

KOMPONENTEN –

Lernziele



(1) ARCHITEKTUR

Die in einer Web-Architektur auftretenden Komponenten und Systeme sind bekannt und der Übergang von statischen Web-Seiten zu dynamischen Web-Anwendungen kann nachvollzogen werden

(2) JEE

Der Aufbau und die Komponenten des Entwicklungs-Rahmenwerk JEE (Java Enterprise Edition) sind bekannt

(3) SERVLET

Einfache Web-Anwendungen können auf der Grundlage der Java-Servlet-Technologie entwickelt werden und auf dem Anwendungsserver Apache Tomcat verteilt und ausgeführt werden

(4) JSP

JSPs (JavaServer Page) können zur Entwicklung von dynamischen Webseiten verwendet werden

1

03.04.2012

WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Komponenten bilden die zentralen Einheiten zum Aufbau einer Softwarearchitektur und im Speziellen auch der Architektur von Web-Anwendungen. Der erste Schritt von statischen Web-Seiten zu Web-Anwendungen führt zu den dynamischen Web-Seiten. Hierbei wird beim Aufruf einer Web-Seite eine Funktionalität durch Aufruf einer Anwendung auf dem Server ausgeführt.

(2) Die Java Enterprise Edition (JEE) liefert einem Software-Entwickler Basisfunktionalität in Form von sog. JEE-Containern und JEE-Komponenten, durch die sich eine 3-Schichten-Komponentenarchitektur in Java effizienter erstellen lässt.

(3) Servlets sind Java-Programme, die als Teil der JEE-Architektur auf dem Web-Server laufen und als Mittelschicht (engl. middle layer) Anfragen vom Web-Browser auf der einen Seite (Client) und Anfragen von Anwendungen, Web-Services, Datenbanken auf der anderen Seite (Server) entgegennehmen und bearbeiten. Zur Ausführung von Servlets wird ein Anwendungsserver (hier: Tomcat) benötigt, auf dem das entwickelte Servlet verteilt (engl. deploy) und ausgeführt wird.

(4) Eine JavaServer Page (JSP) ist, vereinfacht ausgedrückt, eine HTML-Seite, in die Java-Code eingebettet ist. Auch wenn JSPs vom Aussehen her ganz anders wirken wie Servlets, stellen sie eine spezielle Art der Formulierung eines Servlets dar (eine JSP wird vor deren Ausführung in ein Servlet umgewandelt).

HTML	HyperText Markup Language
JEE	Java Enterprise Edition
JSP	JavaServer Pages

Hauptquellen:

[HB11] Marty Hall and Larry Brown: Core Servlets and JavaServer Pages, Free Online Version of Second Edition, <http://pdf.coreservlets.com/>, 2011.

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.

[Sun-JEE5] Sun: Java EE 5 Tutorial, Sun Microsystems, September 2007.

Definition und Eigenschaften von Komponenten



- (1) Eine Komponente ist ein aus Hardware oder Software bestehender Teil eines Systems
 - (1) Eine Komponente kann sich aus weiteren Komponenten zusammensetzen
- (2) Eigenschaften
 - (1) Einfache Verteilbarkeit
 - (2) Keine persistenten Daten
 - (3) Klar definierte Schnittstellen
 - (4) Keine Kenntnis der internen Struktur erforderlich
 - (5) Vertragliche Vereinbarungen

2

03.04.2012

WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Die Begriffsdefinition stammt aus [IEEE-Glos:18]. Hierin wird auch angemerkt, dass die verwandten Begriffe Einheit (engl. unit) oder Modul (engl. module) häufig synonym genutzt werden.

(2) Die Eigenschaften sind aus [Re03:10] entnommen und gehen auf Definitionen von Szyperski und Goos zurück.

(2.1) Hieraus folgt für Szyperski, dass Komponenten ausführbarer Code darstellen.

Es wird die Integration in andere Umgebungen mittels Komposition ("Third-Party-Composition") und Anpassung ermöglicht.

(2.2) Eine Komponente unterscheidet sich nicht von ihrer Kopie.

(2.2) (2.3) (2.4) Diese Eigenschaften fördern die einfache Verteilbarkeit.

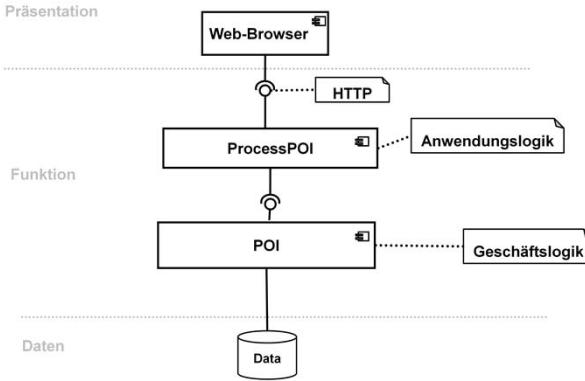
(2.4) Es besteht keine vollständige Black-Box-Sicht, da Werkzeuge durchaus die interne Struktur von Komponenten ausnutzen können.

(2.5) Dieser Aspekt wird insbesondere in den Arbeiten von Ralf Reussner und seiner Forschungsgruppe fokussiert.

[IEEE-Glos] Institute of Electrical and Electronics Engineers: IEEE Standard Glossary of Software Engineering Technology, Std 610-12.-1990, 1990.

[Re03] Ralf Reussner: Component-Based Software Engineering, Vorlesungsfolien, 2003.

KCG: Komponenten zum POI-Management



- (1) ProcessPOI steuert den aus Ein- und Ausgaben bestehenden Vorgang der POI-Abfrage
- (2) POI stellt die eigentliche Funktionalität des Geschäftsobjekts bereit

3

03.04.2012 WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[DM+07]

Zur Modellierung der Softwarearchitektur stellt die Unified Modeling Language (UML) ein Komponentendiagramm bereit. Eine Komponente wird als Rechteck mit einem Stereotyp (Symbol in der rechten oberen Ecke oder alternativ durch Angabe von <<component>>) dargestellt und stellt eine oder mehrere Schnittstellen bereit (eng. provided interface, Ball-Symbol) bzw. nutzt eine oder mehrere Schnittstellen (engl. required interface, Socket-Symbol).

Die Trennung von Anwendungslogik und Geschäftslogik folgt dem Model-View-Controller-Prinzip (MVC), wobei die Anwendungslogik den Controller-Aspekt und die Geschäftslogik den Model-Aspekt abdeckt [:121].

(1) Hierin werden die Schritte realisiert, die zur Ermittlung eines POIs erforderlich sind (Abfrage von Informationen zum POI, Angebot von alternativen POIs, ggf. Verfeinerung der Abfrage, Auswahl des POIs). Diese dynamischen Aspekte der zu entwickelnden Softwarelösung werden in der Analysephase innerhalb von Anwendungsfällen modelliert. Solche Komponenten werden daher auch als Anwendungfallsteuerungskomponenten (engl. use case control components, UCC) bezeichnet.

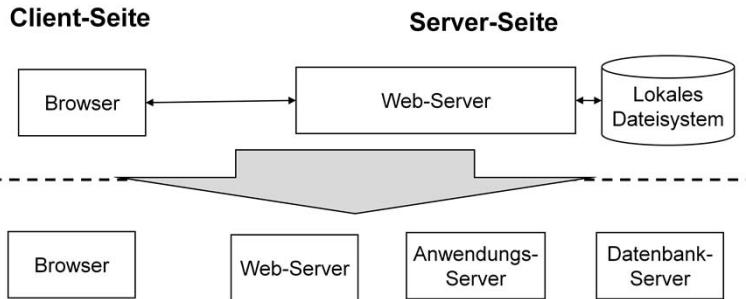
(2) Diese Komponente repräsentiert das Geschäftsobjekt mit den darauf angebotenen Methoden. Im Falle eines POIs sind das neben dem Erzeugen (engl. create), Lesen (engl. read), Ändern (engl. update) und Löschen (engl. delete) insbesondere das in diesem Fall benötigte Suchen (engl. search), wobei verschiedene Suchmethoden (nach Name oder nach der Kategorie oder nach dem nächstgelegenen POI) denkbar sind. Die auf Geschäftsobjekten angebotenen Methoden werden auch als CRUDS-Operationen bezeichnet.

Diese Art von Komponenten werde auch als Geschäftskomponenten (engl. business components, BC) bezeichnet

BC	Business Component
CRUDS	Create, Read, Update, Delete, Search
MVC	Model-View-Controller
POI	PointOfInterest
UCC	Use Case Control
UML	Unified Modeling Language

[DM+07] Naci Dai, Lawrence Mandel, Arthur Ryman: Eclipse Web Tools Platform: Developing Java Web Applications, Addison-Wesley, 2007.

Vom zweistufigen Client-Server-Architekturen zu mehrstufigen Architekturen



- (1) Das Ziel ist eine flexible und effiziente Verarbeitung und Speicherung der Web-Dokumente

[Ta08:589]

Der Kern einer herkömmlichen zweistufigen Web-Anwendung ist ein Prozess auf der Server-Seite, der Zugriff auf ein lokales Dateisystem hat, in dem die Web-Dokumente abgespeichert sind.

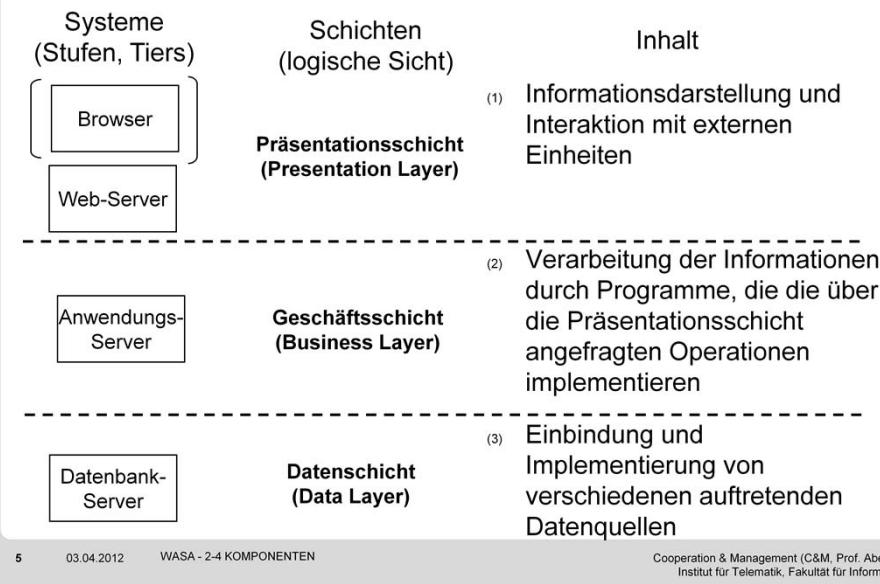
(1) Der Web-Server dient bei der mehrstufigen Architektur dazu, die HTTP-Anfragen entgegenzunehmen und das Ergebnis zurückzugeben. An der eigentlichen Verarbeitung, die im Anwendungs-Server (engl. application server) stattfindet, ist er nicht beteiligt [:593].

Üblich ist heute die Aufteilung der Realisierung einer Web-Site in drei Stufen, eine sog. 3-tier-Architektur.

HTTP HyperText Transfer Protocol

[Ta08] Andrew S. Tanenbaum, Maarten van Steen: Verteilte Systeme – Prinzipien und Paradigmen, Pearson Studium, 2008.

Einordnung der Systeme in die logischen Schichten der Anwendung



[AC+04]

(1) Die externen Einheiten, mit denen die Web-Anwendung kommuniziert, können menschliche Benutzer oder andere Rechensysteme sein [:4]. Entsprechend unterschiedlich kann die Realisierung der Präsentationsschicht sein: neben der üblichen Ausprägung als graphische Benutzeroberfläche kann es sich auch ein Modul zur Formatierung der Daten in ein gefordertes syntaktisches Format sein.

Die Interaktion besteht aus der Übermittlung von Operationen und den Erhalt von Antworten.

Zu beachten ist, dass der Web-Browser dann nicht zur Präsentationsschicht gezählt wird, wenn er vollständig extern und unabhängig zur Web-Anwendung steht. Anders verhält es sich, wenn im Browser präsentationsspezifische Inhalte, z.B. durch Ausführung eines Applets, realisiert werden [:5].

(2) Die Programme werden häufig auch als Services bezeichnet. Ein Beispiel eines solchen "Services" ist ein Programm zur Anmeldung zu einer Schulung.

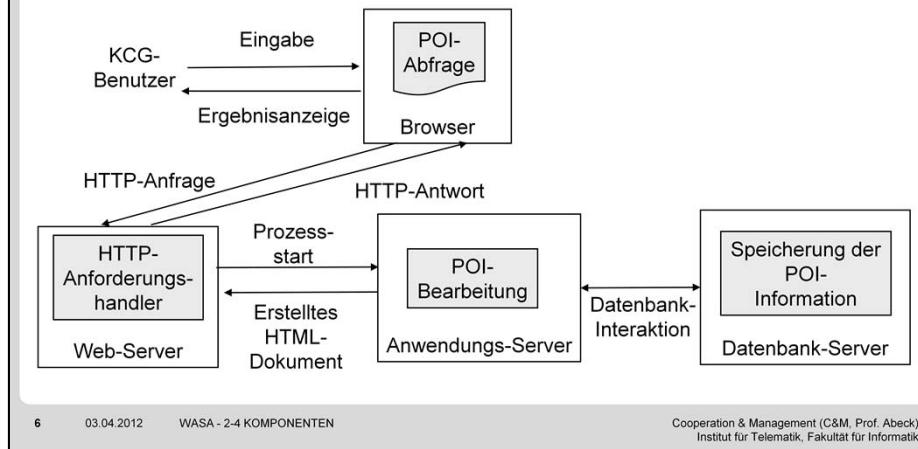
(3) Beispiel verschiedener Datenquellen sind Datenbanken, Dateisysteme oder andere Informationsablagen.

[AC+04] verwendet anstelle von Datenschicht die Bezeichnung "Resource Management Layer", da er bestehende Anwendungssysteme (engl. legacy) in dieser Schicht vorsieht, die im Zuge eines "Bottom-up"-Entwurfs in das Informationssystem einzubinden sind [:9].

[AC+04] Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju: Web Services: Concepts, Architectures, and Applications, Springer, 2004.

KCG: An einer POI-Abfrage beteiligten Systeme

- (1) KCG ermöglicht die Abfrage eines PointOfInterest (POI) zu einem Gegenstand auf dem Campus über einen Standard-Web-Browser
 - (1) Realisiert in Form einer mehrstufigen Architektur



Die in diesem Beispiel auftretenden Systeme, die an der POI-Bearbeitung beteiligt sind, könnten auch als UML-Verteilungsdiagramm (engl. deployment diagram) dargestellt werden.

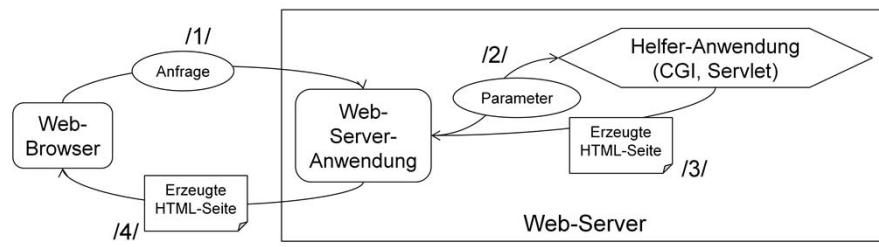
- (1) Der Benutzer des KITCampusGuide (KCG-Benutzer) gibt in seinen Standard-Web-Browser die Web-Adresse ein, über der die Web-Anwendung erreicht werden kann und gibt die notwendigen Informationen zu dem bzw. den POI(s) ein.
- (1.1) Die Einführung eines Anwendungs-Servers bietet sich dann an, wenn die Web-Anwendung komplexere Verarbeitungslogik beinhaltet, an der verschiedene Geschäftsobjekte beteiligt sind. Ein Datenbank-Server dient zur persistenten Speicherung der relevanten Geschäftsinformationen.

