Studie zum systematischen Testen von webbasierten Client-Server Anwendungen

Zwischenbericht - Großer Beleg

Thomas Förster

Betreuerin: Dipl.-Softwaretechnol. Anne Thomas
Hochschullehrer: Prof. Dr. rer. nat. habil. H. Hußmann

Überblick

- Aufgabe und Motivation
- Webbasierte Anwendungen
  - Beispiel: ProjectWeb
- Grundlagen des Testens
  - Grundbegriffe und Testverfahren
  - Vorgehen beim Entwicklertest
- Untersuchte Testframeworks
  - Erstellung von Testklassen
  - Probleme bei mehrschichtigen Anwendungen
  - Erste Bewertung und Einschätzung
- Bisher Erreichtes
- Ausblick

Aufgabe

- Ermittlung und Darstellung vorhandener Testansätze und Verfahren für webbasierte Anwendungen
- Beispiele: ProjectWeb und jExam
- Einarbeitung, Auswertung und Bewertung von verschiedenen Testframeworks für Entwicklertests
- Erstellung einer Testumgebung für ProjectWeb unter Nutzung geeigneter Testframeworks

Motivation

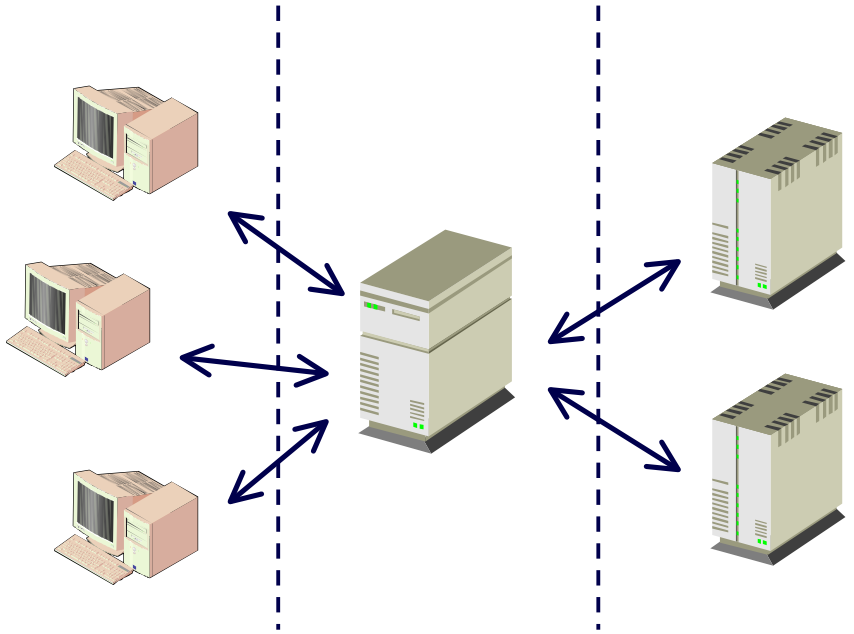
- Test und Wartung erfordert ca. 50% des Gesamtaufwandes eines Softwareprojektes
  - Hoher Zeitbedarf -> höhere Kosten -> mehr Qualität
  - Weniger Tests -> weniger Qualität -> unzufriedene Kunden
- Fragen:
  - Wie wird möglichst effizient getestet?
  - Manuelle oder automatische Tests?
  - Welche Werkzeuge sollten genutzt werden?

„Wenn Sie glauben, daß Entwurf und Codierung dieses Programmes schwierig war, so haben Sie noch nichts mitbekommen.“ (Myers)

Webbasierte Anwendungen

- Meist mehrschichtige Applikationen (2-, 3-, 4-Tier)
  - Anwendungscode nur auf dem Server
  - Thin-Clients (Webbrowser)
- Realisierung auf Grundlage der Java 2 Enterprise Edition (J2EE) möglich
  - Nutzung u.a. von Servlets, Java Server Pages (JSP) und Enterprise Java Beans (EJB)
- Trend der Entwicklung webbasierter Anwendungen hält an
  - E-Commerce
    - Weihnachtsgeschäft 2001: Anstieg um 44% gegenüber Vorjahr in Europa, Umsatz von 2,9 Milliarden Euro
  - Business-To-Business
  - Business-To-Customer

