ARS

Member of

TIMETOACT GROUP
SOFTWARE & CONSULTING

JavaServer Pages - Grundlagen

Java EE Grundlagen

ARS

Inhalte dieses Kapitels

Allgemeines

Model2-Architektur

Syntax

Expression Language

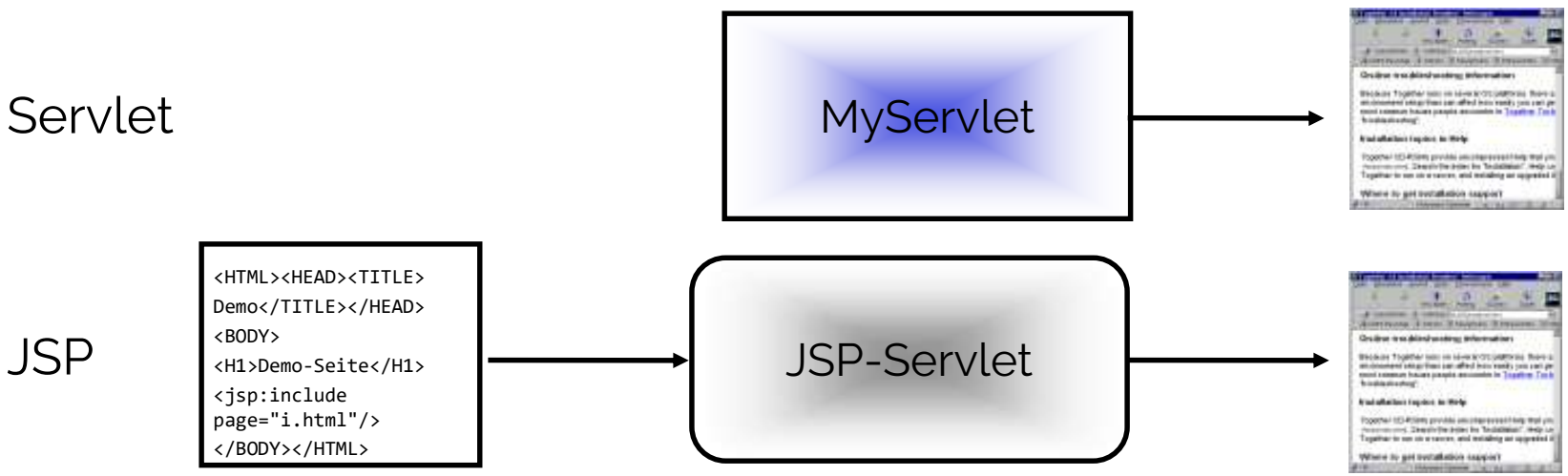
Zusammenfassung

ARS

Allgemeines [1|4]

Unterschiede zwischen Servlets und JSPs

- Servlets = serverseitige Java-Objekte, die (meist) (X)HTML erzeugen
- JSPs = Servlets, die als Skript geschrieben werden
  - Meist in HTML, aber auch andere Ausgabeformate möglich
  - Kompilieren in Bytecode vor der Ausführung
  - Einbettung von Java-Quelltext möglich
  - XML-Syntax, Ausgabelogik in Form von *Custom Tags*



ARS

Allgemeines [2|4]

Charakteristiken

- Text-Template (HTML, XML, JSON, …) mit
  - Expression Language (Nutzung von JavaBeans)
  - Custom Tags (benutzerdefinierte Tags)
  - Verarbeitung als XML-Datei
- Entwicklung durch Webdesigner
  - Kein Java-Know-How notwendig
- Zur Laufzeit automatisch in Servlets kompiliert
  - Hot Code Replacement möglich
- Trennung von Design und Geschäftslogik
- Ähnlich zu Microsoft Active Server Pages (ASP)

ARS

Allgemeines [3|4]