Die mehrstufige Architektur wird innerhalb der Software von KCG durch eine entsprechende Komponentenarchitektur abgebildet. Die Architektur umfasst drei Schichten, weshalb es sich bei dem Beispielsystem um eine Ausprägung einer komponentenbasierten Web-Anwendung handelt. Während das auf einer vorhergehenden Seite eingeführte Komponentendiagramm die logische Softwarearchitektur beschreibt, beinhaltet obige Darstellung die Verteilungsarchitektur des Softwaresystems.

KCG	KITCampusGuide
POI	PointOfInterest

Dynamischer Inhalt in Web-Seiten

- (1) Problem: Eine Web-Server-Anwendung kann nur statische Seiten bedienen
- (2) Just-in-Time-Web-Seiten werden dynamisch durch eine geeignete Helfer-Anwendung bei der Bearbeitung der Anfrage erstellt
 - (1) Beispiele dynamischer Seiten: Kataloge, Web-Log, Uhrzeit
- (3) Grober Ablauf:



7

03.04.2012

WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[BS+08:24]

- (1) Die statischen Seiten liegen in einem Verzeichnis, auf das die Web-Server-Anwendung zugreift.
- (2) Hierzu ist u.a. erforderlich, dass die Helper-Anwendung Daten auf dem Server schreiben bzw. speichern kann.
 - (2.1) Beispiel einer Seite mit Uhrzeit-Angabe:
<body> Die aktuelle Zeit ist [insertTimeOnServer] auf dem Server </body>
- (3) Der Ablauf verdeutlicht, dass auf der Seite des Web-Servers eine sog. Helper-Anwendung einen Verarbeitungsvorgang durchführt, um die HTML-Seite dynamisch zu erstellen bzw. zu erzeugen.
 - /1/ Der vom Benutzer eingegebene URL (Uniform Resource Locator) verweist auf eine Helper-Anwendung und nicht auf eine statische Web-Seite.
 - /2/ Die Web-Server-Anwendung erkennt, dass es sich um die Anfrage an eine Helper-Anwendung handelt, startet diese, und übergibt alle Parameter der GET oder POST-Anfrage.
 - (CGI) CGI steht für Common Gateway Interface. CGI-Programme sind meistens Perl-Skripts, können aber auch in C, Python oder PHP geschrieben sein [:27].
 - /3/, /4/ Die dynamische Seite wird produziert und an der Web-Server-Anwendung bereitgestellt, die diese analog zu einer statischen Web-Seite an den Client ausliefert. Außerdem wird die Helper-Anwendung heruntergefahren.

CGI	Common Gateway Interface
URL	Uniform Resource Locator

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

Gründe für die Entwicklung von dynamischen Web-Seiten

- (1) Die Web-Seite basiert auf Daten vom anfragenden Client
 - (1) Beispiel: Name des POI, der ermittelt werden soll
 - (2) Unterscheidung von expliziten und impliziten Benutzerdaten
- (2) Die Web-Seite beinhaltet sich häufig ändernde Daten
 - (1) Beispiel: Auskunft über den sich ändernden Zustand eines POI
 - (2) Periodische Neuerstellung oder bei Vorliegen einer neuen Anfrage
- (3) Die Web-Seite nutzt Informationen aus Datenbanken oder anderen serverseitigen Quellen
 - (1) Beispiel: Informationen zu dem ermittelten POI
 - (2) Three-Tier-Ansatz bietet die größten Vorteile

[HB11:05]

Dynamische Web-Seiten sind Seiten, die (im Gegensatz zu bereits vorliegenden statischen Seiten) zur Laufzeit von einem Programm erstellt werden müssen, bevor sie an den anfragenden Client ausgeliefert werden.

Aufgrund der folgenden Gründe kann eine Seite erst bei Vorliegen der Anfrage und damit spezifisch für diese Anfrage erstellt werden:

- (1) Die Web-Seite kann erst dann durch den Web-Server erstellt werden, wenn er diese Daten vom Client erhalten hat. Diese Information liegt erst zur Laufzeit vor und kann nicht bereits vorab in einer statischen Seite eingefügt werden.
 - (1.1) Alternative (ähnliche) Beispiele sind Resultatseiten von Suchmaschinen oder Bestellbestätigungsseiten.
 - (1.2) Explizite Daten sind z.B. Daten, die in einem Formular (engl. HTML form data) eingetragen werden, implizite Daten sind im HTTP-Kopf enthalten. Hierauf wird später noch genauer eingegangen.
- (2) Während sich im ersten Fall die Web-Seite bei jeder Anfrage ändert, liegt hier der Fall vor, dass sich die Web-Seite in kürzeren Zeitabständen (z.B. periodisch jede n Sekunden oder Minuten) ändert.
 - (2.1) Der POI kann z.B. ein Arbeitsraum sein mit der Zahl der freien Arbeitsplätze als ein dynamisches Zustandsattribut). Alternative Beispiele sind der Wetterbericht oder Nachrichten-Schlagzeilen
 - (2.2) Es kann von Vorteil sein, mit der Neuerstellung bis zum Vorliegen einer neuen Anfrage zu warten, da für den Fall, dass die in der Vergangenheit ausgelieferte Seite noch aktuell ist, ggf. gar keine Neuerstellung erforderlich ist.
- (3) Entspricht aus der Sicht der Web-Seite dem Fall (1), wobei jetzt die Daten nicht von der Client-Seite, sondern von der Server-Seite kommen.
 - (3.1) Die Datenbank dient zur Persistierung der POI-Daten.
 - (3.2) Alternativen:
 - (i) Vollständige Bearbeitung auf Client-Seite (d.h., alle Daten müssten zum Client) ist aus Performance-Gründen ausgeschlossen ("50 Terabyte Applet")
 - (ii) Two-Tier-Ansatz: Applet greift direkt auf die Datenbank zu – geringere Komplexität wegen fehlendem Web-Tier, aber stattdessen geringere Flexibilität und Sicherheit und ggf. geringere Performance
 - (iii) Three-Tier-Ansatz: Einsatz von Caching und Verbindungs-Pooling führt zu einer verbesserten Performance.

[HB11] Marty Hall and Larry Brown: Core Servlets and JavaServer Pages, Free Online Version of Second Edition, <http://pdf.coreservlets.com/>, 2011.

- (1) Servlets und CGI übernehmen beide die Rolle einer Helper-Anwendung
 - (1) Benutzen mit Java (Servlet) und (meist) Perl (CGI) unterschiedliche Programmiersprachen
 - (2) Performance-Vorteile von Java gegenüber Perl
 - (1) Perl-Anfrage wird ggf. als schwergewichtiger Prozess ausgeführt
 - (3) Sicherheits- und Transaktions-Vorteile von Servlets
 - (1) Wegen Einbettung in die JEE-Umgebung

[BS+08:28]

(1) (1.1) Ein Common Gateway Interface (CGI) kann auch als verallgemeinerter Begriff einer Nicht-Java-Helper-Anwendung angesehen werden [:28].

(2.1) Ein Servlet bleibt in der (als einziger schwergewichtiger Prozess bestehenden) Java Virtual Machine (JVM) geladen, weshalb eine Anfrage keinen erneuten Start der JVM oder das Laden von Klassen erfordert.

Zu beachten: Es gibt Web-Server, die den Perl-Prozess zwischen mehreren Anfragen am Laufen halten können.

(3.1) Servlets profitieren im Vergleich zu Perl-CGI-Programmen von den Vorteilen des Java-Rahmenwerks, da sie ein zentraler Bestandteil der hierdurch eingeführten Komponentenarchitektur sind.

CGI	Common Gateway Interface
JVM	Java Virtual Machine

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

LZ ARCHITEKTUR – ÜA KOMPONENTEN UND DYNAMISCHE WEB- SEITEN

- (1) Durch welche Eigenschaften wird die einfache Verteilbarkeit einer Komponente erreicht?
- (2) Welche Gründe führen dazu, dass eine Helper-Anwendung in der Architektur eines Web-basierten Software-Systems vorgesehen wird?
- (3) Sind die folgenden Aussagen zu Helper-Anwendungen richtig oder falsch
 - (1) Eine Helper-Anwendung läuft auf dem Web-Client
 - (2) Eine Helper-Anwendung steht unter der Kontrolle der Web-Server-Anwendung
 - (3) Servlets spielen die Rolle einer Helper-Anwendung
 - (4) Servlets und CGI nutzen die gleiche Programmiersprachen

Java-Plattform Enterprise Edition (JEE)



- (1) Die Java-Plattform Enterprise Edition (Java EE) unterstützt eine einfache und schnelle Entwicklung von flexiblen und sicheren Unternehmensanwendungen
- (2) Vereinfachtes Programmiermodell durch Annotationen von Deployment-Informationen direkt im Quellcode
- (3) Dependency Injections (Abhängigkeitsinjektionen) durch den Container zur automatischen Einbeziehung von benötigten Ressourcen
- (4) Java Persistence API zum Umgang mit relationalen Daten in Enterprise Beans, Webkomponenten und Anwendungs-Clients

11

03.04.2012

WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[Sun-JEE5:41]

Hinweis: JEE5 ist der Nachfolger von J2EE (J2EE Version 1.4 besteht aus (i) Servlets 2.4 spec, (ii) Java Server Pages JSP 2.0 spec, (iii) Enterprise Java Beans EJB 2.1 spec)

(1) Dem Entwickler werden mit Java EE eine Menge von hilfreichen APIs angeboten [Sun-JEE5Tut:41].

(2) Hierdurch werden die Deployment-Deskriptoren optional. Der Vorteil von Annotationen gegenüber Deployment-Deskriptoren besteht darin, dass ein direkter Zusammenhang zwischen den zusätzlichen (Deployment-) Spezifikationen und dem betroffenen Programmelement besteht.

(3) Es können alle JEE-Komponenten um Abhängigkeitsinjektionen (engl. dependency injections, DI) ergänzt werden. Diese Komponenten sind: EJB-Container, Web-Container, Anwendungs-Clients.

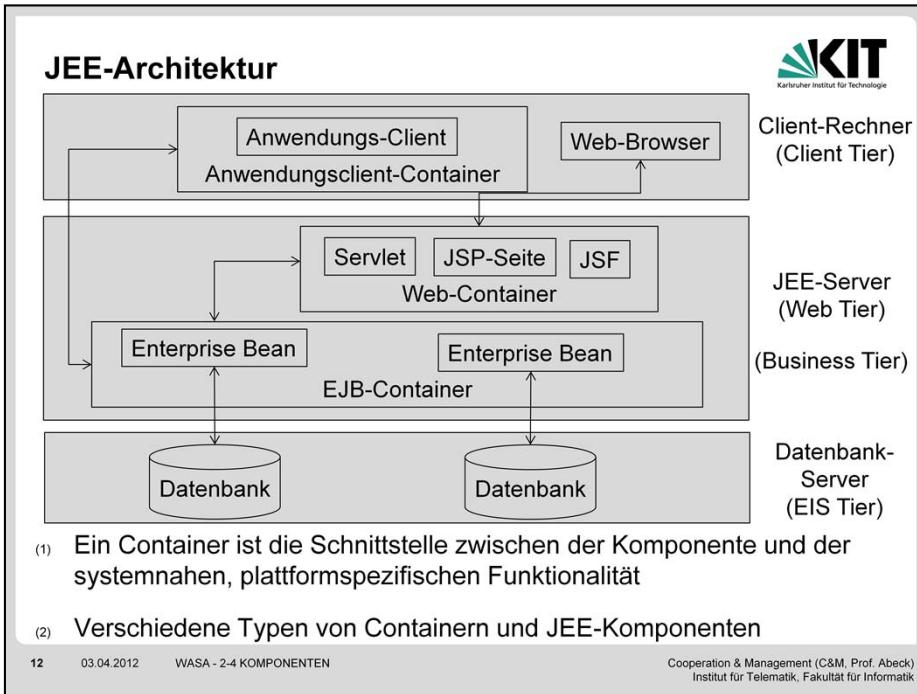
Durch DI wird einer Komponente angegeben, welche anderen Programmteile diese verwenden kann. Es handelt sich um ein Entwurfsmuster, durch das die Abhängigkeit der Komponente zum Framework beherrschbar gemacht wird, indem die Steuerung in das Framework verlagert wird. DI ist somit eine spezielle Form der Umkehrung der Steuerung (engl. inversion of control). Die Umkehrung erfolgt dabei weg von dem eigenen Programm hin zum Framework (Beispiel: UI-Frameworks erreichen durch Event-Handler, dass sich die Steuerung nicht mehr im eigenen Programm, sondern im Framework befindet [Fo04:3]).

(4) Die Java Persistence API ist in der JEE5-Plattform neu dazugekommen und stellt eine Objekt-Relationen-Abbildung bereit.

DI Dependency Injection

[Fo04] Martin Fowler: Inversion of Control Containers and the Dependency Injection Pattern,
<http://martinfowler.com/articles/injection.html>.

[Sun-JEE5] Java EE 5 Tutorial, Sun Microsystems, September 2007.



[SUN-JEE5:43]

JEE-Anwendungen sind 3- bzw. 4-Tier-Anwendungen, die verteilt über die drei Lokationen sind (i) Client-Systeme, (ii) JEE-Serversystem (Web und Business Tier) und (iii) Datenbanksystem oder Legacy-System (EIS Tier, Enterprise Information System)

(1) Das Ziel der Container besteht darin, die Entwickler von diesen systemnahen (engl. low-level) Aufgaben zu entlasten z.B. Transaktions- und Zustandsmanagement, Parallelverarbeitung oder Ressourcenverwaltung.

Die Parallelverarbeitung betrifft die Bewältigung von mehreren Fäden (Multithreading).

(2) Die Ausführung einer JEE-Komponente erfordert

(i) das Zusammenbauen in einem JEE-Modul: Der Prozess des Zusammenbaus (engl. assembly) beinhaltet die Festlegung von Containereinstellungen sowohl für die einzelne Komponente als auch für die gesamte JEE-Anwendung. Die Einstellungen betreffen Sicherheit, Transaktionsmanagement, Namensgebung (Java Naming and Directory Interface, JNDI) und entfernte Konnektivität.

(ii) das Verteilen (engl. deployment) und Ausführen in einem Container

Es lassen sich drei Typen von Containern und sechs Typen von JEE-Komponenten unterscheiden. Eine JEE-Komponente ist dabei eine selbständige funktionale Softwareeinheit, die in eine JEE-Anwendung eingebaut ist und mit anderen Komponenten kommuniziert. JEE-Komponenten sind Java-Klassen, die konform zur JEE-Spezifikation spezielle Anforderungen erfüllen. Erst dadurch ist gewährleistet, dass sich die Komponenten in JEE-Anwendungen einbauen lassen und auf einem JEE-Server verteilen und ausführen lassen.

Die Abbildung zeigt den JEE-Server und die verschiedenen Container-Typen, die in einem JEE-Produkt auftreten können.

(JEE-Server) Der Laufzeitanteil eines JEE-Produkts.

(EJB-Container) Managt die Ausführung der EJBs (Enterprise Java Bean) zu einer JEE-Anwendung.

(Web-Container) Managt die Ausführung der JSP-Seiten, JSFs (JavaServer Face) und Servlets zu einer JEE-Anwendung.

