JSP/Servlets vs. PHP: Ein Vergleich für den Einsatz in der Lehre

Johannes Schirm (732571)

Studiengang Medien- und Kommunikationsinformatik

Fakultät Informatik
Hochschule Reutlingen
Alteburgstraße 150
72762 Reutlingen
johannes.schirm@student.reutlingen-university.de

Abstract: Diese Arbeit vergleicht die beiden Webtechnologien PHP und JSP/Servlets in Bezug auf die praktische Lehre im Studium. Es werden sowohl programmiersprachliche als auch technologische Aspekte betrachtet, wobei davon ausgegangen wird, dass die Studierenden Grundkenntnisse lernen und das Prinzip von Webtechnologien verstehen möchten. Um den Vergleich durchzuführen, werden eigene Kernkriterien für die allgemeine Lehre der Informatik benutzt.

1 Die grundlegende Funktionsweise von Webtechnologien

Bevor man zwei Dinge erfolgreich vergleichen kann, ist es ratsam, sich durch eine kurze Wiederholung einen Überblick über deren Gemeinsamkeiten und ihr Umfeld zu verschaffen. Die wichtigste Gemeinsamkeit wäre im Fall dieser Arbeit, dass zwei Webtechnologien verglichen werden. Auch ohne Genaueres über PHP oder JSP/Servlets zu wissen, kann man sich hier vorstellen, dass diese wohl einen Dienst über das Netzwerk zur Verfügung stellen können. Dabei gibt es folgende Rahmenbedingungen: Beide Umgebungen laufen auf einem Rechner, der diese Programme als Prozess in seinem Betriebssystem ausführt. Diese Prozesse greifen auf ein bestimmtes Verzeichnis des Dateisystems dieses Rechners zu. Dort liegen die Dateien mit dem auszuführenden Quellcode. Durch die Anbindung der Quellcodeausführung an eine Anfrage aus dem Netzwerk wird die Technologie erst zu einer Webtechnologie. Die Prozesse „hören“ hierbei auf dem HTTP – Port (80) unter anderem auch auf GET – Anfragen, die ein Dokument aus dem Quellcodeverzeichnis als Antwort gesendet bekommen möchten.

Wichtig ist, was der Rechner auf diese Anfrage zurücksendet. Die Antwort wird bei der Ausführung des Quellcodes durch diesen zusammengesetzt und liefert meistens eine HTML – Datei zurück, die dann in einem Internetbrowser interpretiert werden kann. Innerhalb des Codes ist es aber auch möglich, komplette binäre Dateien zurückzugeben. Unser Quellcode hat in diesen Technologien also die Aufgabe, die Antwort auf die Anfrage gemäß bestimmter Regeln zu erstellen. Der Aufrufer sieht hierbei nichts von der eigentlichen Ausführung. Er bekommt nur eine Zeichenkette auf seine Anfrage zurück.

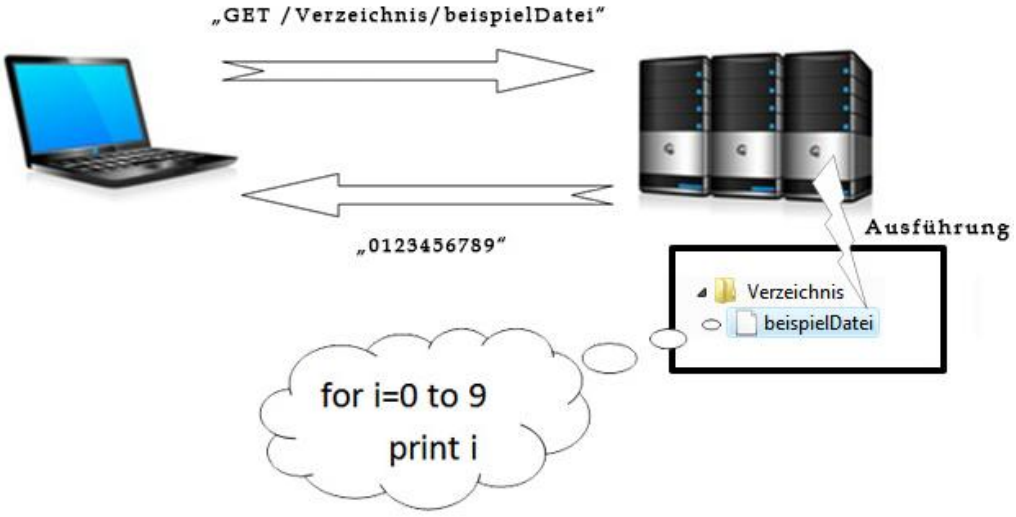

Abbildung 1: Ein Beispiel der Kommunikation zwischen Client und Server.

In Abbildung 1 wird ein Beispiel gezeigt, in welchem der Client eine Anfrage an den Server stellt, um eine bestimmte Datei aus einem bestimmten Verzeichnis zu bekommen. Ohne eine Webtechnologie würde der Server nun die Inhalte der Datei unverändert zurücksenden. Mit einer Webtechnologie aber wird die Datei (wie auch immer) ausgeführt und kann selbst bestimmen, was an den Client gesendet werden soll. Das einfache obige Programm, das für dieses Beispiel in Pseudocode geschrieben wurde, gibt alle Zahlen von 0 bis 9 aus. Diese Zeichenkette wird an den Client zurückgesendet.

2 Eine Vorstellung der zu vergleichenden Technologien

Diese Arbeit interessiert sich für den Vergleich zweier Technologien. In der Lehre können beide eingesetzt werden, doch für welche Situation passt welche am besten? Oder gibt es sogar eine, die der anderen vorzuziehen ist? Um diese Fragen besser beantworten zu können, werden beide Technologien in diesem Kapitel kurz vorgestellt. Zuerst werden JSPs/Servlets betrachtet, die auf der Programmiersprache Java basieren. PHP hingegen ist eine eigenständige Programmiersprache mit eigener Syntax.

Es gibt nicht nur viele Unterschiede zwischen beiden Technologien aufzuzählen, sondern auch einige Gemeinsamkeiten, die sich sowohl auf die Struktur der verwendeten Sprache beziehen, als auch auf die Rahmenbedingungen der Technologien.

2.1 Kurze Vorstellung von JSP/Servlets

Java **S**erver **P**ages und Servlets sind zwei Komponenten, die zur „Java Enterprise Edition“ gehören. Diese Sammlung von Spezifikationen und Technologien aus dem Hause Sun ist (zusammen mit den **A**ctive **S**erver **P**ages von Microsoft) eines der beliebtesten „Betriebssysteme“ für Webanwendungen (vgl. [Sta05, S.15]).

Beide Technologien arbeiten mit der überaus bekannten Programmiersprache Java, die komplett objektorientiert funktioniert. Ein sehr großer Vorteil ist die Kompatibilität mit fast allen Plattformen, da der Quellcode vor der Ausführung in einen Zwischencode (auch Bytecode genannt) übersetzt und dann nur noch interpretiert wird. Das geschieht durch eine virtuelle Maschine, die auf die verschiedenen Betriebssysteme aufsetzt. Schon ganz am Anfang der Geschichte von Java war es den Entwicklern wichtig, eine Sprache für dynamischere Websites zu gestalten (vgl. [Orac1]).