Webbasierte Anwendungen

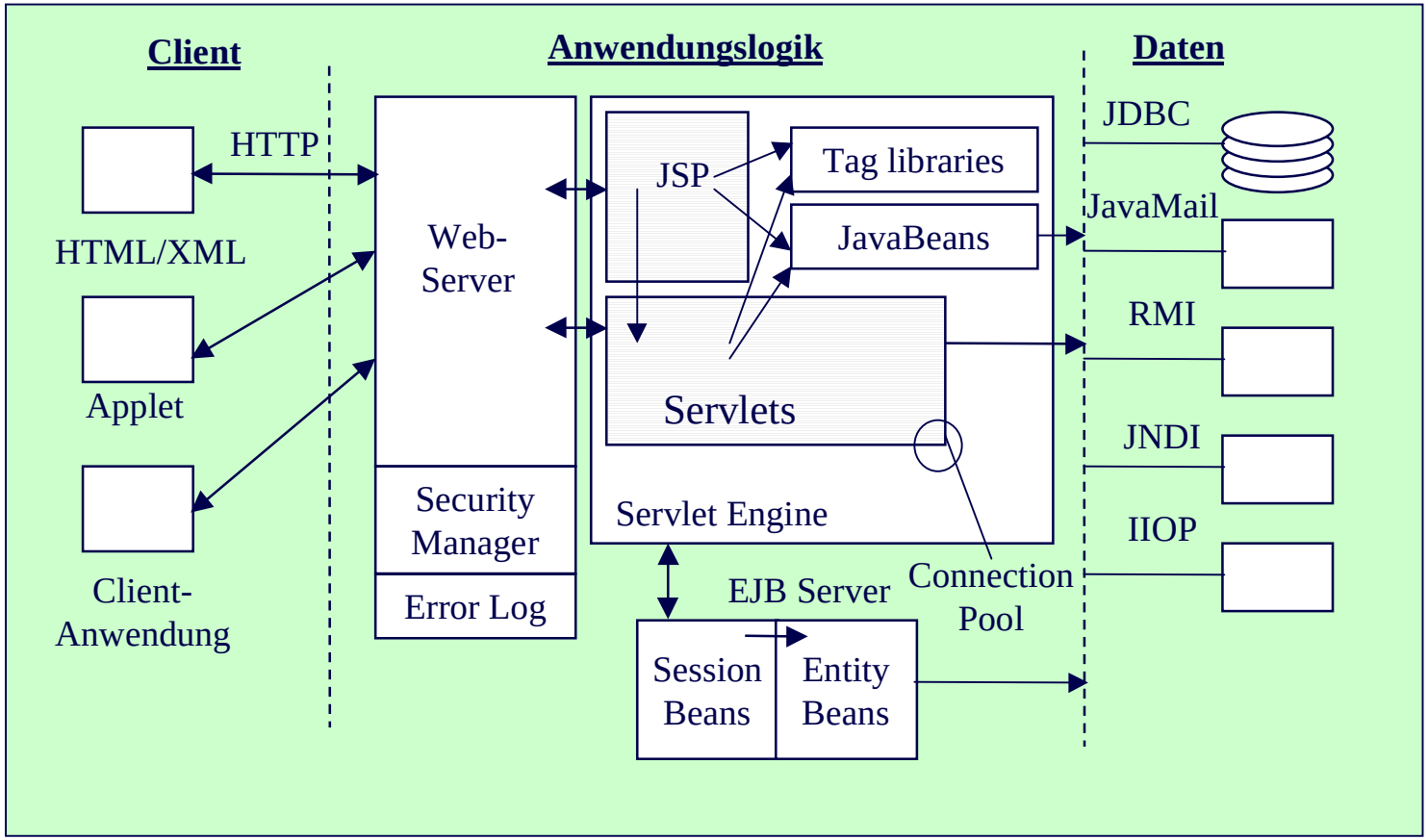


Stufe 1: grafische Benutzeroberfläche

Stufe 2: Web- und Applikationsserver

Stufe 3: Datenbank(en)

Webbasierte Anwendungen



Beispiel: ProjectWeb

- Anwendung zur webbasierten Verwaltung von Forschungsprojekten
- Funktionen:
  - Nutzer- und Rollenmanagement
  - Verwaltung von Projekteinheiten, Arbeitsplänen, Beschreibungen
  - Dokumentenverwaltung
  - Adressbuch

- Realisierung auf Grundlage der J2EE
- Einfache Auszubauen durch Modulstruktur

Grundlagen des Testens

- Definition „Testen“ durch Myers (1978):

  „Testen ist der Prozeß, ein Programm mit der Absicht auszuführen, Fehler zu finden. “