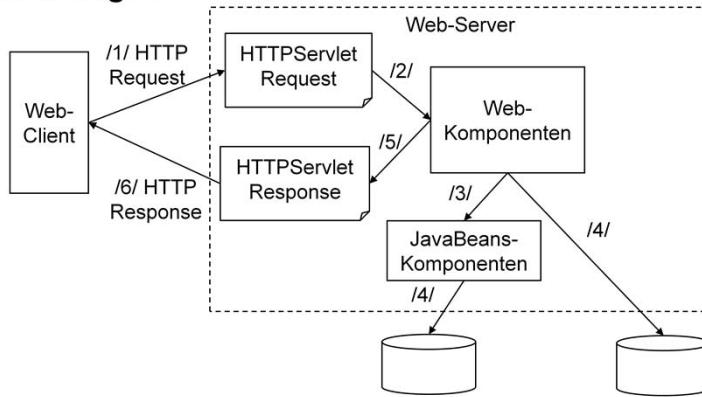
(Anwendungsklienten-Container) Managt die Ausführung von Anwendungsklienten-Komponenten.

(Anwendungs-Container) Besteht aus einem Browser und einem Java-Plug-in zum Management der Ausführung eines Applets.

EIS	Enterprise Information System
EJB	Enterprise Java Bean
JNDI	Java Naming and Directory Interface
JSF	JavaServer Face

[Sun-JEE5] Java EE 5 Tutorial, Sun Microsystems, September 2007.

HTTP-Anfragebearbeitung in Java-Web-Anwendungen



(1) Der Web-Server implementiert die Technologien Java Servlet und Java Server Pages (JSP)

(1) Werden über ein "HttpServletRequest"-Objekt angesprochen und liefern die Antwort in Form eines "HttpServletResponse"-Objekts

[Sun-JEE5:77]

/1/, /2/ Der Web-Server ist dafür zuständig, eine von einem Web-Client kommende HTTP-Anfrage (/1/ HTTP Request) in ein für die Web-Komponente (Servlet oder JSP, /2/) zu verarbeitendes "HttpServletRequest"-Objekt umzuwandeln.

/3/, /4/ Web-Komponenten können mit JavaBeans-Komponenten /3/ oder mit einer Datenbank /4/ interagieren [Sun08:77].

/5/, /6/ Das von der Web-Komponente gelieferte Ergebnis (/5/) ist in ein "HttpServletResponse"-Objekt umzuwandeln und daraus ist eine HTTP-Antwort (/6/ HTTP Response) an den Client zu erzeugen.

[Sun-JEE5] Sun: Java EE 5 Tutorial, Sun Microsystems, September 2007.

JavaBeans-Komponenten

- (1) JEE-Komponenten können (aber müssen nicht) JavaBeans-Komponenten beinhalten
- (2) Anforderungen an JavaBeans-Komponenten
 - (1) Haben Eigenschaften und "get/set"-Methoden, um darauf zuzugreifen
 - (2) Halten sich an Namens- und Entwurfskonventionen, die durch die JavaBeans-Komponentenarchitektur vorgegeben wird
- (3) Das Konzept der JavaBeans wird in verschiedenen Bereichen eingesetzt
 - (1) Graphische Oberfläche
 - (2) Container zur Datenübertragung
 - (3) Datenobjekte

- (1) Die JEE-Spezifikation sieht nicht zwingend JavaBeans-Komponenten als JEE-Komponenten vor. Beispielsweise könnte durch JavaBeans-Komponenten der Datenfluss zwischen einem Anwendungs-Client und einer Komponente auf dem JEE-Server geregelt werden.
- (2) JavaBeans-Komponenten sind typischerweise einfach in Bezug auf deren Entwurf und Implementierung. Eine Anforderung ist die einfache Instanziierbarkeit und Reflexion (Introspektion), d.h. es sollten Informationen zur Klasse bzw. zu deren Instanzen einfach abgefragt werden können.
- (2.1) Die Zugriffsmethoden müssen (wie auch der Standardkonstruktor, d.h. ohne Argumente) öffentlich sein.
- (2.2) Die Komponentenimplementierung muss der "JavaBeans API Specification" entsprechen.
- (3.1) Aus diesem Bereich, der einfachen Instanziierung und Übertragung von GUI-Klassen, ist das Konzept entstanden.
- (3.2) Z.B. Datenfluss zwischen Client und JEE-Serverkomponente
- (3.3) Diese Art der Nutzung führt zu den Enterprise JavaBeans (EJB).

JEE-Clients

- (1) Web-Clients
 - (1) Dynamische Web-Seiten
 - (1) Kann Applets beinhalten
 - (2) Web-Komponenten sind gegenüber Applets zu bevorzugen
 - (2) Web-Browser
- (2) Anwendungs-Clients
 - (1) Stellen eine reiche Benutzerschnittstelle bereit
 - (2) Können direkt auf EJBs (Enterprise Java Beans) zugreifen oder eine HTTP-Verbindung zu Servlets aufbauen
 - (3) Müssen nicht zwingend in Java geschrieben sein

[SUN-JEE5:44]

Es werden zwei Arten von JEE-Clients unterschieden, die Web-Clients und die Anwendungs-Clients.

(1) Web-Clients bestehen aus dynamisch erzeugten Web-Seiten und den diese anzeigenenden Web-Browser.

(1.1) Dynamische Web-Seiten werden von Web-Komponenten erzeugt und können in verschiedenen Markup-Sprachen (z.B. HTML, XML) beschrieben sein.

(1.1.1) Ein Applet ist eine kleine, in Java geschriebene Client-Anwendung, die zur erfolgreichen Ausführung im Browser ein Java Plug-in und möglicherweise eine Sicherheits-Policy-Datei benötigt.

(1.1.2) Web-Komponenten benötigen im Gegensatz zu Applets keine Plugins und keine Sicherheits-Policy-Datei. Außerdem ermöglichen Web-Komponenten einen saubereren und modulareren Entwurf von Client-Anwendungen, indem eine klare Trennung zwischen Anwendungsprogrammierung und Web-Seiten-Entwurf unterstützt wird.

(1.2) Zeigt die vom Server erhaltenen Seiten an.

(2.1) Der Anwendungs-Client ist eine z.B. mittels Swing oder Abstract Window Toolkit (AWT) erstellte graphische Benutzerschnittstelle (GUI).

Die GUI ist reich im Sinne von komplexen Elementen, die durch Markup-Sprachen nicht bereitgestellt werden können.

(2.2) EJBs laufen im Business-Tier, Servlets laufen im Web-Tier (beide Tiers gehören zum mittleren Tier).

(2.3) Auf diesem Weg ist eine Interoperation der JEE-Plattform mit Legacy- und Nicht-Java-Systemen möglich.

AWT	Abstract Window Toolkit
EJB	Enterprise Java Beans (JEE)
GUI	Graphical User Interface

Seerverseitige JEE-Komponenten

(1) Web-Komponenten

- (1) Servlets
- (2) JavaServer Pages (JSP)
- (3) JavaServer Faces (JSF)

(2) Geschäftskomponenten

- (1) Sind EJBs (Enterprise JavaBean), die auf dem Geschäfts-Tier ablaufen
 - (2) Beinhalten eine Logik, die in einer bestimmten Geschäftsdomäne benötigt wird
 - (3) Stehen in Kommunikation mit dem Client und der Datenbank
- ### (3) EIS-Infrastruktursysteme
- (1) Umfassen ERP-Systeme, Großrechner-Transaktionssysteme, Datenbanksysteme und andere Legacy-Systeme

[SUN-JEE5:46]

(1) Keine Web-Komponenten gemäß JEE-Spezifikation sind statische Web-Seiten und Applets (diese können aber beim Zusammensetzen der JEE-Anwendung mit Web-Komponenten gebündelt werden).

(1.1) Servlets sind Java-Klassen, die dynamisch Anfragen verarbeiten und Antworten erzeugen.

(1.2) JSP-Seiten sind textbasierte Dokumente, die als Servlets ausgeführt werden, aber einen natürlicheren Ansatz zur Erstellung des statischen Anteils einer dynamischen Web-Seite bieten.

(1.3) JavaServer Faces setzen auf Servlet- und JSP-Technologie auf und stellen ein Benutzerschnittstellen-Rahmenwerk für Web-Anwendungen bereit.

(2) Durch die Geschäftskomponenten (engl. business component) wird die eigentliche Geschäftslogik der Anwendung erbracht.

(2.1) Enterprise JavaBeans (EJB) sind standardisierte Komponenten innerhalb eines Java-EE-Servers (Java Enterprise Edition). Sie vereinfachen die Entwicklung komplexer mehrschichtiger verteilter Softwaresysteme mittels Java. Sie sind logisch in der Geschäftsschicht (engl. business layer) und physisch im Geschäfts-Tier (engl. business tier) angesiedelt.

(2.2) Ein Beispiel einer Domäne sind die Navigations- und Umweltinformationssysteme, zu denen der KITCampusGuide gehört. Andere Domänen sind Verkehr, Banken, Versicherung.

(2.3) Der Client läuft auf dem Client-Tier und optional auf dem Web-Tier im Falle der Existenz eines serverseitigen Client-Anteils. Die Datenbank läuft auf dem EIS-Tier.

(3) JEE-Anwendungskomponenten benötigen beispielsweise Zugriff zu Unternehmensinfrastruktursystemen, um eine Verbindung zu einer Datenbank herzustellen.

EIS	Enterprise Information System
EJB	Enterprise JavaBean
JSF	JavaServer Faces
JSP	JavaServer Pages

[Sun-JEE5] Sun: Java EE 5 Tutorial, Sun Microsystems, September 2007.

Servlets und Java Server Pages (JSP)

- (1) Servlets sind Java-Klassen, die dynamisch Anfragen bearbeiten und Antworten erstellen
 - (1) Geeignet für serviceorientierte Anwendungen und Steuerungsfunktionen einer präsentationsorientierten Anwendung
 - (2) Stellen die Grundlage aller Web-Anwendungstechnologien dar
- (2) JSP-Seiten sind textbasierte Dokumente, die als Servlets ausgeführt werden
 - (1) Geeignet zur Erzeugung von HTML, XML, WML, SVG
 - (2) JSP Standard Tag Library bietet eine weitere Abstraktionsstufe, durch die die Web-Anwendungsentwicklung weiter beschleunigt wird
- (3) Servlets und JSPs sind Web-Komponenten, die zur Laufzeit durch Dienste eines Web-Containers unterstützt werden
 - (1) Konfigurationsmöglichkeiten zur Veränderung des Verhaltens von bereits installierten und ausgelieferten Web-Komponenten

[SunJEE5:79]

Auch wenn Servlets und JSPs im Prinzip miteinander austauschbar sind, haben beide ihre bevorzugten Einsatzgebiete.

(1) So lassen sich mittels Servlets Webservice-Endpunkte implementieren und Steuerungsfunktionen einer präsentationsorientierten Anwendung

(2) JSPs sind immer dann besser geeignet sind, wenn es um die Erzeugung von Texten, insbesondere in Form von Auszeichnungssprachen (engl. markup languages: HTML, XML, WML) oder SVG (Scalable Vector Graphics) geht.

(3) Die von einem Web-Container bereitgestellten Dienste betreffen u.a. die Anfrageaufbereitung (engl. dispatching), Sicherheit, Nebenläufigkeit, Lebenszyklusmanagement, Sie ermöglichen den Web-Komponenten den Zugriff auf verschiedene APIs (z.B. Namensauflösung, E-Mail) [:79]

Die Konfiguration eines Web-Containers erfolgt in einer XML-Datei, die als Web Application Deployment Descriptor (DD) bezeichnet wird.

DD	Deployment Descriptor
SVG	Scalable Vector Graphics
WML	Wireless Markup Language

[Sun-JEE5] Sun: Java EE 5 Tutorial, Sun Microsystems, September 2007.

Durch Servlets behandelbare Aufgaben

- (1) Der Hauptzweck von Servlets besteht in der Behandlung von HTTP-Operationen
- (2) Sinnvolle durch Servlets zu erbringende Aufgaben sind
 - (1) Sammeln und Prüfen von Benutzereingaben
 - (2) Koordination der Ausgabe
 - (3) Minimale Geschäftslogik
- (3) Aufgaben, die nicht durch Servlets erbracht werden sollten
 - (1) Verarbeitung der Benutzereingaben
 - (2) Erzeugung von dynamischen Webseiteninhalten als Ausgabe
 - (3) Durchführung von umfangreicher Geschäftslogik
 - (1) Ausführung von Datenbankabfragen
 - (2) Behandlung komplexer Web-Transaktionen

[AU01:147]

Der Einsatzbereich von Servlets hat sich in der Vergangenheit geändert, da sie früher umfangreiche Aufgaben übernommen haben, was sie komplex und schwer wartbar gemacht hat.

(1) Servlets dienen dazu, den Informationsaustausch zwischen Client und Server zu unterstützen. Diese Aufgabe erfüllen Servlets gut und effizient (d.h. mit geringen Ressourcenaufwand).

(2) (3) Für Servlets sollte die Devise gelten, klein im Umfang und Komplexität zu bleiben ("best served small"), da ansonsten die Gefahr einer unzureichenden Wartbarkeit der Software entsteht.

[AU01] Khawar Zaman Ahmed, Cary E. Umrysh: Developing Enterprise Java Applications with J2EE and UML, Addison-Wesley, 2001.

Tomcat als Beispiel eines Servlet-Containers



- (1) Tomcat ist eine Standard-Referenzimplementierung für Java-Servlets und JSP
- (2) Einsatzformen
 - (1) Als eigenständiger Web-Server
 - (2) Eingebettet in einen Web-Server
 - (1) z.B. Apache, Microsoft IIS
- (3) Eine übliche Kombination ist Apache Tomcat
 - (1) Apache ist der HTTP-Web-Server
 - (2) Tomcat ist der Web-Container

19

03.04.2012

WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[Li-Tomcat:1] , [BS+08:65]

(1) Entwickelt wurde Tomcat von Apache (www.apache.org) [Li-Tomcat:1].

(2.1) Der Einsatz von Tomcat als eigenständiger Web-Server (HTTP-Server) ist eine eher seltene Einsatzform, da er hinsichtlich Stabilität und Funktionalität unzureichend ist [BS+08:65].

(1.2.1) IIS steht für Internet Information Server

(3) Eine übliche Kombination ist die Verwendung des Web-Servers Apache, was durch den Begriff "Apache Tomcat" zum Ausdruck gebracht wird.

(3.2) Tomcat ist deshalb kein vollständiger JEE-Anwendungs-Server. Zu einem JEE-Anwendungs-Server gehören neben dem Web- und EJB-Container u.a. auch eine JNDI- und JMS-Implementierung. Beispiele für vollständige Anwendungsserver sind JBoss Application Server und IBM WebSphere.

JMS	Java Messaging Service
JNDI	Java Naming and Directory Interface
IIS	Internet Information Server (Microsoft)

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

[Li-Tomcat] Y. Daniel Liang: Installing and Configuring JDK 1.6, Supplement for Introduction to Java Programming, <http://cs.armstrong.edu/liang/intro7e/supplement/Supplement6eTomcat5.5.9.pdf>.

LZ JEE

ÜA AUFBAU UND BESTANDTEILE

- (1) Wozu dienen die von JEE im Quellcode unterstützten Annotationen?
- (2) Die nachfolgenden Begriffe sind sinnvoll anzugeordnen
 - (1) Business, EIS, Client, Web
- (3) Was ist ein Container und welche Arten treten in der JEE-Architektur auf?
- (4) Welche JEE-Komponenten benötigen einen Web-Container?
- (5) Was sind die Gemeinsamkeiten und was sind die Unterschiede von Servlets und JSPs?
- (6) Was wird durch Tomcat bereitgestellt?