Es fällt auf, dass hier zwei Technologien, die immer gemeinsam genannt werden, mit einer weiteren verglichen werden sollen. Das liegt daran, dass JSPs und Servlets eng miteinander verbunden sind. „Java Server Pages ermöglichen es Ihnen, reguläres HTML mit klassischem Java – Code zu kombinieren […].“ [Sta06, S.37] In der Praxis kann man also im HTML – Code an speziell zu kennzeichnenden Stellen Java programmieren. Java – Code wird allerdings immer von einer **J**ava **V**irtual **M**achine interpretiert, sodass aus der Mischung von HTML und Java wieder eine Java – Klasse erstellt werden muss, damit diese dann in Bytecode umgewandelt werden kann.

Servlets sind genau diese Klassen, die bei einer Anfrage aus den JSPs erzeugt werden. Es ist auch möglich, selbst ein solches Servlet zu schreiben: Das Objekt muss nur die Klasse HttpServlet erweitern. Über entsprechende überschriebene Methoden können dann die verschiedenen Anfragen über das HTTP – Protokoll beantwortet werden. Leider ist das Ausführen der Servlets etwas aufwändiger: Es reicht nicht, die Datei mit dem Quellcode oder die kompilierte Datei mit Bytecode in ein Verzeichnis zu legen. Ausführen kann man ein Servlet erst, wenn es im „Web Deployment Descriptor“ eingetragen ist, den man über eine Datei namens web.xml konfigurieren kann. In dieser Datei werden die URL und der symbolische Name des Servlets auf die Klasse bezogen. Doch dieser Aufwand lohnt sich, denn das Servlet erlaubt den Einblick in seinen Lebenszyklus und kann sogar mehrere Anfragetypen unterscheiden (vgl. [Sta05, S.87]).

Meistens lohnt sich aber ein kluges Zusammenspiel der beiden Varianten, je nachdem, wie das Verhältnis zwischen Programmcode und Gestaltung ist. Als Beispiel könnten zwei Teile eines Webshops dienen: Auf der Seite, die die Produktinfos anzeigt, ist es die Hauptaufgabe, die Details übersichtlich anzuzeigen. Am Anfang werden die Informationen mit ein paar Befehlen aus der Datenhaltung abgerufen. Bestellt man aber etwas, wird am Ende nur ausgegeben, ob der Vorgang geglückt ist. Die Hauptarbeit wäre hier wahrscheinlich eher das Ausführen des Bestellvorganges mit viel Programmcode.

2.2 Kurze Vorstellung von PHP

Nach vielen Jahren der Entwicklung ist die Programmiersprache PHP eine der beliebtesten ihrer Art. Das liegt vor allem auch an der schnellen Prototypenerstellung, die von Anfang an eine der Stärken von PHP war (vgl. [Sch06, S.24]).

1995 wurde ein Open Source Softwareprojekt unter dem Namen PHP/FI von Rasmus Lerdorf geschaffen, das zwei Jahre später zu PHP 3.0 wurde. Eine kleine Sammlung von Perl – Skripten zur Verwaltung von Webauftritten hatte sich in dieser Zeit zu einer erweiterbaren Sprache entwickelt, zu welcher auch viele andere Entwickler ihre Ideen beisteuerten. Bis zur aktuell fünften Version von PHP wurde die „Zend Engine", also der Kern des PHP – Interpreters, mehrere Male verbessert. Dies brachte eine höhere Performanz und viele Details wie Anbindungen an bekannte APIs und neue Sprachkonstrukte wie zum Beispiel die Objektorientierung mit sich. (vgl. [PHP1])

Gerade die Möglichkeit der Objektorientierung seit PHP 5 ist ein Grund für viele Unternehmen, auch große Projekte in dieser Sprache entwickeln zu lassen. Doch nicht nur die Unterstützung des beliebten Paradigmas ist der Grund: „Durch die hohe Verbreitung und den guten Ruf von PHP wird die Skriptsprache […] in letzter Zeit verstärkt in größeren Projekten eingesetzt" [Dop10, S.9]. Das bedeutet, dass PHP mit den im zweiten Kapitel behandelten Technologien in der Zwischenzeit so gut mithält, dass es eine ernstzunehmende Konkurrenz für diese darstellt.

Ähnlich wie bei den JSPs wird ein Bereich im HTML – Dokument, der als PHP – Code interpretiert werden soll, speziell gekennzeichnet. Bei PHP gibt es kein Äquivalent zu den Servlets aus dem zweiten Kapitel, denn der Code muss nicht in eine „Zwischenklasse" umgewandelt werden. PHP lässt sich nämlich auch komplett prozedural programmieren. Trotzdem haben die Technologien sehr viel gemeinsam: Auch die Zend Engine, die den PHP – Code als Maschinenbefehle ausführt, ist eine Virtual Machine ähnlich der Java Virtual Machine (vgl. [Sch06, S.532]). Ein wichtiger Unterschied ist auch zum Beispiel, dass die Variablen durch ein führendes Dollarzeichen gekennzeichnet sind. Dies soll sie von zuvor deklarierten Konstanten unterscheiden, die bei PHP getrennt gehandhabt werden (vgl. [Dop10, S.17]).

3 Kernpunkte aus der Lehre der Informatik

Die beiden Technologien, die in dieser Vertiefung betrachtet werden, sollen nun in Bezug auf die Lehre verglichen werden. Doch was ist in diesem Fall „die Lehre"?
Informatik wird zur heutigen Zeit an verschiedensten Stellen unterrichtet, da sie eine Wissenschaft im Mittelpunkt unserer Informationsgesellschaft ist. „Es ist [außerdem] ihr Hineinwirken in andere Wissenschaftsbereiche, das die Informatik so einzigartig und bedeutend für die Allgemeinbildung macht." [SS11, S.117]

3.1 Die Lehre für das Studium der Informatik

Besonders interessant jedoch dürfte für uns wohl die Betrachtung des Studiums der Informatik sein. Denn hier werden professionelle Informatiker ausgebildet, die im Beruf ein fundiertes und strukturiertes Fachwissen benötigen. Es ist also wichtig, dass diese Lehre gut aufgebaut ist und die Grundkenntnisse richtig vermittelt werden. Dabei richtet sich die Lehre im Studium oft sogar in vielen Aspekten nach aktuellen Technologien, die in der Arbeitswelt und auf dem Markt zurzeit benötigt werden. Das sind meistens spezielle Programmiersprachen oder auch Programmierumgebungen, die sich in der aktuellen und daher modernen Softwareentwicklung etabliert haben.