Vergleich von Servlets und JSPs

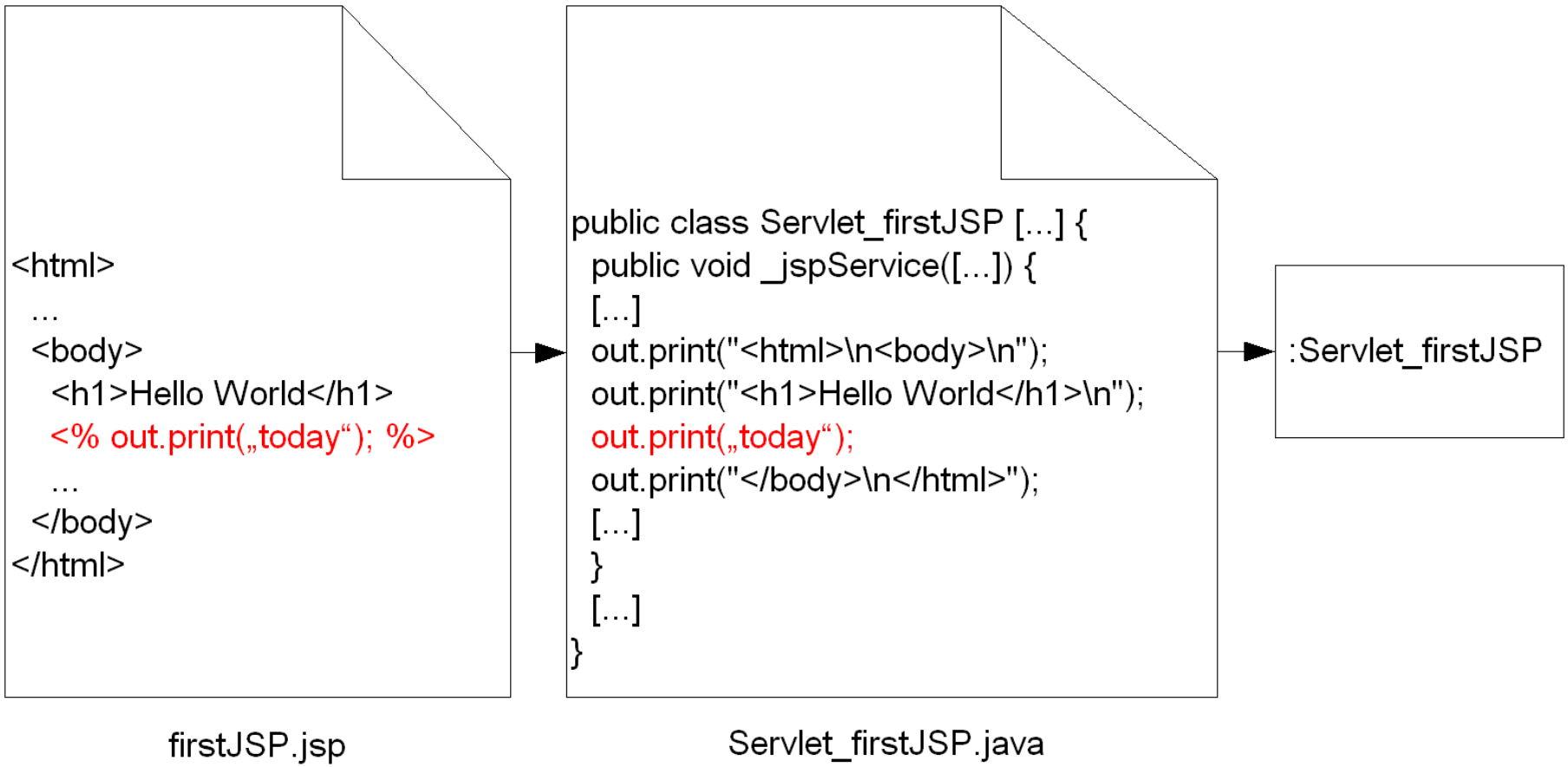
- JSPs = Servlets
  - Automatische Registrierung als Servlet mit Standard-URL-Mapping
  - Serielle Ausführung
  - Manuelle Registrierung im Web Deployment Descriptor
    - angepasstes URL-Mapping, Initialisierungsparameter etc.
- Nachteile bei Servlets (ohne JSPs)
  - Ausgabe steht direkt im Java-Code
  - Schlechte Wartbarkeit bei Design-Wechsel
  - Keine Rollentrennung zwischen Webdesigner und Programmierer
- Probleme bei JSPs
  - Komplexe Ausgabelogik führt zu komplexem XML
  - Serielle Ausführung begrenzt Anwendbarkeit
    - Facelets als Ersatz für JSF

ARS

Allgemeines [4|4]