Java-Servlets

- (1) Java-Servlets bieten eine portable Technologie zur Erzeugung von dynamischen, benutzerorientierten Web-Inhalten
 - (1) Sind Java-Programme, die wie CGI-Programme aufgrund einer Anfrage eines Web-Browsers ausgeführt werden
 - (2) Laufen innerhalb eines Web-Containers (Servlet-Container)
- (2) Servlet-Klasse
 - (1) Eine Java-Klasse, die die Fähigkeiten eines Servers und speziell eines Web-Servers erweitert
 - (1) Allgemeine und HTTP-spezifische Servlet-Klassen
 - (2) Alle Servlets müssen das Interface "Servlet" implementieren
 - (2) Die Klasse "HttpServlet" stellt Methoden zur Behandlung von HTTP-spezifischen Diensten zur Verfügung

(1) Bereits vor den Servlets wurde dieses Ziel mit dem Common Gateway Interface (CGI-Skripte) verfolgt. Der Nachteil von CGI sind insbesondere die Plattformabhängigkeit, die mangelnde Skalierbarkeit [Sun-JEE5:99] und Performance-Probleme [Liang05:1013].

(1.2) Der Servlet-Container wird auch als "Servlet-Server" oder "Servlet-Engine" bezeichnet [Li05:1014].

(2.1) Grundsätzlich können Servlets für beliebige, gemäß einem Request-Response-Programmiermodell arbeitende Serversysteme genutzt werden. Servlets können also nicht nur zur Behandlung von HTTP, sondern auch von FTP genutzt werden (aber: praktisch alle Einsatzformen von Servlets beziehen sich auf HTTP (-> 99,9..%) , und hiervon die meisten auf die GET-Methode).

(2.1.1) "javax.servlet" und "javax.servlet.http" stellen Schnittstellen und Klassen zur Programmierung von Servlets bereit.

(2.2) Beispiele solcher Methoden sind "doGet()" oder "doPost()".

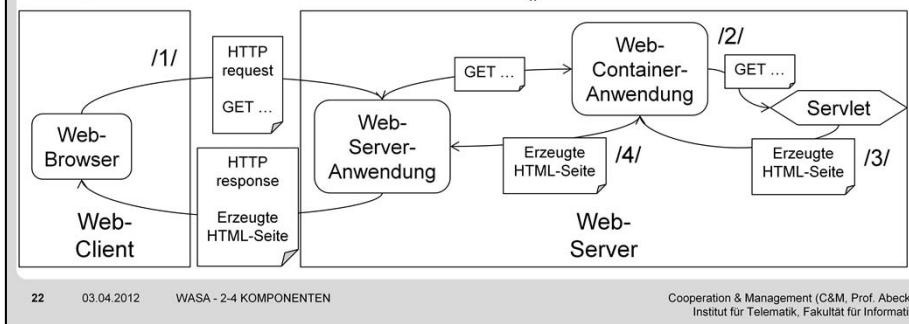
CGI Common Gateway Interface

[Li05] Y. Daniel Liang: Introduction to Java Programming; Pearson Prentice Hall, 2005.

[Sun-JEE5Tut] Java EE 5 Tutorial, Sun Microsystems, September 2007.

Servlet-Container

- (1) Eine Java-Anwendung, in der die Servlets kontrolliert ablaufen
 - (1) Tomcat ist ein Beispiel eines Containers
- (2) Eine Web-Server-Anwendung übergibt die Servlet-Anfrage an den Container, an den das Servlet ausgeliefert ist
- (3) Der Container leitet die HTTP-Operation an das Servlet weiter
 - (1) Aufruf der Servlet-Methode "service()"



[BS+08:39]

Der Ablauf wurde in ähnlicher Form bereits auf einer der vorderen Seiten dieser Kurseinheit behandelt. Jetzt wird eine Verfeinerung hinsichtlich der Umsetzung des Ablaufs in Java vorgenommen, indem entsprechende Java-Objekte und -Methoden eingeführt werden.

(1) Servlets haben keine "main()"-Methode und können daher nicht ohne Vorhandensein eines Servlet-Containers ausgeführt werden.

(1.1) Tomcat ist ein als Open Source verfügbarer Servlet-Container.

Üblicherweise wird Tomcat mit dem Web-Server "Apache" kombiniert, was unter der Bezeichnung "Apache Tomcat" bekannt ist. Apache und Tomcat stammen beide von der Organisation Apache Foundation.

(2) D.h., die Anfrage an das Servlet wird nicht direkt an das Servlet übergeben.

Das Servlet wird in einen Container übertragen bzw. verteilt (engl. deploy).

(3) Operationen sind z.B. "HTTP GET" oder "HTTP POST".

(3.1) Die dazugehörigen Methoden sind "doGet()" oder "doPost()".

Durch die Abbildung wird der Ablauf, der zwischen den auf dem Web-Client und dem Web-Server laufenden Komponenten erfolgt, verdeutlicht.

/1/ Der Benutzer klickt auf einen Link, der eine URL auf ein Servlet anstelle einer statischen Web-Seite beinhaltet [42].

/2/ Der Container erzeugt zwei Objekte (= Instanzen von Java-Klassen), ein "HttpServletRequest" und ein "HttpServletResponse". Diese Objekte werden dann vom Container zu einem zuvor erzeugten Thread des gemäß URL identifizierten Servlets geschickt.

Der Container ruft die "service()"-Methode des Servlets auf, die ihrerseits entweder die "doGet()"- oder die "doPost()"-Methode aufruft. Da es sich beim "HTTP request" um ein GET handelt, ist das hier die "doGet()"-Methode.

/3/ Die "doGet()"-Methode erzeugt die dynamische Seite und fügt diese in das Antwort-Objekt.

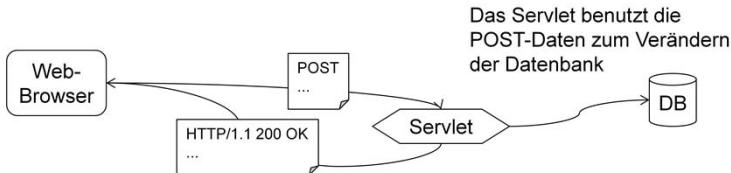
/4/ Der Container konvertiert das Antwort-Objekt in ein "HTTP response" und die "HttpServletRequest"- und "HttpServletResponse"-Objekte werden gelöscht.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

HTTP GET und HTTP POST

(1) Wesentliche Unterschiede

- (1) POST besitzt im Gegensatz zu GET einen Body
- (2) Parameter werden in GET in der Befehlszeile, bei POST im Body übertragen
 - (1) Sicherheitsproblem und Längenbegrenzung bei GET
 - (3) GET-Anfragen können im Gegensatz zu POST-Anfragen mit einem Lesezeichen versehen werden
 - (4) GET-Anfragen sind idempotent, POST-Anfragen hingegen nicht



23

03.04.2012

WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[BS+08:111]

Die Wahl der HTTP-Methode entscheidet darüber, ob die "service()"-Methode die (vom Web-Anwendungsentwickler zu überschreibende) Methode "doGet()" oder "doPost()" des Servlets aufruft.

(1.1) Beispiel:

*GET <Weg zur Ressource auf dem Webserver>?<par>=<Wert>&<par>=<Wert>&.... HTTP/1.1
<Anfrage-Kopf> (-> Host, User-Agent, Accept-Language, Accept-Charset, Keep-Alive, ...)*

*-----
POST <Weg zur Ressource auf dem Webserver>? HTTP/1.1*

*<Anfrage-Kopf>
<Nachrichtenkörper> (-> <par>=<Wert>&<par>=<Wert>&....)*

(1.2) Siehe obiges Beispiel

(1.3) Ist von hoher praktische Relevanz. Beispiel: Eine Suchseite, die von einem Benutzer mit Suchbegriffen gefüllt wird und auf die er nach einer gewissen Zeit zurückkehren möchte.

(1.4) Idempotenz besagt, dass die Methode beliebig häufig ohne Erzeugung von Seiteneffekten ausgeführt werden kann [:116].

Gemäß Spezifikation können die POST-Daten für eine Transaktion genutzt werden, die nicht rückgängig gemacht werden kann.

Auch wenn die Spezifikation "HTTP GET" grundsätzlich als idempotent annimmt, kann die dazugehörige Servlet-Methode "doGet()" mit Seiteneffekten und damit nicht-idempotent implementiert werden [:116].

(DB) Die Frage der Idempotenz bezieht sich auf den Zustand der Datenbank vor und nach Ausführung der entsprechenden HTTP-Operation.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

Zur Verwendung der beiden HTTP-Operationen

- (1) Der Klick auf einen einfachen Hyperlink erzeugt immer ein HTTP GET
- (2) Die in ein HTML-Formular eingetragenen Daten werden üblicherweise durch ein HTTP POST gesendet
 - (1) Muss explizit angegeben werden, weil GET die Voreinstellung ist
 - (1) Im Beispiel:
`<form method="POST" action="SelectPOI.do">`



24

03.04.2012

WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[BS+08:118]

(1) Die GET-Operation ist die übliche und voreingestellte (engl. default) Operation, die ein Web-Browser verwendet.

(2) Ein HTML-Formular (engl. HTML form) wird durch den HTML-Tag "<form>" identifiziert und stellt für den Aufbau von Formularen benötigte Sprachelemente wie z.B. Eingabe- und Auswahlboxen) zur Verfügung.

(<select>) Dieser Tag dient dazu, eine Auswahl anzubieten. Ergänzende Attribute geben z.B. den Namen des Parameters an, der den ausgewählten Wert zugewiesen bekommt (im Beispiel: name = "poiName")

(2.1) Falls das Attribut "method" (hier mit der Zuweisung des Werts "=POST") zum Element "form" nicht angegeben wird, erzeugt der Browser automatisch ein "HTTP GET".

(HTML-Formular) Nutzt die zum Aufbau eines Formulars zur Verfügung stehenden <form>-HTML-Elemente, um ein Formular zu erstellen, aus dem ein Benutzer aus einer Liste von POIs ein POI auszuwählen.

(HTTP POST Request) Das ist die vom Browser erzeugte und zum Server geschickte Anfrage, sobald der Benutzer das Formular ausgefüllt und abgeschickt hat. Im Beispiel hat der Benutzer das "Forum" aus der Liste der im HTML-Formular angegebenen POIs ausgewählt.

(Servlet-Klasse) Die zur Klasse "HttpServletRequest" gehörende Methode "getParameter" wird der Wert (hier "Forum") des Parameters (hier: "poiName") übergeben. Der Name des Parameters ist im Formular durch entsprechende Zuweisung des zum "select"-Element gehörenden Attributs "name" festgelegt.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

LZ SERVLET – ÜA HTTP

- (1) Wie hängen HTTP und Servlets zusammen?
- (2) Welche Teile einer Web-Anwendung arbeiten bei einer HTTP-Anfrage eines Servlets wie zusammen?
- (3) Welche HTTP-Operationen können beim Aufruf von Servlets genutzt werden?
- (4) Wann wird HTTP GET und wann wird HTTP POST genutzt?

Beispiel: Java-Code zu "FirstServlet"

- (1) Ein (Web-) Servlet ist eine Unterklasse der "HttpServlet"-Klasse
 - (1) Implementierung des Servlets erfolgt durch Überschreiben der entsprechenden Methoden
- (2) Java-Code eines einfachen Servlets "FirstServlet"

```
1. import javax.servlet.*;
2. import javax.servlet.http.*;
3. import java.io.*;
4.
5. public class FirstServlet extends HttpServlet {
6.     public void doGet(HttpServletRequest request,
7.                         HttpServletResponse response)
8.             throws ServletException, java.io.IOException {
9.         PrintWriter out = response.getWriter();
10.        java.util.Date today = new java.util.Date();
11.        out.println("<html>" + "<body>" +
12.                   "<h1 align=center>My First Servlet</h1>" +
13.                   "<br>" + today + "</body>" + "</html>");
14.    } // doGet
15. } // FirstServlet
```

[BS+08:44]

(1) Die Unterklassen- bzw. Vererbungsbeziehung wird in Java durch "extends" ausgedrückt.

(5. extends) Die Klasse "FirstServlet" erweitert (erbt von, verfeinert) die im importierten Paket "javax.servlet.http" /2./ enthaltene Klasse "HttpServlet".

Praktisch alle Servlets (laut [BS+08:44] 99,99999%) sind HttpServlets.

(1.1) Im Beispiel wird die Methode "doGet()" überschrieben.

(6. doGet) Diese Methode wird vom Servlet-Container aufgerufen, wenn ein Web-Browser eine HTTP-GET-Methode an das Servlet schickt. Hierzu muss im Browser der URL des Servlets ("http://<rechner>:<port>/<Verzeichnispfad>/FirstServlet") angegeben werden.

Beispiel: "http://localhost:8080/examples/servlet/FirstServlet"

Nahezu alle Servlets überschreiben entweder die Methode "doGet()" oder die Methode "doPost()".

(6. request, 7. response) Referenzen auf die Objekte, die vom Container erzeugt wurden.

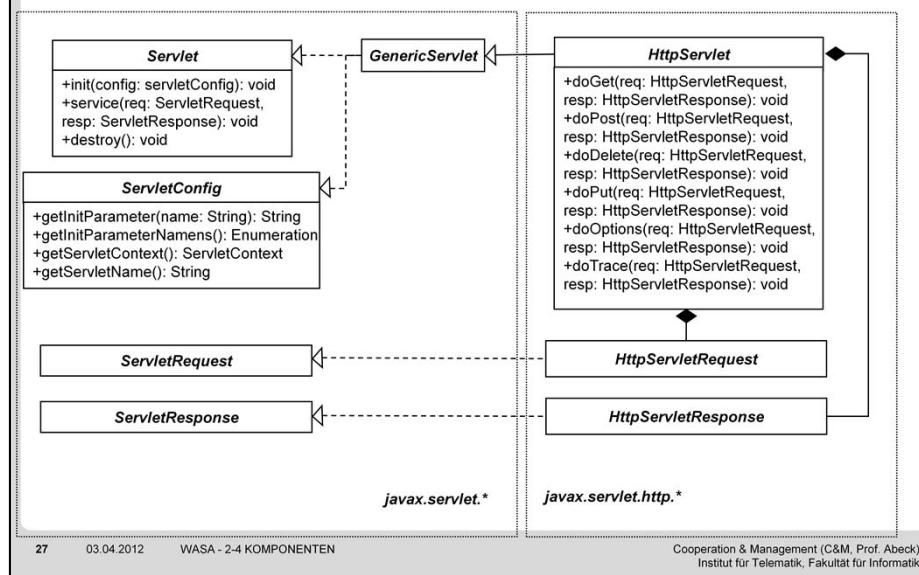
(9. response.getWriter) Vom "response"-Objekt, das der Container dem Servlet übergibt, kann man einen "PrintWriter" erhalten.

(11. out.println) Über das "PrintWriter"-Objekt "out" lässt sich HTML-Text in das "response"-Objekt schreiben. Der HTML-Text ist dadurch im "HttpServletResponse"-Objekt "response" enthalten und wird in einer HTTP GET-Antwortnachricht an den Web-Client gesendet.

Hinweis: Es kann dem ausgebenden Browser explizit angezeigt werden, dass der Inhalt in der Sprache HTML verfasst ist. Einfacher Text wird beispielsweise durch "text/plain", GIF-Bilder durch "images/gif" angegeben.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

Die Servlet-API: Servlet, GenericServlet, HttpServlet



[Li05:1016]

Die Servlet-API stellt die Interfaces und Klassen zur Verfügung, um Servlets programmieren zu können.

(`javax.servlet.*`, `javax.servlet.http.*`) Die zwei Pakete der Servlet-API. Die Klassen und Interfaces des "javax.servlet.http"-Pakets sind abgeleitet und ergänzen die allgemeinen Klassen und Interfaces des ersten Pakets um HTTP-spezifische Aspekte.