Jeder Informatiker hat seine Ausbildung wahrscheinlich einmal mit den Grundlagen der Programmierung begonnen. Damit bildet sich aber nicht nur die Fähigkeit aus, ein Problem mit Hilfe einer Programmiersprache zu lösen, sondern auch das Verständnis von Strukturen, die in vielen anderen Zusammenhängen in der Informatik wichtig sind. Es ist also auch ein wichtiges Ziel bei der Lehre der Informatik, die Studierenden auf neue Umgebungen, neue Sprachen und neue Verfahren vorzubereiten. Der Vergleich der Technologien im nächsten Kapitel wird nach genau diesen Aspekten getroffen. Jede Programmiersprache bietet andere Vor- und Nachteile, die entsprechend mehr oder auch weniger hilfreich für eine Person sein können, die gerade das Programmieren lernt.

Man warnt allerdings auch davor, das Programmieren überzubetonen. „Die Krise des Informatikunterrichts anfangs der 90er Jahre ist zu einem guten Teil auf die Erfahrungen mit überzogener Algorithmusorientierung zurückzuführen.“ [Hub07, S.50]

Es gibt in der Praxis zahlreiche Ansätze, die auf verschiedene Arten versuchen, möglichst gut einige der Ideen aus der Informatik zu transportieren: Auf der einen Seite steht der reine Programmierkurs, der sich stark an eine Programmiersprache hält und über die Zeit immer komplexere Konstrukte dieser Sprache einführt (vgl. [SS11, S.289]). Auf der anderen Seite findet sich der „projektorientierte fächerübergreifende Zugang“, welcher sich stark mit der Anwendung der Informatik in anderen Fachgebieten und den Kompetenzen in der Projektarbeit widmet (vgl. [SS11, S.297]). Unterrichtet man Informatik an der Schule (meistens in der Oberstufe), wird man sich auf ein Konzept an irgendeiner Stelle zwischen den beiden Ansätzen festlegen müssen. Im Studium aber werden die vielen Komponenten wie zum Beispiel Programmierung, Projektarbeit, Softwareentwicklung und „außerinformatische Kompetenzen“ [SS11, S.297] meistens in verschiedenen Modulen behandelt. Der Vergleich dieser Arbeit konzentriert sich unter diesem Aspekt auf den Programmierkurs, welcher beispielsweise die Aufgabe hat, die „Grundlagen der prozeduralen und objektorientierten Programmierung“ [MR14, S.9] zu vermitteln und den Studierenden später auch die Fähigkeit verleiht, selbst „einfache Web-Anwendungen […] mit Session-Handling“ [MR14, S.27] entwickeln zu können. Es geht also um den Anfang des Studiums und um die Grundlagen der Informatik.

3.2 Wichtige Inhalte der Lehre

Besonders werden heute objektorientierte Ansätze gelehrt, da dieses Paradigma Vieles in der Entwicklung einfacher macht. Vor allem in sehr großen Softwareprojekten hat es sich bewährt, über unmissverständliche Schnittstellen auf Objekte anderer Entwickler zuzugreifen. Durch diese Kapselung lassen sich sogar ganze Komponenten austauschen und durch (intern) anders funktionierende Komponenten ersetzen. Dieses Konzept hat sich seit mehr als zehn Jahren vor allem auch durch die Programmiersprache Java verbreitet, aus deren Idee auch weitere Produkte entstanden (vgl. [Dop10, S.121]).

Doch die Ziele der informatischen Ausbildung beinhalten nicht nur die Entwicklung von riesigen Softwaresystemen, sondern zu sehr großen Teilen auch die algorithmische Denkweise, die vor allem durch die prozedurale Programmierung gefördert wird. Eine Ursache für Lernprobleme liegt manchmal zum Beispiel in der Unterschätzung des Variablenkonzeptes bei der objektorientierten Modellierung. (vgl. [SS11, S.157]) Dies wird im nächsten Kapitel einen sehr großen Punkt bilden, welcher sich auch mit den Variablenkonzepten in PHP und in Java beschäftigt. Wie man sehen wird, liegt darin auch einer der größten Unterschiede zwischen den beiden Programmiersprachen. Man wird später auch erkennen, dass die Reihenfolge, in welcher die Sprachen gelernt werden, im Normalfall größeren Einfluss auf die gesamte Geschwindigkeit des Lernens haben wird. Dies wird jedoch erst im großen Vergleich ausführlicher erklärt.

Der Umgang mit Dateien benötigt anfänglich das Verständnis der verschiedenen Abstraktionsebenen und der Perspektiven auf die Vorgänge, denn in der einen Sprache liest man mit einem Befehl den Inhalt der Datei aus, während man in der anderen einen Dateistrom öffnet und die Daten byteweise an der Stelle des Dateizeigers bekommt. Diese Unterschiede einordnen zu können, ist zu Beginn nicht leicht. Man muss sich als Student dafür schon die Frage gestellt haben, wie viel jedes der zusammenarbeitenden Module zur Lösung des Problems beiträgt. Für den Entwurf eigener Datenstrukturen ist dieses Verständnis sehr wichtig. Im Allgemeinen sind abstrakte Konzepte bei der Lehre von welchen Themen auch immer für die Lernenden um ein Vielfaches nützlicher als „spezielle, mechanische Vorgehensweisen“, die sich dann nicht mehr auf kompliziertere Problemstellungen im Alltag übertragen lassen (vgl. [SS11, S.81]). Auch hier gilt es aber, das Gleichgewicht zu halten. Abstrakte Konzepte ohne passende Beispiele zu lehren, dürfte nämlich wiederum sehr schwierig werden.

Ein letzter Kernpunkt, der hier genannt werden sollte, ist die Modellierung von Ideen und das systematische Erstellen von Diagrammen, die eigene Ideen anderen Menschen verständlich mitteilbar machen sollen. Ein Grundgedanke wäre: „Wesentliches Lernziel […] ist das Verständnis, dass sich die Informatik bei der Konstruktion ihrer Gegenstände in erheblichem Maße Baukästen unterschiedlicher Form bedient, die zwar klein, aber außerordentlich leistungsfähig und z.T. unüberschaubar komplex sind.“ [SS11, S.144] Ein aktuelles Beispiel wäre die Modellierungssprache UML, die für eine Vielzahl der Perspektiven auf Abläufe und Datenstrukturen eine passende (einheitliche) Darstellungsweise anbietet. Es sollte früh genug vermittelt werden, dass diese Repräsentation von Daten eines Projektes die Verständigung einfacher macht und sogar oft Denkfehler in der Projektidee durch die übersichtliche Darstellung aufweist.