Lebenszyklus analog Servlet-Klassen

- Kompilieren JSP→Servlet-Klasse zur Laufzeit durch Webcontainer

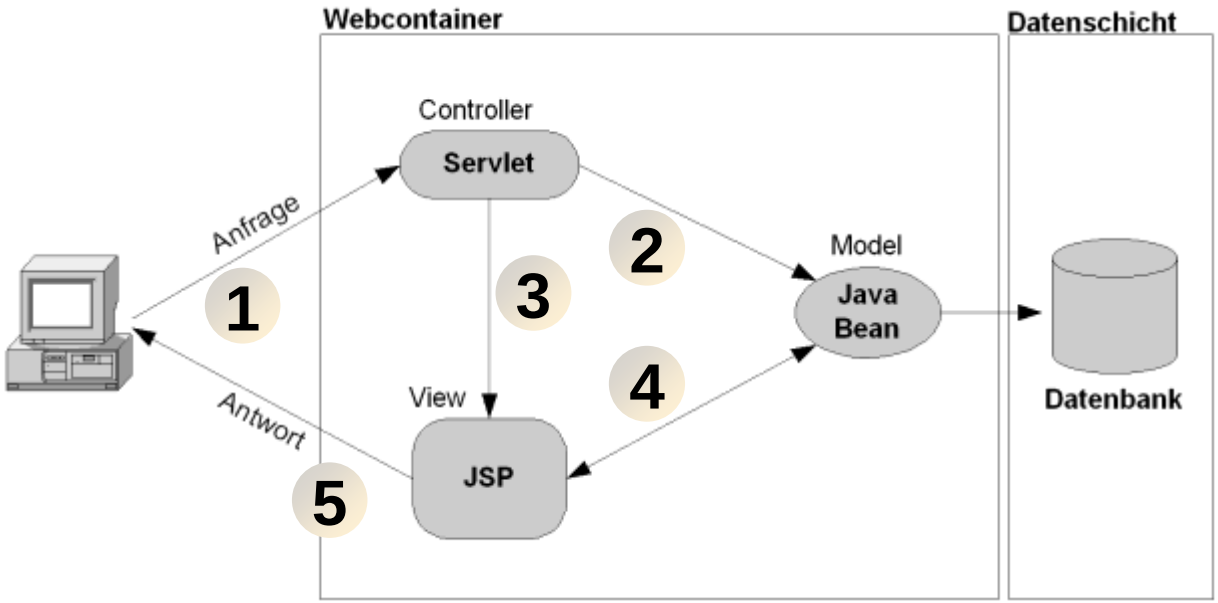


ARS

Model2-Architektur [1|2]

Push Mode

- Client ruft Controller Servlet auf 1
- Controller Servlet enthält Geschäftslogik und
  - Manipuliert/erzeugt Model 2
- Servlet übergibt Kontrolle an JSP 3
- JSP verwendet Model 4 für die Erzeugung der Ausgabe 5

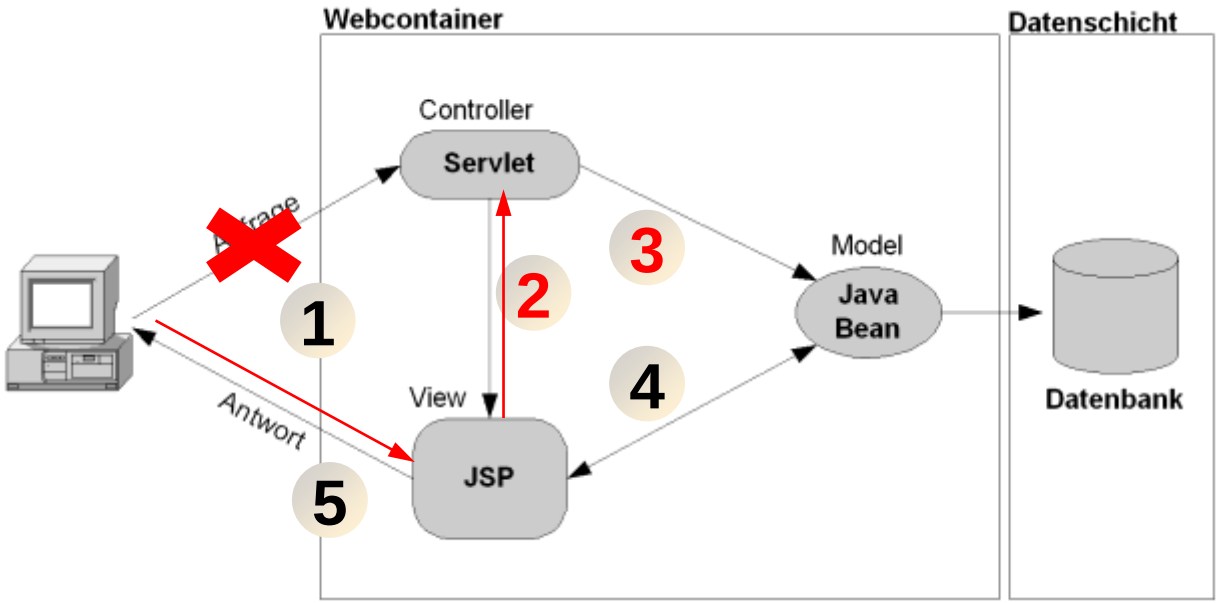


ARS

Model2-Architektur [2|2]

Pull Mode

- Client ruft JSP auf 1
- JSP inkludiert Servlet zum Ausführen von Geschäftslogik 2
  - Manipuliert/erzeugt Model 3
- JSP verwendet Model 4 für die Erzeugung der Ausgabe 5
- Optional: Kein Servlet, sondern Custom Tags für Geschäftslogik



ARS

Syntax [1|5]

JSP pages (`*.jsp`)

- Verarbeitung als Text-Template
- Direkte Ausführung im Webcontainer
- Syntaxelemente: `<% … %>`