Unterschied zwischen Interface und abstrakter Klasse: Ein Interface ist ein klassenähnliches Konstrukt, das ausschließlich Konstanten und abstrakte Methoden enthält. Eine abstrakte Klasse ist eine Klasse, die nicht instantiiert werden kann und ist daher einem Interface ähnlich, kann aber zusätzlich Variablen und konkrete Methoden enthalten [Li05:323].

(Beziehungen) [Ke06]

- Generalisierung zwischen der abstrakten Klasse "GenericServlet" (Oberklasse) und der abstrakten Klasse "HttpServlet" (Unterklasse) [:75].
- Realisierung (gestrichelt, geschlossene nicht ausgefüllte Pfeilspitze) zwischen dem Interface "ServletRequest" (Supplier) und dem Interface "HttpServletRequest" (Client) [:74].
- Es besteht eine Kompositionsbziehung zwischen "HttpServlet" (Ganzes) und "HttpServletRequest" (Teil) sowie "HttpServletResponse" (weiterer Teil) [:69]. Die Servlet-Klasse setzt sich aus den zwei Teilen "ServletRequest" und "ServletResponse" zusammen (und die Lebenszeit der Teile hängt unmittelbar mit der Lebenszeit des Ganzen zusammen).

(Servlet) Ein Interface, das die Methoden definiert, die von allen Servlets implementiert werden müssen. Diese Methoden sind:

- `init()`: wird ausschließlich bei der Erzeugung des Servlets aufgerufen. Hierzu nutzt die Methode das Interface "ServletConfig", das Informationen über den Webserver bereitstellt.
- `service()`: wird jedes Mal aufgerufen, wenn der Server einen Anfrage an das Servlet erhält.
- `destroy()`: wird nach einem Timeout oder beim Herunterfahren des Web-Servers aufgerufen.

(HttpServlet) Die Klasse ist abstrakt, obwohl alle ihre Methoden nicht abstrakt, also konkret sind. Der Grund ist, dass ein Servlet ausschließlich von einer generalisierten (engl. extended) Klasse erzeugt werden sollte.

Die Methoden zur Bearbeitung eines HTTP-Request implementieren die Methode "service()" der Klasse "GenericServlet".

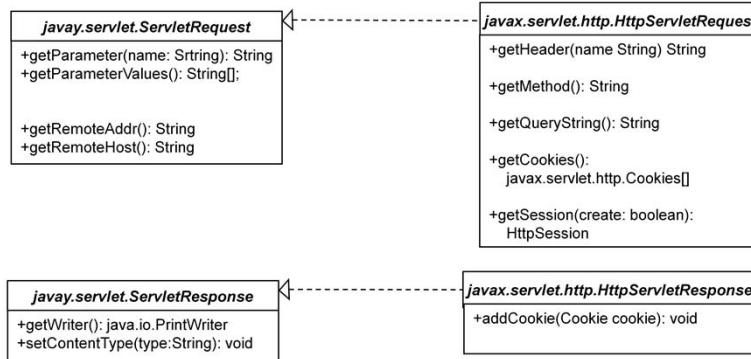
Die ersten vier aufgelisteten Methoden werden gemäß der erhaltenen Anfrage (GET, POST, DELETE, PUT) aufgerufen und sind durch geeignete Methoden zu überschreiben. In "HttpServlet" sind die Methoden leer, d.h. es erfolgt keine Reaktion.

Die zwei folgenden Methoden "doOptions()" und "doTrace()" beinhalten hingegen Funktionalität und werden üblicherweise nicht überschrieben.

(ServletRequest, ServletResponse, HttpServletRequest, HttpServletResponse) Diese Klassen werden nachfolgend behandelt.

Die Servlet-API: "[Http]ServletRequest", "[Http]ServletResponse"

- (1) Die von einem Anfrage- bzw. Antwort-Objekt zu implementierenden Methoden werden durch Interfaces festgelegt
 - (1) "HttpServletRequest" und "HttpServletResponse"
 - (2) Implementiert durch den Container



28

03.04.2012

WASA - 2.4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[BS+08:106], [Li05:1018]

- (1) Das Anfrage-Objekt und das Antwort-Objekt sind der Schlüssel zu allem. Außerdem sind sie die Argumente zur Methode "service()" [BS+08:106].
 - (1.1) Diese Interfaces "HttpServletRequest" und "HttpServletResponse" erben von den allgemeineren Interfaces "ServletRequest" und "ServletResponse" und stellen HTTP-spezifische Methoden zu Cookies ("getCookies()", "addCookie()") oder zu Steuerinformationen (Köpfen, engl. header), zu Sitzungen (engl. session) oder auch zur Behandlung von Fehlern bereit.
 - (1.2) Der Container übergibt Referenzen zu zwei Objekten, die genau die in den Interfaces angegebenen Methoden implementieren. Ein Bezug wird nur über den Interface-Typ genommen, weshalb der genaue Name der implementierten Klasse und des Typs nicht bekannt sein muss [:107].

(javax.servlet.http.HttpServletRequest) Ein Sub-Interface von "ServletRequest". Ein Objekt vom Typ "HttpServletRequest" enthält die Information der HTTP-Anfrage, wie z.B. Parameternamen und -werte, Attribute und einen Eingabe-Stream. Im Klassendiagramm sind nur die wichtigsten Methoden aufgeführt.

(getParameter, getQueryString) Die in einem HTTP-Request enthaltenen Parameter werden nicht mittels der allgemeinen Methode "getParameter()", sondern mittels der HTTP-spezifischen Methode "getQueryString()" bereitgestellt.

(getRemoteAddr, getRemoteHost) Liefert die IP-Adresse bzw. den Rechnernamen des aufrufenden Web-Client.

(javax.servlet.http.HttpServletResponse) Unterstützt das Servlet beim Senden der Antwort an den Client.

(addCookie) Methode, die den spezifizierten Cookie zur Antwort hinzufügt. Die Methode kann häufiger aufgerufen werden, um mehr als ein Cookie zu setzen.

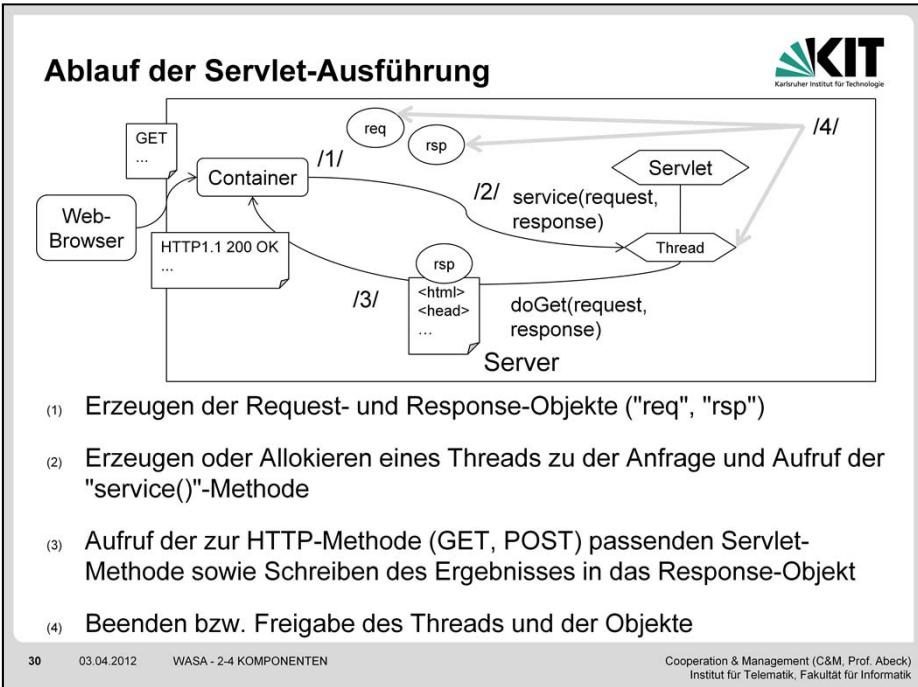
[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

[Li05] Y. Daniel Liang: Introduction to Java Programming; Pearson Prentice Hall, 2005.

[Sun-JEE5] Java EE 5 Tutorial, Sun Microsystems, September 2007.

LZ SERVLET – ÜA JAVA-KLASSEN

- (1) Wie heißen die beiden behandelten Pakete der Servlet-API und wie hängen diese zusammen?
- (2) Von welcher Java-Klasse ist ein Servlet in praktisch allen Fällen abgeleitet und in welchen Beziehungen zu anderen Klassen steht diese?
- (3) Welche Methode von dieser Klasse wird bei Empfang eines "HTTP GET" aufgerufen und welche Parameter hat diese Methode?
- (4) Welche Klasse beinhaltet die Informationen zu einer HTTP-Anfrage und was ist ein Beispiel einer Methode, die von dieser Klasse bereitgestellt wird?



[BS+08:95]

Nachfolgend wird der Ablauf rund um die "service()"-Methode und die beiden erzeugten Objekte (Instanzen der Klassen "HttpServletRequest" und "HttpServletResponse") detaillierter dargestellt.

(Web-Browser, GET) Der Ablauf wird dadurch angestoßen, dass der Benutzer auf einen Link zu einem Servlet klickt.

Der Container übernimmt die Steuerung der nachfolgend beschriebenen Schritte, die den Lebenszyklus eines Servlets beschreiben.

(1) /1/ Erzeugen der beiden Objekte "HttpServletRequest" und "HttpServletResponse".

(2) /2/ Das Servlet wird vom Container aufgrund der URL festgestellt.

Allokieren bedeutet, dass ein aus einem Pool bereits erzeugter Thread für die Bearbeitung der Anfrage ausgewählt und reserviert wird.

Beim "service()"-Aufruf werden die beiden erzeugten Objekte übergeben.

(3) /3/ Die zu den HTTP-Operationen passenden Servlet-Methoden sind "doGet()" bzw. "doPost()".

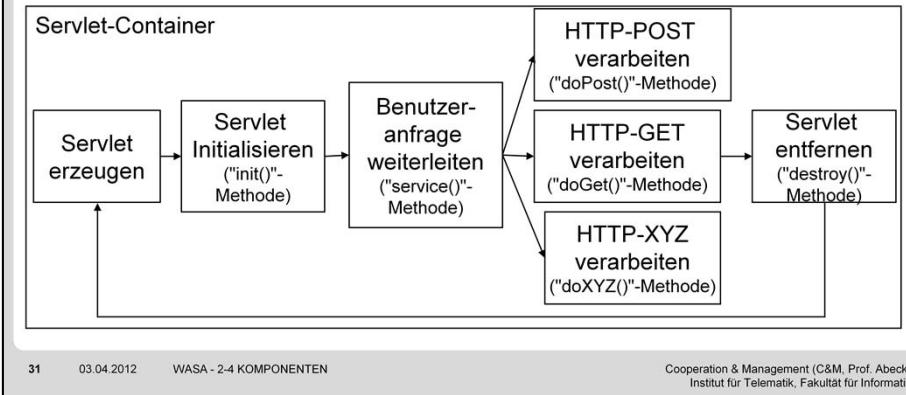
(4) /4/ Der Thread kann auf zwei Arten abgeschlossen werden:

(i) Er wird beendet für den Fall, dass er am Anfang explizit erzeugt wurde.

(ii) Er wird an einen vom Container gemanagten Thread-Pool zurückgegeben.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

- (1) Servlets besitzen einen vom Servlet-Container kontrollierten Lebenszyklus, der bei der Programmierung beachtet werden muss
- (1) Einzelne Methoden eines Servlets werden in den Zyklusphasen vom Container aufgerufen



[C&M-TD-WAJE:11]

- (1) Die Servlet-Klasse wird durch den Servlet-Container geladen und das Servlet wird instantiiert [BS+08:97].
 (1.1) Jedes Servlet besitzt eine Methode "init()", welche beim Erzeugen des Servlets aufgerufen wird. Nach der Erzeugung wird bei jeder Benutzeranfrage die Methode "service()" des Servlets aufgerufen. Diese leitet die Anfrage, abhängig von ihrer Art, an eine der "doXXX()"-Methoden weiter ("doGet()" und "doPost()" sind hier die gängigsten). Falls mehrere Anfragen gleichzeitig eintreffen, werden mehrere Threads erzeugt, welche einzeln die "service()"-Methode des Servlets aufrufen. Der Lebenszyklus eines Servlets endet mit dem Aufrufen seiner "destroy()"-Methode.

Kurzbeschreibung der einzelnen Methoden:

- "init()": Diese Methode erbringt Aufgaben, die nicht bei jeder Benutzeranfrage, sondern nur einmal beim erstmaligen Laden des Servlets erledigt werden sollen. Solche Aufgaben können beispielsweise der Setup von Datenbank-Verbindungen oder das Laden einer Datei in eine HashMap sein.
- "service()": Jedes Mal, wenn den Server eine Anfrage für das Servlet erreicht, wird ein neuer Thread erzeugt und die Methode aufgerufen. Abhängig von der HTTP-Anfrage wird dann "doGet()", "doPost()", "doPut()" oder "doDelete()" des Servlets aufgerufen. Die "service()"-Methode sollte selber nicht überschrieben werden. Falls ein Servlet erstellt werden soll, welches POST- und GET-Anfragen gleich behandeln soll, ist es besser, "doGet()" zu implementieren und in "doPost()" diese dann aufzurufen.
- "doGet()", "doPost()", "doPut()", "doDelete()": In diesen Methoden findet die eigentliche Programmierung des Servlets statt. Üblicherweise liegt der Fokus auf den GET- und POST-Anfragen. Beim Aufruf der Methode werden ein Anfrage- (vom Typ "HttpServletRequest") und ein Antwort-Objekt (vom Typ "HttpServletResponse") übergeben. Während sich im Anfrage-Objekt Daten über die Anfrage befinden (meist Daten eines ausgefüllten Formulars), wird das Antwort-Objekt genutzt, um die Antwort und eventuelle Statuscodes an den Nutzer zurückzuschicken.
- "destroy()": Bevor der Server eine zuvor geladene Servlet-Instanz beseitigt, wird die "destroy()"-Methode aufgerufen. Dies ermöglicht beispielsweise das Schließen von Datenbank-Verbindungen oder das Beenden von Threads, die im Hintergrund liefen und nicht mehr gebraucht werden.

Lebenszyklus-Methoden

- (1) Die drei Lebenszyklus-Methoden sind
 - (1) "init()": Servlet-Zustand wechselt von "nicht existent" zu "initialisiert"
 - (2) "service()": Behandelt Client-Aufrufe im Zustand "initialisiert"
 - (3) "destroy()": Zustandswechsel von "initialisiert" zu "nicht existent"
- (2) Das Interface "Servlet" erbt die Lebenszyklus-Methoden
 - (1) Von der abstrakten Klasse "GenericServlet"
 - (2) Die Methode "service()" wird in der Klasse "HttpServlet" im Hinblick auf die HTTP-Spezifik verfeinert
- (3) Jede Anfrage wird vom Container in einem eigenen Thread bearbeitet

[BS+08:103]

(1.1) Der Container ruft "init()" von einer Servlet-Instanz auf, nachdem diese Instanz erzeugt wurde. Die Methode "init()" wird dann überschrieben, wenn z.B. eine Verbindung zu einer Datenbank hergestellt werden soll.

Zum Zustand "initialisiert": Ein alternativer Name wäre "bereit für Service-Client-Anfragen" [:103].

(1.2) Behandelt die Anfrage und ruft die entsprechende Methode gemäß der HTTP-Operation auf. "service()" sollte nicht überschrieben werden.

Ruft hierzu entweder die Methode "doGet()" oder "doPost()" auf. Eine dieser Methoden ist auf jeden Fall zu überschreiben.

(1.3) Wird genauso wie "init()" genau einmal aufgerufen.

