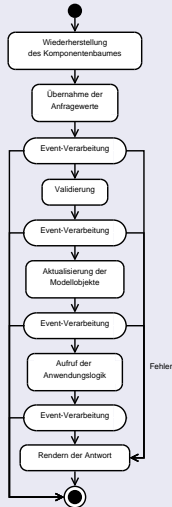
- JSF bietet einige Taglibs zur Erstellung der View
- Komponentenbasiert
- Teilweise unabhängig vom Renderkit

Arbeitsweise - Navigation

Navigation

- Navigation läuft primär über Strings
- faces-config.xml legt die Regeln fest

Ablauf einer Anfrage



Quellen

Quellen



j2ee: Intro to JSP.

<http://condor.depaul.edu/~mwright1/j2ee/lectures/cla>



Ajax & Java EE Training, Tutorials, Consulting, Books & Resources

<http://www.coreservlets.com/>



Oracle JSP

<http://www.oracle.com/technology/docs/tech/java/serv>



Apache Struts Projekt

<http://struts.apache.org/>



Oracle JSF

<http://www.oracle.com/technetwork/java/javasee/javase>

Quellen



Bernd Müller.

JavaServerFaces: Ein Arbeitsbuch für die Praxis.
Hanser Verlag, 2006.



Matthias Weißendorf.

Struts: Websites effizient entwickeln.
W3L Verlag, 2008.