  - Destruktiver Vorgang
  - Kann nur die Anwesenheit von Fehlern und Mängeln feststellen
  - Restfehlerzahl gegen Null minimiert
  - In der Praxis häufigste Maßnahme zur Qualitätssicherung

Grundbegriffe

- Testfall:
  - spezifiziert Ausgangszustand, Eingabewerte oder Bedingungen und erwartete Ausgabewerte
  - Erfolgreich wenn er einen Fehler entdeckt
- Testsequenz:
  - Zusammenfassung von Testfällen
- Spanne oder Reichweite:
  - Ganze Methoden bis hin zum gesamten System

- Verschiedene Testverfahren:
  - Blackbox-Test:
    - Betrachtet Programm von aussen in seinen Schnittstellen
  - Whitebox-Test:
    - Untersucht innere Struktur des Programms

Weitere Testverfahren

- Unit-Test:
  - Test einer einzelnen Klasse getrennt von ihrer Umgebung
- Interaktionstest:
  - Test des Zusammenspiels mehrerer Objekte
- Regressionstest:
  - Test nach Erweiterung, Änderung und Korrektur von Code
- Systemtest:
  - Test der Funktionalität des Gesamtsystems
- Akzeptanztest:
  - Test durch Kunden ob das Programm richtig arbeitet
- u.v.a.m

Vorgehen beim Entwicklertest

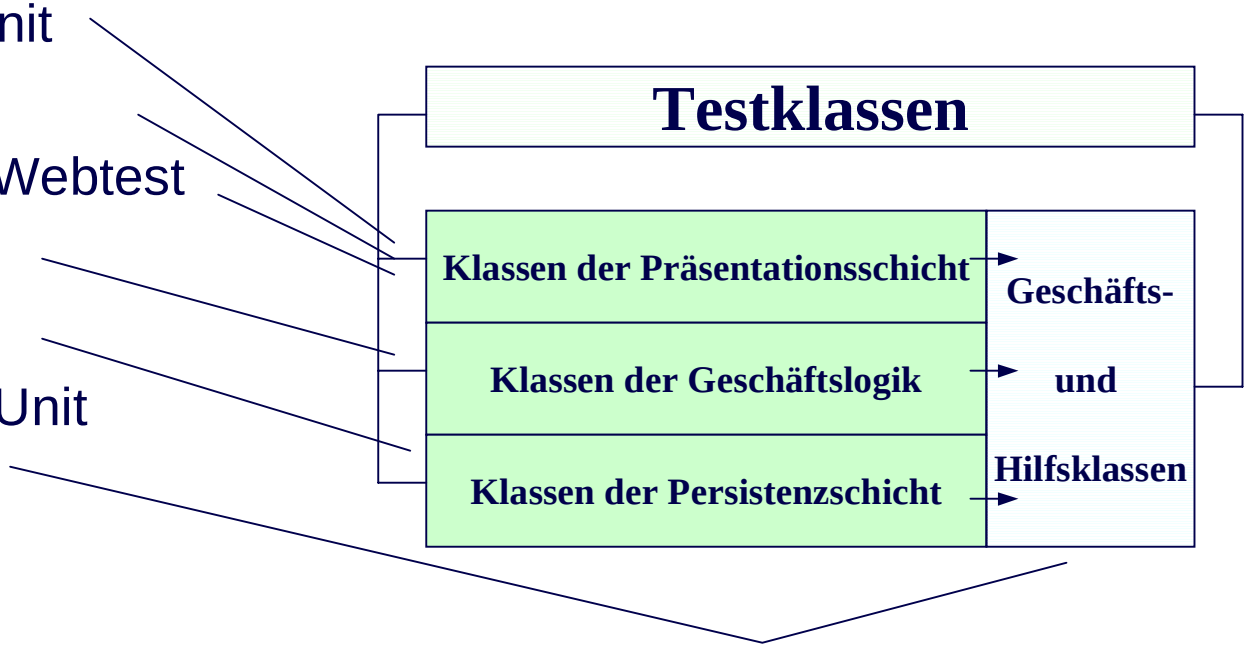
- Mindestens eine Testklasse pro Anwendungsklasse
- Auswahl geeigneter Eingabedaten (Testfallentwurf)
  - Äquivalenzklassen
  - Grenzwertanalyse
    - Nullreferenzen, leere Strings, Zahl 0
    - Kollektionen: leer, ein Element, mehrere Elemente
    - Insb.: gültige und ungültige Werte
- Starten des Tests durch geeigneten Testtreiber (Framework)
- Auswertung des Tests