(2) Legt fest, dass alle Servlets die drei Lebenszyklus-Methoden sowie die Methoden "getServletConfig()" und "getServletInfo()" haben.

(2.1) Liefert den größten Anteil des Servlet-Verhaltens. Diese Klasse wird niemals erweitert (d.h. es wird keine Unterklasse eingeführt, die die Eigenschaften von "GenericServlet" erbt).

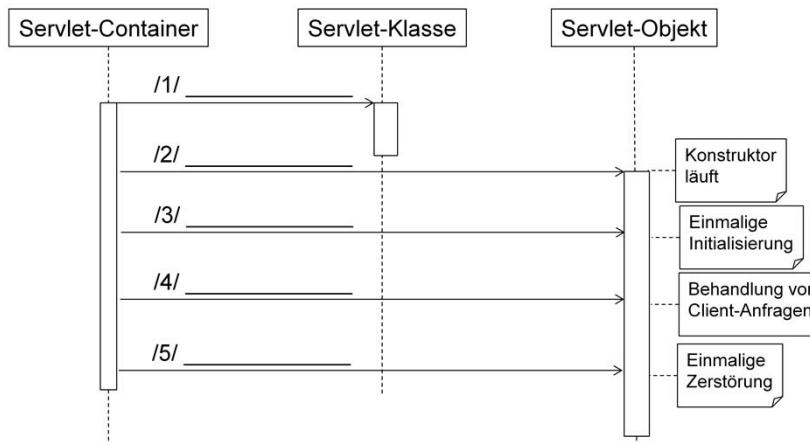
(2.2) "HttpServlet" ist eine Unterklasse von "GenericServlet" und überschreibt die Methode "service()".

(3) Bevor die erste Anfrage bearbeitet und der erste diese Anfrage Thread erzeugt/gefunden wird, startet der Container einen initialen Thread, der die "init()"-Methode aufruft.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

LZ SERVLET – ÜA LEBENSZYKLUS

(1) Die im Sequenzdiagramm fehlenden Nachrichten sind zu ergänzen



Namen eines Servlet

- (1) Vollqualifizierter Klassename
 - (1) Vom Entwickler vergebener exakter Pfad und Dateiname
 - (2) Beispiel: "\classes\FirstServlet.class"
- (2) Geheimer interner Name
 - (1) Vom Auslieferer vergeben und dem Entwickler bekannter Name
 - (2) Beispiel: "MyFirstServlet"
- (3) Öffentlicher URL-Name
 - (1) Für den Client vergebener Name
 - (2) Beispiel: "/servlets/servlet/FirstServlet"

[BS+08:46]

Die Behandlung dieser Abbildung von der URL (Uniform Resource Locator) auf das Servlet gehört zu den fundamentalsten Problemen, die sich ein Web-Anwendungsentwickler stellen muss. Entsprechend der drei Abbildungsformen lassen sich drei Arten von Namen unterscheiden, die ein Servlet haben kann.

(1) Es wird der Pfad und der tatsächliche Dateiname (engl. file path name [:46]) des Servlets verwendet.

(1.1) Der Entwickler wählt den Klassennamen (sowie ggf. den Paketnamen, der einen Teil der Verzeichnisstruktur beschreibt) und die Lokation auf dem Server.

(1.1.2) Ausgangspunkt (relative Wurzel) der Pfadangabe ist das Heimat-Verzeichnis, in dem die vom Web-Server betriebenen Web-Anwendungen liegen (z.B. "C:_Install\Apache-tomcat-6.0.29\webapps\examples\WEB-INF").

(2) Ein speziell für das Deployment ausgedachter Name, der mit dem Klassennamen übereinstimmen kann, aber auch von diesem verschieden sein kann.

(2.1) (2.1) Wird außen niemals sichtbar, da nur der Auslieferer (engl. deployer) und andere Personen, die direkt in der realen Betriebsumgebung auftreten, den Namen kennen.

Alternative Namen wären "FirstServlet" oder "SimpleServlet".

(3) Das ist der in der HTML-Datei auftretende Name, der beim Anklicken im Browser zum Aufruf des Servlet führt. Hierzu ist eine entsprechende Abbildung auf den vollqualifizierten Klassennamen erforderlich, da der URL-Name (wie der interne Name) frei erfunden ist.

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

Deployment Descriptor zur Abbildung des URL-Namen auf den vollqualifizierten Servlet-Namen

- (1) Mittels des Deployment Descriptor werden dem Container Informationen zur Ausführung des Servlets oder JSPs mitgeteilt
 - (1) Eine dieser Informationen betrifft die Abbildung
- (2) Zur Beschreibung der URL-Abbildung werden zwei XML-Elemente genutzt
 - (1) <servlet>: Bildet den internen Namen <servletName> auf den vollqualifizierten Klassennamen <servletClass> ab
 - (2) <servletMapping>: Bildet den internen Namen <servletName> auf den öffentlichen URL-Namen <urlPattern> ab

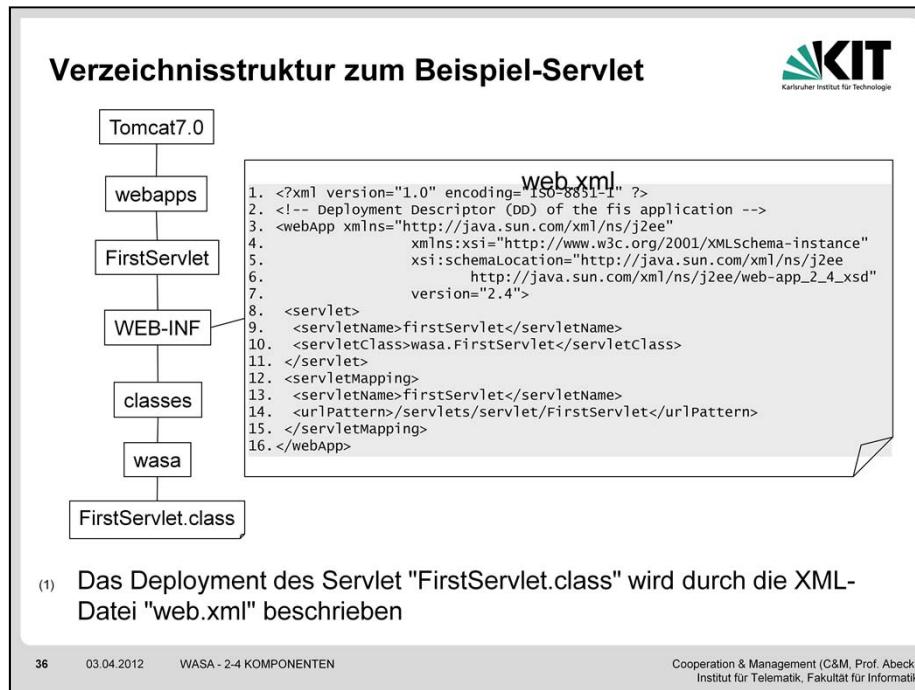
[BS+08:48]

- (1) Der Deployment Descripor (DD) ermöglicht eine deklarative Änderung der Anwendung, ohne dabei den Quellcode ändern zu müssen [:49].
 - (1.1) Andere Aspekte betreffen Sicherheitsrollen, Fehlerseiten, Tag-Bibliotheken, initiale Konfigurationsinformationen.
- (2) Der DD ist ein einfaches XML-Dokument. Das Wurzelement eines DD ist <webApp> und liefert eine deklarative Konfigurationsbeschreibung einer Web-Applikation.
 - (2.1) (2.2) Die Abbildung erfolgt in zwei Stufen URL – <servletMapping> – interner Name – <servletName> -- vollqualifizierter Name

DD Deployment Descriptor

[BS+08] Bryan Basham, Kathy Sierra, Bert Bates: Head First Servlets & JSP, Second Edition, O'Reilly, 2008.

Verzeichnisstruktur zum Beispiel-Servlet



(webapps) Um Web-Anwendungen in Tomcat ausführen zu können, muss der Code in dem "webapps"-Ordner unter der Tomcat-Installation ("Verzeichnis "Tomcat7.0") abgelegt werden. Eine eigenständige Web-Anwendung (hier: "FirstServlet") wird durch einen Ordner im "webapps"-Ordner dargestellt.

(FirstServlet) Der "FirstServlet"-Ordner ist der Wurzelordner. Dateien, die für den Benutzer sichtbar sein sollten (z.B. HTML oder JSP-Dateien), können direkt im Wurzelordner oder in einem Unterordner platziert werden.

(WEB-INF) Dieser Ordner enthält den Anwendungs-Code, d.h., Java-Klassen (u.a. Servlets) sowie in JAR angelegte Klassen und externe Bibliotheken.

Der "WEB-INF"-Ordner ist von der Client-Seite aus nicht zugreifbar. Alles, was unter diesem Ordner liegt, muss in der "web.xml"-Datei angegeben werden, um zugreifbar zu sein (vgl. Servlet-Deklaration in späteren Folien). Der "WEB-INF"-Ordner ist ein geeigneter Platz, um Ressourcen abzulegen, die nur auf der Serverseite sichtbar sein sollen [TOMCAT-DEPL].

(web.xml): Beschreibt die Komponenten, aus denen die Anwendung gebildet ist.

(FirstServlet.class) Ist das Ergebnis der Übersetzung von "FirstServlet.java".

LZ SERVLET- ÜA DEPLOYMENT

- (1) Auf welche Arten kann ein Servlet benannt werden?
- (2) Warum sollte ein Servlet nicht über seinen vollqualifizierten Namen, sondern über den frei vergebenen öffentlichen URL-Namen angesprochen werden?
- (3) Wie werden die verschiedenen in einer Web-Anwendung genutzten Servlet-Namen beschrieben?

Servlet-basiertes POI-Management

- (1) Das Management von POIs (PointOfInterest) ist ein Bestandteil der Web-Anwendung KITCampusGuide (KCG)
 - (1) Ein POI ist hier ein für KIT-Angehörige oder externe Besucher potentiell interessanterer Punkt auf dem Campus
 - (2) POIs lassen sich geeignet in Kategorien einteilen
 - (1) Institutsgebäude, Mensa, Cafeteria, Parkplatz, ...
- (2) Das POI-Management stellt Funktionen zum Erzeugen, Lesen, Ändern, Löschen und Suchen bereit
 - (1) Die Servlet-Technologie dient auf der Server-Seite zur Entgegennahme der Anfrage und zur Erstellung der Antwort
 - (2) Teile der Geschäftsfunktionalität werden i.d.R. durch eine oder mehrere andere Komponenten erbracht

(1) Die Servlet-Technologie wird im Folgenden dazu genutzt, den zum KITCampusGuide (KCG) gehörenden Teilaspekt des POI-Managements (PointOfInterest) in Form einer verteilten Web-Anwendung zu realisieren.

(1.1) Ein POI entspricht einer Datenstruktur, die in systematischer Form die für einen KCG-Anwender relevanten Informationen zu einem interessanten Punkt (definiert durch Koordinatenwerte aus dem Wertebereich, die die Fläche des Campus festgelegt) umfasst.

(1.2) Je nach Kategorie kann die Datenstruktur um weitere Attribute ergänzt werden (z.B. Anzahl aller [-> statisch] oder aller freien [-> dynamisch] Plätze auf einem Parkplatz).

(2) Die genannten Funktionen im Zusammenhang mit POIs treten üblicherweise auch bei anderen Geschäftsobjekten auf und werden in Anlehnung an die Anfangsbuchstaben der englischen Operationsbezeichnungen Create, Read, Update, Delete, Search als CRUDS-Operationen bezeichnet.

(2.1) Die Anfrage erfolgt üblicherweise mittels einer HTTP-GET- oder HTTP-PUT-Operation. Das Servlet übernimmt die Aufgabe, die in der HTTP-Operation enthaltenen Informationen so aufzubereiten, dass eine geeignete Weiterverarbeitung im Java-Programm ermöglicht wird.

(2.2) Häufig bestehen für die Geschäftsobjekt-bezogenen Funktionalitäten Komponenten (Business-Object-Komponenten, BO), die von Servlets (wieder-) verwendet werden können.

BO	Business Object (Komponente)
CRUDS	Create, Read, Update, Delete, Search
KCG	KITCampusGuide
POI	PointOfInterest

POI-Management-Ausschnitt: POI-Suche

- (1) Der POI-Management-Benutzer ruft die Anwendung aus einem Web-Browser auf
- (2) Auf der Startseite erhält der Benutzer die Möglichkeit, eine POI-Kategorie auszuwählen
- (3) Die Anwendung liefert alle POIs zurück, die zu der angegebenen Kategorie erfasst wurden

URL: http://.....

Eingabe der Kategorie, zu denen POIs gesucht werden:

Mensa

OK

URL: http://.....

Zu der gewünschten Kategorie "Mensa" bestehen die folgenden POIs:

1. Mensa am Adenauerring
2.

Die hier beschriebene Suche nach POIs erfolgt aufgrund der POI-Kategorie. Ein POI gehört zu einem oder mehreren POIs.

Hinweis: In der zu dieser Kurseinheit bestehenden Praktischen Aufgabe soll der hier beschriebene (kleine) Ausschnitt des POI-Managements durch eine Servlet-basierte Web-Anwendung realisiert werden. Die Anwendung wird so einfach wie möglich gehalten, damit der Code zu der in den nachfolgenden Praktischen Aufgabe zu erstellenden Web-Anwendung hinsichtlich seines Umfangs nicht den Rahmen sprengt. Die Anwendung enthält aber alle wesentlichen Elemente einer Servlet-basierten Web-Anwendung und lässt sich daher hinsichtlich der bereitgestellten Geschäftsfunktionalität einfach erweitern.

(1) Der Aufruf erfolgt durch die Eingabe der entsprechenden URL, unter der sich die Startseite der Anwendung befindet.

(2) Die Auswahl sollte über eine Liste erfolgen, in der alle möglichen Kategorien, die bislang in der Anwendung angelegt wurden, angezeigt werden.

Im Beispiel wählt der Benutzer die Kategorie "Mensa" aus. Durch Anklicken von "OK" wird die Anfrage bearbeitet (d.h. es erfolgt eine Anfrage vom Client zum Server).

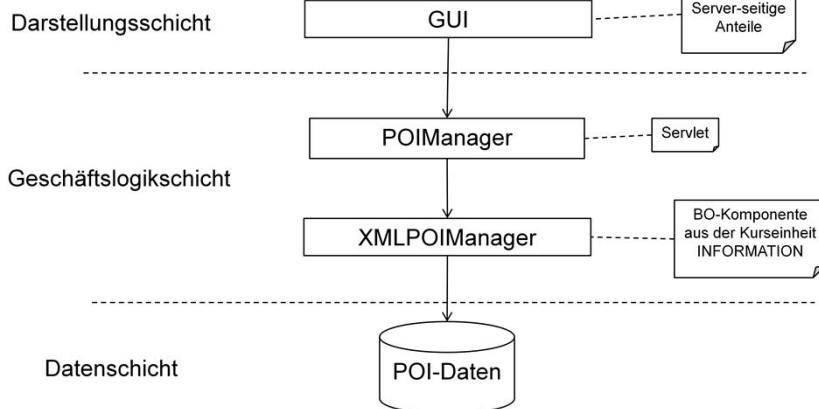
(3) Ggf. können zu jedem gefundenen POI noch die wichtigsten weiteren Informationen (z.B. Lage in Form der Koordinaten) angegebenen werden.

Benutzerfreundlicher (aber auch sehr viel aufwändiger) wäre die Anzeige der POIs in der Campuskarte, um dem Benutzer die geographische Lage der POIs zu verdeutlichen.