JSP documents (`*.jspx`)

- XML-konforme Syntax
  - Ausgabe (z.B. HTML) muss XML-konform sein
- Verarbeitung durch XML-Parser vor Ausführung im Webcontainer
- Vorteile:
  - XML-Validierung des generierten Markups (XHTML)
  - Nutzen von XML-Tools
  - Transformation mit XSLT

ARS

Syntax [2|5]

Syntaxelemente

- Markup (HTML, JSON, RSS…) o.a. Antwort-Formate
- Direktiven (Anweisungen zur Verarbeitung)
- Aktionen (z.B. Include einer anderen JSP - Templating)
- Kommentare
- Ausdrücke (Expressions)
  - Implizit vorhandene Objekte
  - Eigene Objekte
- Java-Code (nicht empfohlen)
  - Scriptlets
  - Deklarationen
- XML Custom Tags (XML-Elemente mit Java-Implementierung)

ARS

Syntax [3|5]

Beispiel – Custom Tags und Expressions

```
<?xml version="1.0"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
          xmlns:c  ="http://java.sun.com/jsp/jstl/core">
   <html>
     <head><title>Hallo Welt</title></head>
     <body>
       <c:if test="${not empty param['name']}">
         <h1>
           Hallo
           <c:out value="${param['name']}"/>
         </h1>
       </c:if>
     </body>
  </html>
</jsp:root>
```

ARS

Syntax [4|5]

Beispiel – Scriptlet (nicht empfohlen)

```
<?xml version="1.0"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page">
<html>  <head>...</head>
  <body>
    <jsp:scriptlet>
      if (request.getParameter("name") == null) {
        out.println("Hello World");
      } else {
        out.println("Hello " + request.getParameter("name"));
      }
      List&lt;String&gt; names = new LinkedList&lt;String&gt;();
    </jsp:scriptlet>
  </body>
</html>
</jsp:root>
```

ARS

Syntax [5|5]

Beispiel – Kommentare

```
<?xml version="1.0"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page">
   <html>
     <head><title>Hallo Welt</title></head>
     <body>
       <!-- XML-Kommentar; wird NICHT übertragen -->
       <jsp:scriptlet>
          /* Java-Kommentar; wird NICHT übertragen */
          // Java-Kommentar; wird NICHT übertragen
       </jsp:scriptlet>
       <jsp:text><![CDATA[<!-- HTML-Kommentar -->]]></jst:text>
     </body>
  </html>
</jsp:root>
```

ARS

Expression Language [1|4]

Motivation

- JSPs in XML-konformer Syntax
- Ausdrücke in für Webentwickler bekannter Syntax

Ergebnis

- Expression Language (EL, JSR-341)
- Syntax ähnlich zu PHP, zu XML kompatibel
  `${expression}`
- Rückgabetyp hängt von Expression ab
  - Objekte und primitive Typen möglich
- Verwendung auch außerhalb JSPs möglich

ARS

Expression Language [2|4]

Verwendung

- Übergabe von Attributen bei Tags

```
<x:myTag attribute="${myExpression}" />
```

- Mehrfache Verwendung, z.B. für Stringkonkatenation

```
<x:myTag attribute="${expr1}${expr2}" />
```

- Blank in der JSP zur direkten Ausgabe

```
<p>Das Ergebnis von 3 mal 5 ist ${3*5}.</p>
```

Auswertung

- Variablen werden in Sichtbarkeitsbereichen gesucht
  - Vom kleinsten zum größten
- Auswertung von EL vor Ausführung des Tags
  - Ergebnis wird als Call-By-Value übergeben

ARS

Expression Language [3|4]

Datentypen (Literale)

- Zahlen

```
${3.25}
```

- Zeichenketten

```
${'This is a string.'}
```

- Arrays

```
${myArray[0]}
```

- Maps

```
${myMap['Zeichenkette']}
```

- Boolesche Literate

```
${true}
```

ARS

Expression Language [4|4]

Weitere Syntaxelemente

- Logische Verknüpfungen und Funktionen
  `${empty list or fn:length(list) gt 5}`
- Lamdba Expressions
  `${((x,y)-> x+y)(4,5)}`
- Bulk Operations
  `${list.stream().filter( x -> x <= 3).map(x -> x * 5).toList()}`
- Deferred Expressions
  `#{…}`
  - Auswertung erst durch Custom Tag (bedingt, mehrfach)

ARS

Kontrollfragen

- Was sind JSPs und wofür werden sie verwendet?
- Wie werden JSPs auf dem Server verarbeitet?
- Worin besteht der Unterschied zwischen JSP pages und JSP documents? Wie unterscheidet der Webcontainer zwischen beiden Formaten?
- Warum ist es nicht empfehlenswert, Skriptlets zu verwenden?