3.3 Herleitung der Vergleichskriterien

Warum wurden gerade diese Themen als Kernpunkte ausgewählt? Empfehlungen für wichtige Lernziele aus dem Buch „Didaktik der Informatik“ ([SS11]) sind unter anderem „die Bewältigung von informatischen Alltagsanforderungen und die Befreiung von Aberglaube auf dem Gebiet der Informatik“ [SS11, S.37]. Dazu gehöre auch die „Beherrschung von Komplexität durch Strukturierung“ [SS11, S.37], was sicher eine Kernkompetenz in der Informatik darstellt. Um nach diesem Kriterium die beiden Technologien zu vergleichen, wird ein exemplarischer Blick auf die **Übertragbarkeit** der gewonnenen Kenntnisse geworfen. Für den anfänglichen Zugang zu einer Programmiersprache werden „schlank[e] und orthogonal[e]“ [SS11, S.290] Sprachen bevorzugt. Ein wichtiger Vergleichspunkt ist also auch die **Einfachheit** einer Sprache. „Die bloße Beherrschung dieser Sprachen, die meist nur wenige, syntaktisch einfache Sprachmittel besitzen, tritt in den Hintergrund.“ [SS11, S.290] Die Studierenden konzentrieren sich so mehr auf die eigentliche Modellierung und Lösung von Problemen (vgl. [SS11, S.290]). Es ist gemäß der Übertragbarkeit wichtig, allgemeine Prinzipien zu vermitteln. Dennoch tragen die Studierenden auch Nutzen davon, wenn zusätzlich aktuelle Technologien mit Bezug zur Industrie vorgestellt werden. Zum Beispiel der „joborientierte[…] Ansatz“ [SS11, S.19] berücksichtigt als Hauptpunkt auch die **Aktualität** der eingesetzten Technologien. Dieser Punkt wird allerdings auch je nach Einrichtung mehr oder weniger betont. Aus der Einfachheit und der Übertragbarkeit lässt sich ein weiterer Punkt ableiten, der auch bei der Lehre von Programmiersprachen eine Rolle spielen dürfte: **Konsistenz**. Einen Lernenden verwirrt jede Inkonsistenz, die er als Spezialfall abtun muss. Einheitliche Konzepte helfen an dieser Stelle viel.

Diese Auswahl von Kernpunkten erhebt keinerlei Anspruch auf Vollständigkeit. Jedoch lässt sich hiermit eine sinnvolle Vergleichsgrundlage schaffen, die aus den folgenden (guten Gewissens ausgesuchten) Kriterien besteht:

- ➔ Es sollten keine „starren Vorgehensweisen“, sondern besser die Konzepte und Werkzeuge der Programmierung vermittelt werden. Damit werden die Studierenden die Idee hinter den Mechanismen von Programmiersprachen auf andere Technologien übertragen und mit diesen intuitiv umgehen können. (**Übertragbarkeit** und gedankliche Flexibilität)
- ➔ Eine Programmiersprache ist sehr hilfreich zum Zeigen und Ausprobieren des Lehrstoffes, aber diese sollte sich nicht in den Mittelpunkt drängen. Eher sollte sie als Vermittlungsebene von informatischen Ideen dienen.
  (**Einfachheit** stellt Sprachspezifisches in den Hintergrund)
- ➔ Es ist nützlich, aktuelle Technologien zu behandeln. Maschinencode zum Beispiel kann einen interessanten Exkurs bilden, ist aber auf dem heutigen Markt nur noch in speziellen Branchen von Nöten.
  (**Aktualität** motiviert und eröffnet Möglichkeiten in der Industrie)
- ➔ Lernt man als Studierender ohne den großen Erfahrungsschatz eine neue Programmiersprache, sollte man sich aus dem bisher Gelernten durch logisches Denken neue Lösungswege erschließen können.
  (**Konsistenz** in einer Sprache steigert Erfolg und Motivation)

4 JSP/Servlets und PHP im Vergleich

Nach drei Kapiteln der Vorbereitung geht es nun um den Vergleich der Technologien. Untergliedert wird dieses Kapitel durch die zu behandelnden Unterschiede der beiden Sprachen, die zuerst wiederholt und danach auf die Lehre im Studium bezogen werden. Wichtig für diese Vergleiche wird noch sein, ob man JSPs und Servlets immer zusammen mit PHP vergleicht, oder ob man nur die zu PHP ähnlichere Technologie (also die **J**ava **S**erver **P**ages) mit PHP vergleicht. Diese Arbeit geht davon aus, dass JSPs und Servlets immer gemeinsam genutzt werden. Das heißt: Es gibt einige Bereiche der Anwendung, die aus Servlets bestehen und andere Bereiche, die aus JSPs bestehen. Eine Problemstellung ist dabei nämlich, dass PHP in der Verwendungsweise mehr den JSPs entspricht. PHP ist zwar um beliebige Bausteine in der Sprache C zu erweitern, jedoch lässt sich das nicht wirklich gut mit den Servlets vergleichen. Die Erweiterung der Sprache ist außerdem nur besonderes Spezialwissen, das man sicher in keinem Grundlagenkurs finden wird. Um eine Lösung für die einheitliche Beurteilung zu finden, wird in den folgenden Kapiteln versucht, JSPs und Servlets immer gemeinsam zu betrachten und auch kurz die Unterschiede beider Java – Technologien zu erwähnen.

4.1 Einrichtung der nötigen Rahmenbedingungen

Aus der Perspektive eines Studierenden, der nun in die Programmierung einsteigen möchte, ist eine der ersten Fragen wahrscheinlich folgende: „Wie kann ich ein Programm dieser Sprache auf meinem Computer ausführen?“ Die vielen theoretischen Ansätze möchten natürlich auch sofort auf dem eigenen Rechner ausprobiert werden. Schon hier lässt sich die Motivation des Studierenden dadurch beeinflussen, wie schnell und übersichtlich die Rahmenbedingungen für die Ausführung erfüllt werden können.

Da beide Technologien Webtechniken sind, ist es beiderseits notwendig, einen Server auf dem System zu installieren, der dann die HTTP – Anfragen an den lokalen Computer entgegennimmt, das verlangte Skript ausführt und zum Schluss eine passende Antwort sendet. Das alles kann er aber nur tun, wenn die jeweilige ausführende Einheit (die virtuelle Maschine) der benutzen Sprache auch auf diesem System vorhanden ist. Für beide Technologien muss also sowohl ein Server (Apache HTTP bzw. Apache Tomcat), als auch die jeweilige Laufzeitumgebung (PHP Engine bzw. Java Runtime Environment) installiert werden. Im Quellverzeichnis liegen dann bei den JSPs und PHP die passenden Skriptdateien. Bei den Servlets muss der Java – Code auf die gebräuchliche Weise kompiliert und die zugehörige Klasse in die web.xml – Datei eingetragen werden.

Wie man an dieser groben Zusammenfassung sieht, ähneln sich die beiden Technologien in Bezug auf die Installation sehr. Beide funktionieren nach ähnlichen Prinzipien, sodass man hier schwer eine Technologie einer anderen vorziehen kann. Als Studierender lernt man deshalb durch beide Varianten, wie die nötigen Rahmenbedingungen für eine Webtechnologie einzurichten sind und wofür die jeweiligen Komponenten arbeiten.

4.2 Überschaubarkeit des Quelltextes

Der Vergleichspunkt für die Beispiele dieses Unterkapitels werden die **Einfachheit** und die **Übertragbarkeit** sein. Es geht um die Basis der Programmiersprachen, die sich aus elementaren Konzepten der prozeduralen und objektorientierten Programmierung bildet. Dabei können natürlich nicht alle Aspekte betrachtet werden, jedoch dienen die aufgeführten Beispiele dazu, ein übertragbares Bild der Sprachen in Bezug auf diesen Vergleichspunkt zu gewinnen.