Software-Architektur zur Web-Anwendung "POI-Management"



KIT
Karlsruher Institut für Technologie



- (1) Die Komponenten zum POI-Management sind in einer 3-Schichten-Architektur angeordnet

40

03.04.2012

WASA - 2.4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

Zunächst wird die Software-Architektur informell (ohne Nutzung der Modellierungssprache UML) dargestellt.

(1) Die Beschreibung dient insbesondere dazu, die wesentlichen für die Web-Anwendung des POI-Managements benötigten Komponenten einzuführen und in die bekannte 3-Schichten-Architektur einzuordnen.

(POI-Daten, XMLPOIManager) Die Daten zu den POIs liegen in einer XML-Datei vor, auf die die Komponente "XMLPOIManager" lesenden und schreibenden Zugriff hat. Der "XMLPOIManager" ist die Business-Object-Komponente (BO) in der vorliegenden Anwendungsarchitektur.

(POIManager) In dieser Komponente befindet sich der dynamische Anteil der Geschäftslogik, die aufgrund seiner Einfachheit vollständig durch ein Servlet erbracht wird.

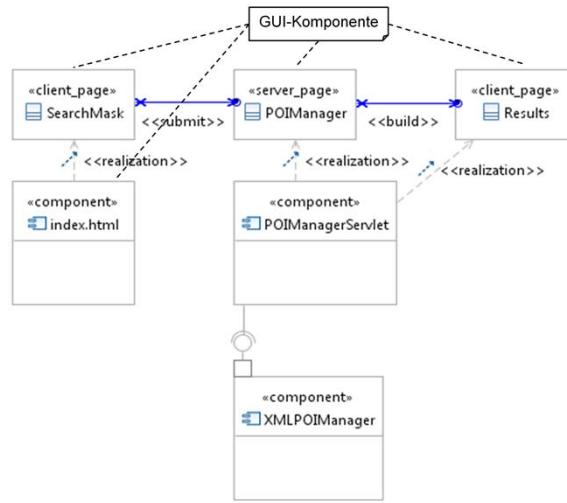
(GUI) Die GUI wird größtenteils durch einen Standard-Web-Browser erbracht. Allerdings bestehen auch Server-seitig Komponenten, die zur Darstellungsschicht und damit zur GUI beitragen, wie im nachfolgenden Komponentendiagramm zu der Web-Anwendung deutlich wird.

BO	Business Object
GUI	Graphical User Interface

Komponentendiagramm zur Beschreibung der logischen Struktur der Web-Anwendung

(1) <<client_page>>
und
<<server_page>>
dienen zur
Verdeutlichung der
beiden bei einer
Web-Anwendung
auftretenden Seiten

(2) Eine geeignete
Stereotypisierung
der beteiligten
Komponenten ist zu
empfehlen
(1) Beschränkung
durch die UML2-
Tools



41

03.04.2012 WASA - 2-4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

[Co00:157]

Das Komponentendiagramm veranschaulicht die logischen Verbindungen zwischen den einzelnen Komponenten.

(1) Um eine Web-Anwendung besser beschreiben zu können, ist es angebracht, den Unterschied zwischen Client- und Server-Seiten darzustellen und aufzuzeigen, welche Komponenten für diese verantwortlich sind. Bei Laufzeit-Komponenten, wie die Instanzen eines Servlets oder JSP-Seiten, werden die Stereotypen <<client_page>> und <<server_page>> benutzt. Diese sind eine Erweiterung der UML-Klasse [Co00:157]. Eine <<client_page>> bezeichnet eine Seite, die dem Benutzer zugeführt wird. Eine <<server_page>> bezeichnet eine Seite mit dynamischen Elementen, wie z.B. Java-Code, die nur auf der Server-Seite ausgeführt werden.

(2) Auch die Komponenten sollten stereotypisiert sein, um dem Diagramm mehr Aussagekraft zu verleihen. Die Deployment-Komponenten unterscheiden sich auch durch die <<web_page>>, <<servlet>> oder <<jsp>> Stereotypen.

(2.1) Bei den UML2-Tools ist es noch nicht möglich, die Stereotypen der Komponenten anzuzeigen also wird hier davon abstrahiert.

[Ke09] Christoph Kecher: UML2 Das umfassende Handbuch, 3. Auflage, Galileo Computing, 2009.
[Co00] Conallen Jim: Building Web Applications with UML, Addison-Wesley Longman, 2000.

Zu den verwendeten Modellierungselementen

- (1) Stereotypen zu Assoziationen
 - (1) <<submit>>
 - (2) <<build>>
 - (3) <<realization>>
- (2) Komponenten
 - (1) index.html
 - (2) POIManagerServlet
 - (3) XMLPOIManager
- (3) Web-Seiten
 - (1) <<client_page>> SearchMask
 - (2) <<client_page>> Result
 - (3) <<server_page>> POIManager

(1.1) Mit dem <<submit>>-Stereotypen einer gerichteten Assoziation kann man darstellen, dass eine Seite eine Anfrage an eine andere Seite stellt. Hier wird das Senden der POI-Kategorie von der HTML-Form zu dem Servlet modelliert.

(1.2) Mit dem <<build>>-Stereotypen einer gerichteten Assoziation kann man darstellen, dass eine Seite eine andere Seite aufbaut. Diese Assoziation kann nur von einer Server-Page zu einer Client-Page gezogen werden da nur Server-Pages dynamische Elemente besitzen. Hier wird die Client-Page "Results" von der Server-Page "POIManager" gebaut.

(1.3) Mit einer <<realization>>-Beziehung kann man darstellen, welche Komponente für welche der Seiten verantwortlich ist und sie erzeugt.

In den UML2-Tools wird die "realization" nicht unterstützt, weshalb ein "realization"-Stereotyp auf Basis einer Abhängigkeit (engl. dependency) erstellt wird.

(2.1) Die Datei "index.html" ist eine statische HTML-Seite und ist verantwortlich für die Generierung der "SearchMask"-Client-Page.

(2.2) Diese Komponente ist ein Servlet, gekennzeichnet durch den "Servlet"-Suffix. Sie ist verantwortlich für die Generierung der Antwort auf die Anfrage des Klienten, indem sie eine Server-Page "POIManager" realisiert.

(2.3) Die Komponente "XMLPOIManager" bietet eine Schnittstelle für den Zugriff auf POI-Daten. Dieses Verhalten wird mit der "Ball-and-Socket"-Notation dargestellt.

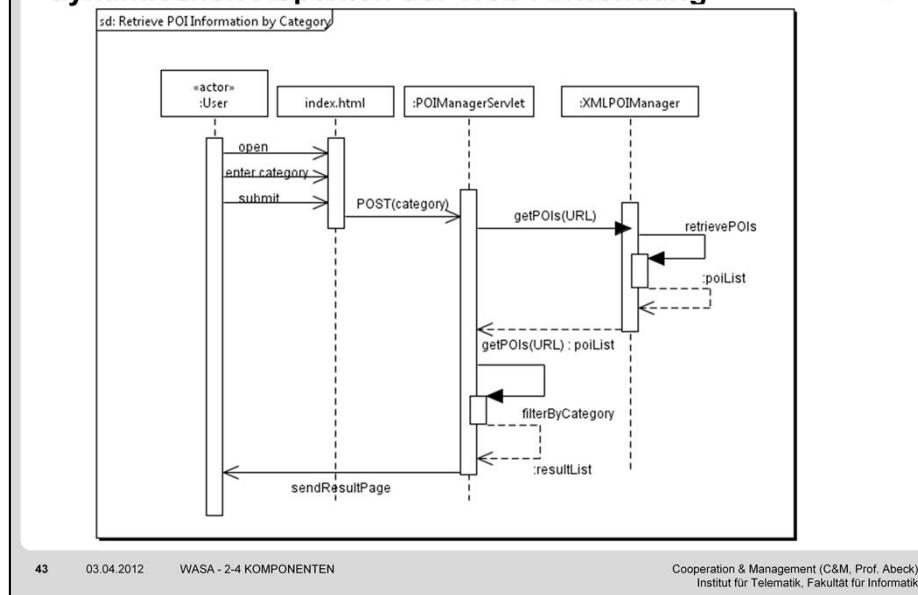
(3.1) Diese Seite wird dem Benutzer angezeigt.

(3.2) Dies ist die von der Server-Page "POIManager" generierte Antwort auf die Anfrage des Benutzers und stellt die Information als HTML dar.

(3.3) Die Server-Page "POIManager" enthält den dynamischen Teil (die ausführbare Logik) und erstellt die Antwort der Anfrage in Form der Client-Page "Results".

(XMLPOIManager)

Sequenzdiagramm zur Beschreibung von dynamischen Aspekten der Web-Anwendung



Die Abfolge der Interaktionen zwischen den Artefakten und dem Benutzer wird anhand eines Sequenzdiagramms gezeigt.

Hinweis: Die UML2-Tools sind leider noch nicht in einem Entwicklungszustand, der die Erstellung guter Sequenzdiagramme ermöglicht. Deshalb wird in diesem Beispiel ein Diagramm benutzt, dass in Eclipse mit dem Papyrus-Diagramm-Editor-Plug-In erstellt wurde [CM-TD-PIB].

(User) Der Benutzer ruft die Startseite "index.html" des POI-Management-Services des KITCampusGuide auf.

(index.html) Dem Benutzer wird die Möglichkeit angeboten, nach POIs einer bestimmten Kategorie zu suchen. Mit Hilfe einer Web-Form wird der vom Benutzer eingegebene Suchbegriff an das "POIManager"-Servlet weitergeleitet.

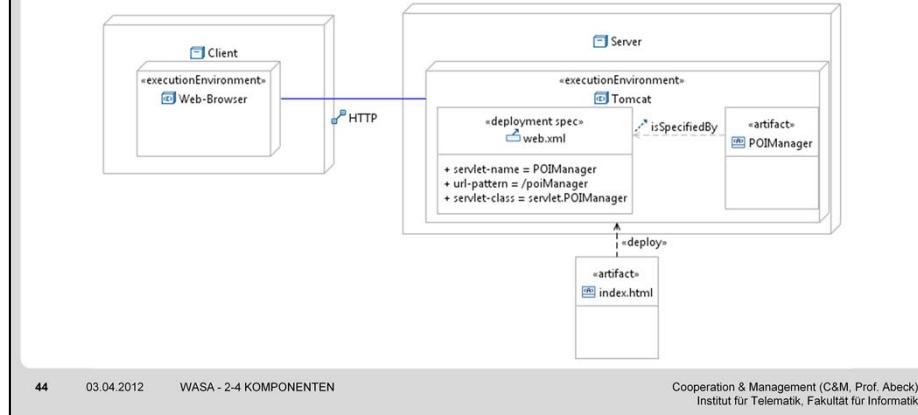
(POIManagerServlet) Das Servlet erhält die POST-Anfrage und sendet einen Aufruf an den "XMLPOIManager", um alle POIs zu erhalten. Nachdem die POI-Daten empfangen wurden, wird eine Filterung der POIs nach ihrer Kategorie durchgeführt. Die Informationen der POIs mit passender Kategorie werden mittels HTML formatiert und dem Benutzer wird eine Seite mit den angefragten Informationen zurückgeliefert.

(XMLPOIManager) Die "XMLPOIManager"-Komponente wurde in einer in der Kurseinheit INFORMATION gestellten Praktischen Aufgabe detailliert behandelt und wird hier als Black-Box-Komponente betrachtet. Die Komponente liefert eine Liste aller gerade im Datenbestand vorliegenden POIs.

[CM-TD-PIB] Cooperation & Management: Papyrus – Installation und Benutzung, Technische Dokumentation, KIT, C&M (Prof. Abeck), 2011.

Verteilung der Anwendung

- (1) Tomcat ist die Ausführungsumgebung für das Servlet "POIManager"
 - (1) Die für die Ausführung des Servlets benötigte Deployment-Information wird durch "web.xml" bereitgestellt



(1) (Tomcat) Die Ausführungsumgebung wird mit Hilfe des Stereotyps <>executionEnvironment<> dargestellt. Der POIManager ist das Servlet, das in Tomcat ausgeführt wird und die gewünschte POI-Funktionalität bereitstellt. Die Artefakte werden mit Hilfe des Stereotyps <>artifact<> dargestellt [Ke09:161].

(1.1) Der Deployment-Deskriptor "web.xml" ist die Spezifikation für das Deployment der Artefakte, wie z.B. das "POIManager"-Servlet. Die Konfigurationsdaten des "POIManager"-Servlets werden bei der Implementierung in "web.xml" eingetragen.

(isSpecifiedBy) Dies ist eine Beziehung zwischen der Deployment-Spezifikation "web.xml" und dem zu deployenden Artefakt, dem "POIManager"-Servlet.

(Client) Der Client-Knoten mit einem Web-Browser als Ausführungsumgebung. Die Ressourcen werden mit Hilfe des Web-Browsers aufgerufen.

(HTTP) Ein Kommunikationspfad (engl. communication path), der durch das HTTP-Protokoll die Kommunikation des Web-Browsers mit dem Web-Container Tomcat ermöglicht.

(index.html) Dies ist das Frontend der Web-Anwendung, die zum Tomcat-Server in einer <>deploy<>-Beziehung steht.

(deploy) Diese Beziehung sagt aus, dass das Artefakt "index.html" in der Ausführungsumgebung "Tomcat" laufen wird.

[TOMCAT-UG] Tomcat - A Minimalistic User's Guide, http://www.jjakarta.org/tomcat/tomcat3.2-4.0/tomcat-3.2.3/doc/uguide/tomcat_ug.html

[Ke09] Christoph Kecher: UML2 Das umfassende Handbuch, 3. Auflage, Galileo Computing, 2009.

LZ SERVLET – ÜA BEISPIEL POI-MANAGEMENT

- (1) Welche Funktionen stellt das POI-Management bereit und wie heißt das Akronym, durch das solche Geschäftsobjekt-bezogenen Operationen bezeichnet werden?
- (2) Was leisten die Komponenten "POIManager" und "XMLPOIManager"?
- (3) Welche Arten von Seiten treten in der Softwarearchitektur einer Web-Anwendung auf und wie kann in UML eine entsprechende Modellierung unterstützt werden?
- (4) Wie erfolgt in der beschriebenen Lösung die Ermittlung der POIs zu einer bestimmten Kategorie und wie wurde das Vorgehen in UML modelliert?
- (5) Welche Informationen enthält das Verteilungsdiagramm zum Artefakt "web-xml"?

JavaServer Pages (JSP)

- (1) JSP ermöglicht eine Ergänzung von regulärem statischen HTML um dynamische Inhalte
 - (1) Verwendung von speziellen Tags für die Einbindung der dynamischen Anteile
- (2) Zusammenhänge mit Servlets
 - (1) Äußerlich erscheinen Servlets und JSPs völlig unterschiedlich
 - (2) Hinter den Kulissen ist eine JSP eine besondere Präsentation eines Servlets
 - (3) Erfüllen ganz unterschiedliche Einsatzzwecke
- (3) Nachteile von Servlets, die durch JSPs behoben werden
 - (1) Das Schreiben und Warten von HTML ist umständlich
 - (2) Standard-HTML-Werkzeuge können nicht verwendet werden
 - (3) Keine Trennung von Web-Designer und Anwendungsprogrammierer

[HB11:303-Chap10]

Servlets haben den Nachteil, dass die Erzeugung von HTML in Form von "print"-Befehlen nicht angemessen ist. Der Grund hierfür besteht darin, dass ein Servlet als Java-Quellcode vorliegt und HTML nur in dieser Form eingebunden werden kann. JavaServer Pages (JSP) drehen die Nutzung der beiden beteiligten Sprachen (Java und HTML) um, indem von HTML ausgehend dynamische Teile durch Java-Befehle ergänzt werden.

(1) Der HTML-Inhalt einer JSP lässt sich mit einem normalen HTML-Editor erstellen.