Vorgehen beim Entwicklertest

- Testorganisation
  - Verwaltung der Testklassen
  - Verwaltung und Erstellung von Testsequenzen
- Testausführung
  - Wie oft und wann werden die Tests ausgeführt?
- Unterstützung durch geeignete Testframeworks zur Automatisierung – Anforderungen:
  - Trennung von Test- und Applikationscode
  - Sprache der Testspezifikation = Programmiersprache
  - Erfolg oder Misserfolg der Testausführung auf einen Blick erkennbar
  - Unterstützung von Testsequenzen

Untersuchte Testframeworks

- Testframeworks sind für verschiedene Schichten eine webbasierten Client-Server Anwendung vorhanden
- 7 Testframeworks wurden (werden) untersucht:
  - HTTPUnit
  - Cactus
  - Canoo Webtest
  - jUnitEE
  - dbUnit
  - jUnit, jxUnit



Erstellung von Testklassen

- Testklasse erbt von frameworkspezifischer Oberklasse
  - jUnit: `TestCase`, dbUnit: `DBTestCase` Cactus: `ServletTestCase`, `JspTestCase`
- Methoden zum erzeugen und zerstören einer Objektstruktur (Fixture)
  - jUnit: `setUp()`, `tearDown()`
- Überschreiben oder Implementation von Methoden
  - dbUnit: `getConnection()`, `getDataSet()`
- Implementation jedes Testfalles in einer eigenen Methode
  - `public void testXXX()`
- Starten des Testframeworks mit Testklasse als Argument
  - `java junit.textui.TestRunner test.ExampleTest`

Probleme bei mehrschichtigen Anwendungen

- Anwendungsklassen sind abhängig von einem Applikations-server
  - z.B.: EJBs, Servlets, JSP
  - Ständiges Deployment
- Unit-Tests sind nahezu unmöglich
  - Evtl. mit Attrappen, Mocks
  - Zahlreiche Versuche gescheitert
- Ausführungsgeschwindigkeit der Tests sehr hoch
  - Datenbankzugriffe
- Ausweg:
  - Nutzung geeigneter Testframeworks und -strategien
  - Testlauf in sog. „Nightly Builds“

Erste Bewertung und Einschätzung

- Vor allem die Nutzung von Applikationsservern macht das Testen von webbasierten Anwendungen schwierig
  - großer Zeitaufwand, Abhängigkeit vom Server, Deployment
  - Testen im Container z.B. mit Cactus
- Unit-Tests beschränken sich auf autonome Klassen
  - jUnit zum Test sehr nützlich
- Test von EJBs erfordert meist vorhandene Datenbankeinträge
  - dbUnit ist hierfür sehr nützlich
- Problem der Integration mit Frameworks und definierten Umgebungen für Klassen bleibt auch in Zukunft bestehen
  - Entwicklung für Werkzeuge zum Test

Bisher Erreichtes

- Literaturrecherche
- Einarbeitung in ProjectWeb
- Schreiben einer ersten Version des Großen Beleges (theoretischer Teil)
- Genaue Untersuchung der Testframeworks jUnit, jxUnit, dbUnit, jUnitEE
- Implementierung von Testfällen für EJBs (Bsp.: EmailManagement) unter Berücksichtigung verschiedener Teststrategien
- Beginn der Erstellung der Testumgebung (Konzept)

Ausblick

- Implementation der Testumgebung auf Grundlage der vorhandenen
  - Einfache Erstellung von Tests für SessionBeans
  - Überarbeitung der Loggingfunktion
- Genaue Untersuchung der restlichen Frameworks
- Erstellung des Nightly Builds für die Ausführung der Tests

Vielen Dank für Ihre Aufmerksamkeit.

Fragen & Diskussion

Literatur

- Glenford J. Myers. Methodisches Testen von Programmen, Oldenbourg Verlag, München, 1982
- Johannes Link, Peter Fröhlich. Unit Tests mit Java, Der Test-First-Ansatz, dpunkt-Verlag, Heidelberg, 2002
- Robert V. Binder. Testing Object-Oriented Systems, Models, Patterns and Tools, Addison Wesley, Massachusetts, 2000

- URLs:
  - http://www.testing.com
  - http://www.mockobjects.com
  - http://www.xprogramming.com
  - http://projectweb.inf.tu-dresden.de