4.2.1 Basisverständnis der Programmiersprachen

Nachdem alles eingerichtet ist, wird das erste Programm geschrieben und ausgeführt. Wie schnell ist es bei JSPs/Servlets bzw. PHP möglich, die Struktur zu erkennen und zu benutzen? Hier fällt besonders die starke Ähnlichkeit zwischen JSPs und PHP auf:

(PHP)

```php
<?php
    // Das Programm sagt: "Guten Tag!".
    echo "Guten Tag!";
?>
```

(JSP)

```jsp
<%
    // Das Programm sagt: "Guten Tag!".
    out.print("Guten Tag!");
%>
```

(Servlet)

```java
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/GutenTag")
public class GutenTag extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public GutenTag() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Das Programm sagt: "Guten Tag!".
        response.getWriter().print("Guten Tag!");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.doGet(request, response);
    }
}
```

Abbildung 2: Ein einfaches Programm als JSP, Servlet und PHP – Skript

Abbildung 2 zeigt, dass es jeweils nur eine Zeile benötigt, um einen Text auszugeben. Dieser wird in den Ausgabestrom geschrieben, der direkt an den Anfragenden zurückgesendet wird. Sichtbar wird allerdings, dass es nicht sinnvoll wäre, die Programmierung mit den Servlets zu beginnen. Der eigentliche Code, der die Logik des Servlets beschreibt, fängt im Beispiel erst bei Zeile 18 an. Zusätzlich werden die HTTP – Methoden, die man zum Aufruf benutzen kann, unterschieden. Man muss also eine Anfrage über POST zur Funktion für GET weiterleiten, wenn man die Methoden nicht unterscheiden möchte. Es wäre also sicher schwieriger, mit dieser Technik anzufangen.

Wegen der JSPs, die automatisch auf die Objektstruktur von Java abgebildet werden, kommt es aber wieder zum Gleichstand. Würde die Java Technologie nur aus den Servlets bestehen, wäre das einfache Beginnen ein klarer Pluspunkt für PHP. Doch mit dieser Alternative bietet die Java Enterprise Edition ein passendes Äquivalent hierfür. Das einzige, was im obigen JSP – Code die objektorientierte Denkweise benötigen könnte, wäre die Tatsache, dass nicht wie in PHP „allgemein“ etwas ausgegeben wird, sondern dass in den Ausgabestrom „out“ des Dokuments geschrieben wird. Andererseits kommt PHP in diesem einfachen Beispiel sogar ohne einen expliziten Methodenaufruf zurecht. Das ist wunderbar für das anfängliche Verstehen des imperativen Vorgehens.

Schnell werden die Beispiele während des Semesters komplexer. Während die gängigen Basiskonstrukte wie Schleifen, Verzweigungen und Funktionen in beiden Sprachen ungefähr gleich funktionieren, gibt es einen sehr großen syntaktischen Unterschied bei der Benutzung von Variablen. Ein Beispiel dafür wären folgende Quelltexte:

(PHP)

```php
<?php
    // Das Programm gibt das Ergebnis einer Rechnung aus:
    if (true) {
        $ergebnis = 20 * (95 + 1404) / 10;
    }
    echo "Das Ergebnis ist: " + $ergebnis;
?>
```

(JSP)

```jsp
<%
    //Das Programm gibt das Ergebnis einer Rechnung aus:
    int ergebnis = 20 * (95 + 1404) / 10;
    out.print("Das Ergebnis ist: " + ergebnis);
%>
```

(Servlet)

```java
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/Rechnungen")
public class Rechnungen extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Rechnungen() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //Das Programm gibt das Ergebnis einer Rechnung aus:
        int ergebnis = 20 * (95 + 1404) / 10;
        response.getWriter().print("Das Ergebnis ist: " + ergebnis);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.doGet(request, response);
    }
}
```

Abbildung 3: Rechnungen als Beispiel für den Variablenumgang

So würde das wahrscheinlich kein Programmierer schreiben. Abbildung 3 soll allerdings wichtige Unterschiede der Sprachen zeigen. Zum einen ist es in PHP offenbar möglich, in Zeile 6 auf die Variable mit dem Ergebnis der Rechnung zuzugreifen, obwohl diese in einem darüberstehenden Block definiert wurde. Im Beispiel macht der Block keinen Sinn, jedoch passiert es in komplexeren Programmen leicht, dass man Variablen aus untergeordneten Blöcken überschreibt. Zum anderen wird in allen Beispielen aus Abbildung 3 das Ergebnis mit einem Pluszeichen an die Erklärung angehangen und ausgegeben. Wer sich nicht auskennt, weiß nicht, dass das verschiedene Effekte haben wird: Java hängt die Zahl an den String, PHP hingegen rechnet den String auf die Zahl!

Das hat auch viel mit dem nächsten Kapitel zu tun, daher konzentriert sich diese Betrachtung eher auf die syntaktischen Elemente als auf den Typenunterschied. Nur so viel ist darüber schon zu wissen: Im Beispiel wird der String in eine Zahl umgewandelt. Doch was für eine Zahl ist wohl „Das Ergebnis ist: “ nach der Umrechnung? Da der String keine gültige Zahl wie z.B. „1“ oder „-4“ darstellt, wird er einfach zu 0 umgewandelt. PHP hat einen eigenen Operator für das Anhängen von Elementen an einen String, den man an dieser Stelle hätte benutzen müssen, damit keine Rechnung durchgeführt wird. Aus der Perspektive der **Einfachheit** ist es schwer zu beurteilen: Einerseits erkennt man nicht direkt aus der Zuweisung, welchen Typ das Ergebnis der Rechnung nun haben wird. Man muss den zusätzlichen Verknüpfungsoperator kennen, um die Strings korrekt aneinanderzureihen. Andererseits ist es eine Unterstützung für das bildliche Vorstellen: Schreibt man „1“ + 1, dann sieht der Unerfahrene darin eine Rechnung, auch wenn der erste Summand eine Zeichenkette ist. PHP nimmt in diesem Fall freundlicherweise an, dass wohl die Zahlen addiert werden sollen, denn sonst hätte man den Verknüpfungsoperator benutzt. Das kann für Anfänger unter Umständen stundenlanges Suchen nach der Quelle von verwirrenden Ergebnissen bedeuten.

Aus der Perspektive der **Übertragbarkeit** kann man keinen klaren „Sieger“ feststellen. Die Frage ist nämlich, worauf die Kenntnisse übertragen werden. Schaut man auf die kurze, aber breit verzweigte Geschichte der Programmiersprachen, lässt sich kein eindeutig überwiegendes Paradigma erkennen. Es ist lediglich auszusagen, dass Java im Moment am ehesten die aktuelle Welt der Programmiersprachen repräsentiert. Doch auch in PHP lassen sich viele Grundgedanken finden, wenn auch weniger deutlich.