(1.1) Die meisten der Tags beginnen mit "<%" und enden mit "%>".

(2.1) Weil Servlets primär Java-Code und JSPs primär HTML-Code sind.

(2.2) JSPs kann als eine spezielle Beschreibungsform eines Servlets aufgefasst werden, weil es zu dessen Ausführung zunächst in ein Servlet überführt werden muss.

(2.3) Servlets unterstützen die Programmierung und die Datenverarbeitung, während mittels JSPs das Präsentationsproblem besser gelöst werden kann.

(3.1) Die Umständlichkeit ergibt sich aus den syntaktischen Gegebenheiten aufgrund der Verwendung der "print"-Anweisungen (Klammern, Semikolon, Schrägstrich, Stringverknüpfung, ...).

(3.2) Diese Werkzeuge tragen zu einer hohen Effizienzsteigerung bei der HTML-Bearbeitung bei.

(3.3) Ein Web-Designer sollte nicht mit Java-Code umgehen müssen, um sich ganz auf die Web-Oberflächengestaltung konzentrieren zu können. Durch eine klare Trennung der Kompetenzen können die Kompetenzen im Entwicklerteam in größeren Projekten besser aufgeteilt werden.

JSP

JaverServer Pages

[HB11] Marty Hall and Larry Brown: Core Servlets and JavaServer Pages, Free Online Version of Second Edition, <http://pdf.coreservlets.com/>, 2011.

Vergleich mit konkurrierenden Technologien

(1) ASP.NET

- (1) Mangelnde Portierbarkeit
- (2) Einschränkungen hinsichtlich der Sprache

(2) PHP

- (1) Freie Open-Source, in HTML eingebettete Skriptsprache
- (2) Eigene und im Vergleich zu Java eingeschränkte Sprache

(3) Reine Servlets

- (1) Keine Trennung von Präsentation und Inhalt und damit auch keine Arbeitsteilung von Web-Designer und Programmierer möglich

(4) JavaScript

- (1) Sprache zum dynamischen Generieren von HTML auf der Client-Seite
- (2) Komplementäre und nicht konkurrierende Technologie zu JSP

[HB11:306]

JSP war nicht die erste Technologie, die die Idee umsetzte, statisches HTML mit dynamische Code zu vermischen. In diesem Bereich war neben ColdFusion auch Microsoft mit seinem ASP-Konzept (Active Server Pages) früher auf dem Markt als die Java-Community mit dem JSP-Konzept.

(1) JSP hat gegenüber der gut entworfenen Microsoft-Technologie zwei Vorteile

(1.1) JSP ist im Gegensatz zu ASP.NET auf beliebige Serversysteme und darauf laufenden Betriebssystemen portierbar. Zwar läuft die .NET-Plattform auch auf einigen Nicht-Windows-Plattformen, hingegen der ASP-Anteil nicht.

(1.2) Die Nutzung von C# und der zugehörigen Bibliotheken ist weitaus weniger verbreitet als die Nutzung von Java. Der immer noch stark genutzte Vorgänger ASP hat außerdem den Nachteil, dass für den dynamischen Anteil wenig mächtige Sprachen (z.B. VisualBasicScript VBScript) verwendet werden.

(2) PHP ist ein "rekursives Akronym" und steht für "PHP: Hypertext Preprocessor".

(2.1) PHP ist ähnlich zu JSP und zu ASP.

(2.2) JSP bietet im Vergleich zu PHP den Vorteil, dass die bereits weit verbreitete und aufgrund der zahlreich vorhandenen APIs mächtige Sprache ein besseres Erlernen unterstützt.

Ein weiterer Vorteil von JSP ist seine im Vergleich zu PHP sehr viel stärkere Unterstützung durch Werkzeuge und Serveranbieter.

(3) JSPs werden in Servlets überführt und sind daher, rein technisch gesehen, nicht anderes als Servlets.

(3.1) JSPs haben einen engen Bezug zu Servlets, die ein Programmierer von JSPs wissen sollte, um insbesondere auch entscheiden zu können, wann JSPs und wann Servlets genutzt werden sollten.

(4.2) JSP-Seiten können SCRIPT-Tags für JavaScript beinhalten oder sogar JavaScript-Code dynamisch generieren.

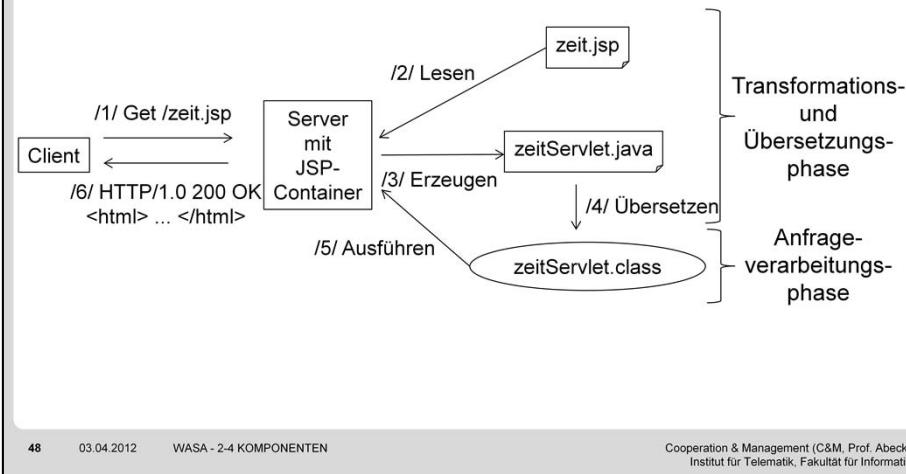
Hinweis: Es bestehen Alternativ-Vorschläge zu JSP, wie z.B. Velocity oder WebMacro, von denen aus Gründen der Standardisierung und Integration abzuraten ist. Sinnvoll sind hingegen Ansätze, durch die die JSP- und Servlet-Technologie verbessert wird; hierzu gehört z.B. die Lösung Apache Struts.

ASP	Active Server Pages (Microsoft)
PHP	PHP: Hypertext Processor
VBScript	VisualBasicScript (Microsoft)

[HB11] Marty Hall and Larry Brown: Core Servlets and JavaServer Pages, Free Online Version of Second Edition, <http://pdf.coreservlets.com/>, 2011.

JSP-Ausführungsablauf

- (1) Zur Ausführung eines JSPs wird dieses durch den Container in ein Servlet übersetzt



[Li05:1062]

(1) Eine JSP-Seite wird in ein Servlet transformiert und anschließend übersetzt. Zum Zeitpunkt der Benutzeranfrage wird eine JSP-Seite also als Servlet ausgeführt. Der einzige Unterschied zu Servlets liegt daher in der Entwicklung.

/1/ Aufruf der JSP-Seite durch eine HTTP-GET-Operation, die den URL dieser Seite enthält.

/2/ Lesen der den Inhalt der JSP-Seite enthaltenen Datei von dem als persistenten Datenbehälter dienenden Dateisystem.

/3/ Die Transformation von der JSP-Seite (hier: "zeit.jsp") in ein äquivalentes Servlet ("zeitServlet.java") führt der Container durch.

/5/ /6/ Das Servlet erzeugt die gewünschte HTML-Ausgabe gemäß den (statischen und dynamischen) Inhalten, die in der JSP-Seite enthalten sind.

Die im Beispiel angegebene JSP-Seite "zeit.jsp" mit der aktuellen Zeit als dynamischem Inhalt wird auf der nachfolgenden Seite gezeigt.

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.

JSP-Beispiel "zeit.jsp"

- (1) Eine JSP entspricht einer HTML-Seite mit speziellen JSP-Tags
- (2) Durch die JSP-Tags wird der dynamische Inhalt erzeugt
- (3) Syntax eines JSP-Tags: <%= Befehlsfolge %>
- (4) Beispiel

```
1. <!-- zeit.jsp -->
2. <html>
3. <head>
4.   <title>
5.     AktuelleZeit
6.   </title>
7. </head>
8. <body>
9. Die JSP-Seite wurde ausgeführt um <%= new java.util.Date() %>
10. </body>
11.</html>
```

[Li05:1062]

JSP liefert eine einfache Möglichkeit zur Erstellung dynamischer Web-Seiten. Ein Nachteil von Servlets ist die Einbettung von HTML-Tags und Text in Java-Code. Durch JSP wird der statische Anteil der Webseite vom dynamischen Anteil getrennt, wodurch der Code einfacher zu verstehen und zu warten ist.

(1) Die HTML-Seite ohne die JSP-Tags repräsentiert den statischen Teil.

(2) (3) Im Beispiel ist das die aktuelle Zeit (Zeile 9.), die durch den mittels JSP-Tag eingebetteten Java-Befehl berechnet wird.

(9. ... <%= new java.util.Date() %>) Es wird durch den Befehl innerhalb des JSP-Tags ein Objekt der Klasse "java.util.Date" erzeugt.

Die Ausgabe der aktuellen Zeit als String wird durch (automatisch durchgeföhrten) Aufruf der Methode "toString" auf das Objekt erreicht.

[Li05] Y. Daniel Liang: Introduction to Java Programming; Companion Website: www.prenhall.com/liang, Pearson Prentice Hall, 2005.

(1) Modell1-Architektur

- (1) Verwendung von JSPs sowohl für die Präsentation als auch für die Geschäftslogik
- (2) Vorteil: Einfachheit des Konzepts und dessen Umsetzung
- (3) Nachteil: Nur für einfache Aufgaben geeignet

(2) Modell2-Architektur

- (1) Verfolgung des Model-View-Controller-Paradigmas (MVC)
 - (1) Entgegennahme der Anfragen durch die als Controller agierenden Servlets
 - (2) Servlets beauftragen die den View realisierende JSPs
 - (3) Verwendung von JavaBeans als Modell
- (2) Vorteil: Klare Trennung von Präsentation und Logik
- (3) Nachteil: Aufwändiger und schwieriger zu implementieren

[AU01:172]

Als die JSPs eingeführt wurden, haben sich in der JSP-Entwicklergemeinschaft zwei als "Modell1" und "Modell2" bezeichnete Architekturen herausgestellt.

(1) "Modell1" hat durchaus seine Berechtigung, wenn eine einfache Web-Anwendung mit unbedeutenden Verarbeitungsanforderungen zu entwickeln ist.

(1.1) Hierdurch wird die für komplexere Anwendungen unbedingt zufordernde Abgrenzung von Präsentation und Funktion (Geschäftslogik) abgeschwächt.

(1.2) Es sind keine weitergehenden Architekturkenntnisse beim Entwickler notwendig, was die Programmierung insgesamt erheblich vereinfacht.

(1.3) Der Code wird schnell unübersichtlich, was negative Folgen auf die Erweiterbarkeit und Wartbarkeit der Web-Anwendung hat.

(2.1) "Modell2" ist die von den meisten Entwicklern favorisierte Architektur zur Verwendung von JSPs.

(2.1.1) Eine wichtige Entwurfsentscheidung betrifft die angemessene Anzahl an Servlets, die zwischen einem Servlet für alles" und "ein Servlet für jede einzelne Aktion oder Anwendungsfall" liegen kann.

(2.1.2) In diesem Fall ist ein JSP nur für die Präsentationsaspekte zuständig. Der eingebettete Java-Code dient ausschließlich dazu, mit Servlets oder anderen Steuer-/Dateneinheiten zu kommunizieren.

(2.1.3) JavaBeans sind das Kommunikationsvehikel zwischen den (Controller-) Servlets und den (View-) JSPs.

(2.2) (2.3) Es werden höhere Anforderungen an den Entwickler gestellt, was aber durch eine höhere Programmierfreundlichkeit dieses Modells gerechtfertigt ist.

Hinweis: Es besteht eine als "Modell1.5" bezeichnete Architektur, die im Wesentlichen dem "Modell1" entspricht, in dem die Logik durch JavaBeans anstelle von JSPs erbracht wird.

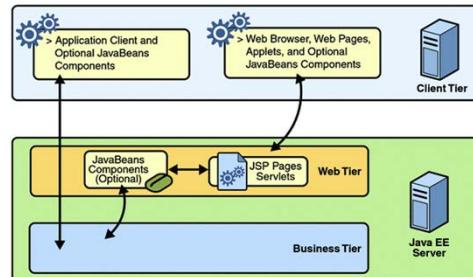
MVC

Model-View-Controller

[AU01] Khawar Zaman Ahmed, Cary E. Umrysh: Developing Enterprise Java Applications with J2EE and UML, Addison-Wesley, 2001.

JavaBeans in JSP-Dokumenten

- (1) Trennung von Präsentation, Inhalt und Logik durch JavaBeans
- (2) JavaBeans sind Java-Klassen mit bestimmten Konventionen
- (3) Mit speziellen Anweisungen direkt in JSP-Seiten benutzbar
- (4) Zwischen JSP-Seiten austauschbar



51 03.04.2012

WASA - 2.4 KOMPONENTEN

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Um bei der Entwicklung die Trennung zwischen Präsentation und Inhalt zu fördern (besonders nützlich wenn man die Arbeit zwischen Web-Entwickler und Java-Entwickler verteilen will), versucht man in JSP-Dokumenten den eingebetteten Java-Code so gering wie möglich zu halten. Ein wichtiges Instrument bei diesem Vorgehen sind JavaBeans-Komponenten. Die Nutzung von JavaBeans ermöglicht die Handhabung von Java-Objekten mit XML-kompatibler Syntax. Zusätzlich erleichtert es die gemeinsame Nutzung von Objekten zwischen mehreren Seiten oder Anfragen.

(2) Diese Konventionen sind:

- (i) Eine Bean-Klasse hat einen "default"-Konstruktor.
- (ii) Eine Bean-Klasse hat keine öffentlichen (engl. public) Instanzvariablen.
- (iii) Auf Werte innerhalb einer Bean-Klasse können nur über Methoden zugegriffen werden ("getX" / "setX") .

(3) "jsp:useBean" ermöglicht die Erzeugung oder den Zugriff auf eine existierende Bean.

"jsp:getProperty" ermöglicht das Lesen eines Wertes innerhalb einer Bean. Dies entspricht dem Aufruf einer "getX()"-Methode in Java. "jsp:setProperty" ermöglicht das Modifizieren eines Bean-Wertes.

(4) Durch das "scope"-Attribut kann der Bereich festgelegt werden, in dem die Bean genutzt wird. Die vier möglichen Werte von "scope" sind:

- (i) "<jsp:useBean...scope='page' />" ist gleichzeitig auch der voreingestellte (engl. default) Wert. Hier wird die Bean für jede Anfrage neu erzeugt.
- (ii) "<jsp:useBean...scope='request' />" ermöglicht die gemeinsame Nutzung der Bean innerhalb zweier JSP-Seiten.
- (iii) "<jsp:useBean...scope='session' />" wird benutzt, um eine Bean zwischen mehreren Anfragen des selben Benutzers zu verwenden.
- (iv) "<jsp:useBean...scope='application' />" ist gleichbedeutend mit der gemeinsamen Nutzung der Bean innerhalb aller JSP-Seiten und Benutzern einer Web-Anwendung.

LZ JSP – ÜA GRUNDLAGEN

- (1) Welche Zusammenhänge und welche Unterschiede bestehen zwischen JSPs und Servlets?
- (2) Was ist ein Beispiel einer konkurrierende Technologie zu JSP und wie ist diese zu bewerten?
- (3) Welche Nachteile von Servlets werden durch JSPs behoben?
- (4) Welche Schritte werden bei der Ausführung einer JSP durchlaufen?
- (5) Kann es sinnvoll sein, eine Web-Anwendung ausschließlich mittels JSPs zu entwickeln?
- (6) Durch welche Komponenten werden Modell, Sicht und Steuerung in der Modell2-Architektur ("Model2") realisiert?