4.2.2 Das Verständnis der Objektorientierung

Beschäftigt man sich dann noch mit der Objektorientierung, bietet es sich oft an, ein kleines Spiel zu programmieren, denn durch das Durchführen von Projekten werden die Studierenden automatisch mit vielen Problemen und deren Lösungen konfrontiert und lernen, mit dem neu erworbenen Wissen umzugehen und Lösungen zu finden.
Der Spieler steht normalerweise in einer solchen Objektstruktur für sich und könnte wie im folgenden Beispiel aus Abbildung 4 als eigene Klasse dargestellt werden. In diesem Beispiel hat der Spieler einen Namen, der länger als zwei Zeichen sein muss und ein Level, das bei der Erzeugung mit 1 initialisiert werden soll. Natürlich kann der Spieler später seinen Level erhöhen, dafür wird dann die levelUp() – Methode zuständig sein. Beide Attribute können auch ausgelesen werden, wogegen das Setzen nicht erlaubt ist. All dies sind die speziellen Bedingungen in unserem Beispiel. Erstellt man eine Klasse, die diese Logik mit dem Prinzip der Kapselung umsetzen soll, müssen dabei die grundlegenden Gedanken der Objektorientierung gedacht werden.

Man sieht daran, dass bestenfalls nicht jeder den Level einfach nach Belieben verändern können sollte. Auch erkennt man, dass es bestimmte Bedingungen für die Existenz eines Objektes gibt, z.B. ein gültiger Name mit mindestens drei Zeichen. Abbildung 4 zeigt nun ein Testprogramm in allen drei Varianten, das einen Spieler erzeugt und den Level dreimal erhöht. Danach wird der Spieler „ausgegeben“, wobei auf die überschriebene toString() – Methode zugegriffen wird, um die Informationen zu formatieren.

(PHP)

```php
<?php
    class Spieler {

        private $level;
        private $name;

        public function __construct($sName) {
            if (!is_string($sName) || strlen($sName) < 3) {
                throw new Exception("Kein gültiger Spielername übergeben.");
            }
            // Nur ein Name mit mehr als drei Zeichen kann gesetzt werden.
            $this->name = $sName;
            // Nach der Erzeugung hat der Spieler Level 1.
            $this->level = 1;
        }

        public function levelUp() {
            // Den Level des Spielers erhöhen:
            $this->level++;
        }

        public function getName() { return $this->name; }

        public function getLevel() { return $this->level; }

        public function __toString() {
            return "Der Spieler " . $this->getName() . " hat Level " . $this->getLevel() . " erreicht.";
        }
    }
    // Nach der gegebenen Klasse wird ein Spieler erzeugt:
    $ich = new Spieler("Johannes");
    // Das Programm soll den Level des Spielers erhöhen und diesen dann ausgeben:
    $ich->levelUp();
    $ich->levelUp();
    $ich->levelUp();
    echo "Aktueller Stand: " . $ich;
?>
```

(JSP)

```jsp
<%
    class Spieler {

        private int level;
        private String name;

        public Spieler(String sName) {
            if (sName == null || sName.length() < 3) {
                throw new RuntimeException("Kein gültiger Spielername übergeben.");
            }
            // Nur ein Name mit mehr als drei Zeichen kann gesetzt werden.
            this.name = sName;
            // Nach der Erzeugung hat der Spieler Level 1.
            this.level = 1;
        }

        public void levelUp() {
            // Den Level des Spielers erhöhen:
            this.level++;
        }

        public String getName() { return this.name; }

        public int getLevel() { return this.level; }

        @Override
        public String toString() {
            return "Der Spieler " + this.getName() + " hat Level " + this.getLevel() + " erreicht.";
        }
    }
    // Nach der gegebenen Klasse wird ein Spieler erzeugt:
    Spieler ich = new Spieler("Johannes");
    // Das Programm soll den Level des Spielers erhöhen und diesen dann ausgeben:
    ich.levelUp();
    ich.levelUp();
    ich.levelUp();
    out.print("Aktueller Stand: " + ich);
%>
```

(Servlet)

```java
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/Spiel")
public class Spiel extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Spiel() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Nach der gegebenen Klasse wird ein Spieler erzeugt:
        Spieler ich = new Spieler("Johannes");
        // Das Programm soll den Level des Spielers erhöhen und diesen dann ausgeben:
        ich.levelUp();
        ich.levelUp();
        ich.levelUp();
        response.getWriter().print("Aktueller Stand: " + ich);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.doGet(request, response);
    }
}
```

Abbildung 4: Ein Spieler – Objekt mit einigen Attributen und Methoden

Die Klasse des Spielers wurde bei PHP und JSP direkt im Anweisungsblock definiert. Das ist bei PHP normal, JSPs und Servlets erlauben jedoch mit den nötigen Verweisen auch den Zugriff auf eine Objektstruktur, die im gleichen Projektverzeichnis liegt. Das Beispiel setzt also voraus, dass neben der Klasse des Servlets die Datei der Spieler – Klasse existiert. Darin befindet sich exakt die Klasse, die auch in der JSP definiert ist.

Die Bewertung zur **Einfachheit** bevorzugt in dieser Sache sicher eher die Sprache Java. Es wäre ratsamer, die grundlegende Objektorientierung in Java zu lehren, da die Syntax weniger verwirrend ist. Nachdem man sich an das Dollarzeichen in PHP gewöhnt hat, muss man über die Pseudovariable $this und einen Pfeiloperator auf Methoden und Attribute des eigenen Objektes zugreifen. Dabei wird das Dollarzeichen von Attributen jedoch wieder weggelassen, da man bei einem solchen Zugriff wiederum nicht mehr zwischen Konstanten und Variablen unterscheiden muss. Zu viele vorgegebene Schlüsselwörter wie z.B. „__construct“ als Namen des Konstruktors machen den Umgang mit der Sprache komplizierter. Die Geschichte der Sprachen verrät aber auch, dass PHP nur um Objektorientierung erweitert wurde und daher Kompatibilität mit alten Verfahrensweisen berücksichtigen musste. Java ist von Grund auf objektorientiert.

4.3 Typsicherheit

Das Verhalten einer Sprache zur Typsicherheit kann sich in mehreren Formen äußern. Eine Sprache kann statisch typisiert sein, sodass schon beim Schreiben des Quelltextes genau angegeben werden muss, welches Element von welchem Typ ist. Sie kann aber auch dynamisch typisiert sein, wobei der Typ jedes Elementes sich während der Programmausführung ändern kann, weil er automatisch abgeleitet wird. Statisch typisierte Sprachen sind auch immer stark typisiert. Das heißt, dass eine Ausnahme ausgelöst wird, sobald Elemente nicht gemäß ihrem Typ verwendet werden. Schwach typisierte Sprachen wandeln das Element dann ohne weitere Behandlung möglichst passend in einen anderen Typ um. (vgl. [Sch06, S.538f])

Ist eine Programmiersprache typsicher, so ist also bereits zur Kompilationszeit für jeden Ausdruck und jede Variable sichergestellt, von welchem Typ dieser bzw. diese ist. Jeder dieser Typen hat einen Wertebereich, sodass man bei allen Variablen und Ausdrücken genau feststellen kann, welche Werte diese annehmen können. (vgl. [JBG15, S.41])

Doch warum sollte man sich die Mühe machen, den Wertebereich der verwendeten Variablen einzugrenzen? „Das Strukturierungskriterium sind die Operationen, die mit den Daten möglich sind. Jeweils alle Datenobjekte, auf denen die gleiche Menge von Operationen möglich ist und die sich strukturell gleich verhalten, fasst man zu einem Datentyp zusammen. So sichert man ab, dass alle Werte eines Datentyps das gleiche operationale Verhalten aufweisen.“ [SS11, S.144f] Praktischerweise kann man die Prüfungen der verwendeten Typen, die deshalb gemacht werden müssen, auch automatisch bei der Kompilation durchführen lassen (vgl. [SS11, S.145]).

Daraus entsteht ein großer Vorteil: Einfache Fehler im Quelltext können sofort gefunden werden. Diese Fehler werden auch als Compilerfehler bezeichnet. Im Gegensatz zu den Laufzeitfehlern bemerkt man diese nämlich nicht erst während der Ausführung des Programmes mit konkreten Daten. Und nicht nur bei der Fehlersuche hilft eine statische Typisierung: Arbeiten viele Menschen an einem Projekt, kann das Gefüge aus Methoden, Klassen und Schnittstellen leicht unübersichtlich werden. Eine klare Definition trägt dann immer mehr zu einer intuitiven Benutzung fremder Elemente bei. Obwohl sich dies alles „unschlagbar“ anhört, kann es trotzdem manchmal hilfreich sein, eine dynamische Typisierung in einer Programmiersprache zu haben. Denn es gibt viele Anwendungsfälle, in welchen der Ersteller des Quellcodes sehr wenige Vorkenntnisse mit sich bringt oder auch schnell zu einem Ergebnis kommen möchte. Natürlich spielt keiner dieser Faktoren in der professionellen Softwareentwicklung eine Rolle, doch diese Arbeit beschäftigt sich ja hauptsächlich mit den Fragen der Lehre: Ist es also hilfreich, das Programmieren ganz zu Anfang in einer Programmiersprache zu beginnen, die dynamisch und schwach typisiert ist?

Java ist eine statisch typisierte Sprache, genauso wie viele andere bekannte Sprachen. Für die Entwicklung eines Projektes und das professionelle Zusammenarbeiten ist eine statische Typisierung auch ohne Frage die bessere Wahl. Möchte man jedoch zum Beispiel einen Einstieg in die einfache prozedurale Programmierung ermöglichen, ohne ausführlich die Zusammenhänge und Schwierigkeiten im Umgang mit strengen Typendeklarationen vorzustellen, bietet PHP dafür eine bessere Möglichkeit. Zusätzlich dürfte es auch einfacher sein, das Skript auszuführen, denn auf beinahe allen mietbaren Webservern ist PHP zurzeit aufgrund der weiten Verbreitung schon fest integriert.

Aber am Ende wird die Entscheidung in Bezug auf diesen Aspekt lediglich nach dem Geschmack des Dozenten getroffen. Denn es ist sowohl möglich, einen leichten Einstieg in PHP zu ermöglichen und später in anderen Sprachen die Typsicherheit einzuführen, als auch direkt mit den Prinzipien der Typsicherheit anzufangen. Diese Arbeit empfiehlt den sofortigen Einstieg in Java, weil dadurch mit etwas mehr Anfangsaufwand ein wichtiges Prinzip geklärt ist, das später auf die meisten anderen Sprachen angewandt werden kann, und man mit diesem Wissen natürlich ebenfalls PHP programmieren kann. Erst Java zu lernen kostet insgesamt auch weniger Zeit, weil die Studierenden sich nicht zwei Mal in (aus deren Sicht) völlig neue Konzepte einfinden müssten. Im Fall der Typsicherheit steht es also beinahe unentschieden, weil man die Lehre der Typisierung auf die jeweils vorliegende Situation anwenden kann, wie man möchte und Zeit hat.

4.4 Konsistente Namensgebung

Sobald man programmiert, muss man sich in einer Umgebung zurechtfinden, in der man für jede zu erledigende Aufgabe bestimmte Konstrukte kennen sollte, die man sich in irgendeiner Form merken muss. Hierbei sind natürlich zum großen Teil auch Erfahrung in der Programmierung und Entwicklungsumgebungen hilfreich. Dennoch ist es ein Qualitätskriterium von Programmiersprachen, eine möglichst konsistente Benennung aller verwendbaren Elemente zu bieten. Gerade für Programmieranfänger wäre es wichtig, sich möglichst schnell die wichtigsten Konzepte der Programmiersprache anzueignen. Das ist auch ein wichtiger Unterschied der beiden Sprachen.

In Java gibt es klare Namenskonventionen, an die sich Hersteller guter Schnittstellen in der Regel halten müssen. Ein gutes Beispiel sind Funktionsnamen: Es hat sich bewährt, die Bezeichnung im „camel case“ oder genauer dem „lowerCamelCase“ zu halten. Im Fall der Methoden ist dann das erste Wort die Aktion, die hinter der Methode steht. Möchte man zum Beispiel einen Wert bekommen, ruft man getWertname() auf. Möchte man ein Element löschen, heißt die Funktion mit großer Wahrscheinlichkeit deleteWertname(). Diese Vorgehensweise ist sehr praktisch, um schnell den gewünschten Methodennamen zu finden. PHP wiederum setzt ein solches Konzept leider nicht konsequent um. Hier gibt es zahlreiche Varianten der Benennung, die von der kompletten Beschreibung in einem Wort (zum Beispiel strtolower() für das Umwandeln eines Strings in Kleinbuchstaben) bis zur Trennung durch einen Unterstrich (zum Beispiel str_replace() für das Ersetzen einer Unterzeichenkette durch eine andere) gehen. Die Namen sind zwar klein geschrieben und sind deshalb im ähnlichen Stil, aber nicht nur für einen Einsteiger kann es sehr umständlich sein, jedes Mal nachlesen zu müssen, wie die benötigte Methode denn nun wirklich heißt. Betrachtet man also die **Konsistenz** der Sprachen, wäre Java aus den oben genannten Gründen die bessere Wahl.

5 Zusammenfassung: Der Vergleich in der Übersicht

Der Vergleich hat für die Hauptkriterien aus Kapitel 3 exemplarische Eindrücke zur jeweiligen Technologie gegeben. Bewertet wurde unter folgenden Annahmen:

- ➔ Studierende der Informatik, also Lernende mit einem professionellen Anspruch, sollen mit einer Webtechnologie die Prinzipe der Webentwicklung verstehen.
- ➔ Hauptsächlich geht es aber um das Erlernen der Programmiergrundlagen, die auch unabhängig von Webtechnologien angewandt werden können.
- ➔ Weniger eine Rolle spielt die einfache Benutzung der Umgebungen und Handhabung von Servern, Laufzeitumgebungen und Ähnlichem.

Das Kriterium der **Aktualität** wurde in diesem Vergleich nicht ausführlich besprochen, da beide Technologien in der heutigen Webentwicklung im Einsatz sind und dort ihre Berechtigung haben. Für den Lernenden spielt dies auch keine größere Rolle.

Die folgenden Unterkapitel sollen die Bewertung beider Programmiersprachen aus Kapitel 4 unter den gerade besprochenen Annahmen zusammenfassen:

5.1 Handhabung

PHP: Auf vielen Mietservern ist PHP schon vorinstalliert. Die einfache Technik ist wenig fehleranfällig und genau richtig für Anfänger.

JSP/Servlets: Der Tomcat – Server zeigt zum Teil kryptische Fehler, z.B. bei falsch registrierten Servlets und ist auch weniger übersichtlich.

5.2 Übertragbarkeit

PHP: Oft werden sehr spezielle Konstrukte verwendet, die in vielen Fällen keine universelle Verwendung finden können. Als eine Webtechnik erkennt man an PHP aber sehr gut die generelle Funktionsweise.

JSP/Servlets: Die Sprache Java bietet viele spezielle Konstrukte, die aber von Anfängern nicht unbedingt benutzt werden müssen. Die Java – Struktur gibt ein gutes Vorbild für das Benutzen von anderen Sprachen.

5.3 Einfachheit

PHP: Stellenweise wird exaktes Wissen benötigt und Fehler sind schwer zu finden. Andererseits hat man durch fehlende Typsicherheit u.a. die Möglichkeit, einen sehr einfachen Einstieg in die Sprache zu bekommen und schnell die ersten eigenen Anwendungen zu erstellen. PHP eignet sich vor allem für prozedurale Übungen.

JSP/Servlets: Hat man einige wenige Grundprinzipien gelernt, lassen sich daraus die meisten Vorgehensweisen, die später noch benötigt werden, schnell herleiten. JSPs sind den Servlets vorzuziehen, da sonst viel Wissen vorgegriffen wird. Denn die Sprache Java arbeitet ja auf komplett objektorientierte Weise.

5.4 Aktualität

PHP: PHP ist eine eher jüngere Sprache, die mit extrem hohen Installationsquoten in den Statistiken vorliegt. Viele Unternehmen führen auch in PHP sehr große Projekte durch, wobei dies viel Disziplin erfordert…

JSP/Servlets: Java wurde schon gerne zu den Entstehungszeiten des weltweiten Internets als Webtechnologie eingesetzt. Neben einer ausgefeilten Umgebung (der Enterprise Edition) gibt es schon viele Entwicklungswerkzeuge.

5.5 Konsistenz

PHP: Leider sind durch viele verschiedene Entwickler an dem OpenSource – Projekt Inkonsistenzen bei der Namensgebung vorhanden.

JSP/Servlets: Durch konsequent erfüllte Konventionen lassen sich Quelltexte normalerweise leicht durchschauen und aus dem Kontext ableiten.

5.6 Die Endbewertung

PHP: Es reicht eine Einführung in die groben syntaktischen Basisfunktionen von PHP, um schon erste Ergebnisse zu erzielen. Jedoch kann viel Wissen nicht direkt auf andere Sprachen bezogen werden. Üblicherweise ist hier keine IDE im Einsatz, daher lernt man mehr über interne Verfahren. PHP eignet sich auch speziell für Prototypenerstellung.

JSP/Servlets: Beinahe alle der Lernziele können durch die Wahl von JSPs und Servlets als Webtechnologie erreicht werden. Der theoretische Teil sollte jedoch etwas umfassender sein, da viele Konventionen von Anfang an erfüllt werden müssen. Servlets sollten z.B. erst zu dem Thema Objektorientierung vorgestellt werden.

Die Empfehlung dieser Arbeit wird klar: Für den Anfängerkurs in der Programmierung eignet sich Java mit einigem Vorsprung besser. Untersucht man außer den oben wiederholten Annahmen auch noch andere, schwankt das Ergebnis stark.

- ➔ Hat man fortgeschrittene Kenntnisse in der Programmierung, versteht man mit PHP einfach und schnell die Funktionsweise von Webtechnologien.
- ➔ Möchte man kein professioneller Programmierer werden, sondern nur in seiner Freizeit etwas Neues lernen, ist PHP sicher leichter zugänglich.
- ➔ Plant der professionelle Informatiker ein Webprojekt, das leicht zu warten und zu verstehen ist, greift er wahrscheinlich zur Enterprise Edition.
- ➔ Will beispielsweise ein Student über die Semesterferien (schnell) ein unkompliziert zu betreibendes Webprojekt entwickeln, wählt er vielleicht PHP.

Doch wie man gemerkt hat, sind die Unterschiede meistens nur minimal. Aus diesem Grund liegt das Hauptbestreben dieser Arbeit nicht primär im Fällen einer Entscheidung zwischen den beiden Technologien, sondern eher in der Bekanntmachung mit beiden, damit sich der aufmerksame Leser am Schluss selbst ein Bild machen und nach seiner eigenen Situation eine Entscheidung treffen kann.

Literaturverzeichnis

[Dop10] F. Dopatka: PHP – Endlich objektorientiert; 1. Auflage; entwickler.press; Frankfurt; 2010; 3-86802-039-7.

[Hub07] P. Hubwieser: Didaktik der Informatik; 3., überarbeitete und erweiterte Auflage; Springer Verlag; Heidelberg; 2007; 3-540-72477-3.

[JBG15] J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley: The Java® Language Specification Java SE 8 Edition; Specification: JSR-337; Oracle America; 2015.

[Job11] F. Jobst: Programmieren in Java; 6., vollständig überarbeitete Auflage; Carl Hanser Verlag; München; 2011; 3-446-41771-7.

[Kra04] J. Krause: Programmieren lernen in PHP 5; 1. Auflage; Carl Hanser Verlag; München; 2004; 3-446-22737-7.

[MR14] Das Modulhandbuch des Studienganges Medien- und Kommunikationsinformatik (Bachelor) der Hochschule Reutlingen, Stand: 07.02.2014.

[Orac1] Offizielle Website von Oracle zur Geschichte der Entwicklung von Java: <http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html>.

[PHP1] Offizielle Website von PHP zur Geschichte: <http://php.net/manual/de/history.php.php>.

[Sch06] G. Schlossnagle: Professionelle PHP 5 – Programmierung; 1. Auflage; Addison-Wesley Verlag; München; 2006; 3-8273-2381-9.

[SS11] S. Schubert, A. Schwill: Didaktik der Informatik; 2. Auflage; Spektrum Akademischer Verlag; Heidelberg; 2011; 3-8274-2652-9.

[Sta05] T. Stark: J2EE – Einstieg für Anspruchsvolle; 1. Auflage; Addison-Wesley Verlag in Kooperation mit Pearson Studium; München; 2005; 3-8273-2184-0.

[Sta06] T. Stark: Jetzt lerne ich J2EE; 1. Auflage; Markt + Technik Verlag; München; 2006; 3-8272-6979-2.