



Die Komplexität bei der Entwicklung und der Pflege von JEE Webanwendungen wird häufig unterschätzt.

JEE Webanwendungen, die sich in einer permanenten technischen und funktionalen Weiterentwicklung befinden, führen oft zu überhöhten, schwer planbaren Entwicklungs-, Test- und Betriebsaufwänden und somit zu erheblichen Beeinträchtigungen ihrer Eigenschaften bezüglich Time-to-Market und Total-Cost-of-Ownership. Diese ungünstigen Eigenschaften sind durch häufig wiederkehrende Defizite in Webanwendungen bedingt.

TYPISCHE DEFIZITE UND IHRE FOLGEN

Viele Webanwendungen weisen hohe *Redundanzen* auf. Oft besitzen bereits einfache Webanwendungen vermeidbare fein granulare Redundanzen. Prominente Beispiele finden sich bei der Ein- und Ausgabekontvertierung einer Webanwendung. Die Darstellung eines grundlegenden Datentyps, wie zum Beispiel eines Datums oder eines Geldbetrags, ist häufig mehrfach, redundant in einer Anwendung realisiert.

In komplexen Webanwendungen, vor allem aber in Anwendungslandschaften, existieren häufig zusätzlich grob granulare Redundanzen. Diese können Teile von Eingabemasken bzw. vollständige Eingabemasken bis hin zu Maskenfolgen inkl. der zugehörigen Konvertierungen und Geschäftslogik beinhalten. Beispielsweise wird die Erfassung einer auf Gültigkeit geprüften Kundenadresse häufig in mehreren Anwendungen einer Anwendungslandschaft redundant realisiert.

Wird eine Änderung an einer redundanten Funktion nötig, muss diese an mehreren statt an nur einer Stelle nachvollzogen werden. Das führt zur Erhöhung der Projektkosten und Verringerung der Kostentransparenz.

Fachliche Funktionsbausteine werden häufig mit einer ungeeigneten Abgrenzung zu anderen Bausteinen entworfen und realisiert. Hieraus resultiert eine hohe *Kopplung* zwischen den

Bausteinen einer Anwendung, häufig mit einer geringen funktionalen Kohäsion der Bausteine einhergehend.

Die Missachtung des Prinzips von „Low coupling, high cohesion“ führt zu einer monolithischen Bauweise bei Webanwendungen. Monolithen vereinen eine Vielzahl negativer Eigenschaften, darunter schlechte Erweiter- und Änderbarkeit und hohe Fehleranfälligkeit.

CHANCEN DER MODERNISIERUNG

Moderne Frameworks, wie JSF (Java Server Faces) oder Wicket, haben viele konzeptionelle Schwächen ihrer „Vorgänger“ abgelegt und bieten zudem eine große Basis wieder verwendbarer Standardkomponenten und die weitestgehend unkomplizierte Integration von Ajax.

Grob und fein granulare Redundanzen können einfacher vermieden werden, die Einhaltung des Prinzips „Low coupling, high cohesion“ wird durch den komponentenbasierten Ansatz moderner Frameworks optimal unterstützt.

Ausserdem lassen sich die Entwicklungskosten für neue Funktionen reduzieren.

Die steigenden Kosten bei der Weiterentwicklung und Pflege, die Möglichkeit, Kosten mittelfristig durch den Einsatz eines moderneren Frame-

works senken zu können und der gestiegene Anspruch bezüglich der Benutzerfreundlichkeit von Webanwendungen, insbesondere im Hinblick auf die Interaktivität (Richness) der Anwendungen, machen eine Modernisierung veralteter JEE Webanwendungen attraktiv.

WOHIN GEHT DIE REISE

Die meisten JEE-Webanwendungen, die in dieser Dekade entstanden, wurden sicherlich auf Basis von Struts realisiert. Deshalb kann davon ausgegangen werden, dass der höchste Modernisierungsbedarf bei Struts-Anwendungen vorliegt. Als Zielplattform wird wahrscheinlich ein GUI-Komponenten-basiertes Webframework wie JSF oder Wicket gewählt werden.

HERAUSFORDERUNGEN DER MODERNISIERUNG

Der typische Sourcecode von JEE Webanwendungen ist stark mit dem verwendeten Framework verwoben oder zumindest strukturell durch dieses geprägt. Deshalb scheint die Modernisierung einer Webanwendung zunächst gleichbedeutend mit der kompletten Neuimplementierung der Gesamtanwendung zu sein, da sich existierender Sourcecode nur im geringen Maße in der modernisierten Anwendung weiter verwenden lässt.

Betrachtet man Risiko und zu erwartende Kosten beim Nachbau einer kompletten Anwendung, so wird deutlich, dass eine derartige Modernisierung nur in sehr seltenen Fällen empfehlenswert ist. Häufiger wird eine Modernisierungsstrategie benötigt, die eine schritt- und ggf. nur teilweise Umstellung auf ein moderneres Framework erlaubt. Dabei wirkt begünstigend, wenn bereits existierende Funktionen der Altanwendung beibehalten werden können und lediglich

bei akutem Bedarf auf das neue Framework umgestellt werden müssen. Gleichzeitig muss es möglich sein, neue Funktionen mit dem neuen Framework zu realisieren und diese nahtlos mit der Funktionen der Altanwendung zu integrieren.

Auf diese Weise werden wichtige Erleichterungen bei der Modernisierung erreicht. Der Lebenszyklus stabiler Funktionen der Altanwendung kann trotz der generellen Modernisierung verlängert werden. Der Umfang der akut zu modernisierenden Funktionen wird reduziert. Zur Realisierung neuer Funktionen kann auf ein neueres, effizienteres Framework zugegriffen werden. Somit sinken Fehlerrisiko und Kosten der Modernisierung.

ERFOLGSFAKTOREN BEI DER MODERNISIERUNG

Ein wichtiger Erfolgsfaktor bei der Modernisierung einer veralteten Webanwendung ist die Integrationsfähigkeit zwischen dem alten und neuen Webframework. Es muss möglich sein, Funktionen der Altanwendung weitestgehend unverändert mit neuen, modernen Funktionen zu integrieren.

Funktionen der Altanwendung und neue Funktionen müssen mit geringem Aufwand und Risiko wechselseitig nutz- bzw. aufrufbar gemacht werden können. Hierbei wirkt sich zusätzlich begünstigend aus, wenn die Möglichkeit existiert, Funktionen der Altanwendung durch wenige Anpassungen als GUI-Komponente wieder verwendbar zu machen.

Betrachtet man beispielsweise die Modernisierung einer Struts-Anwendung auf das Zielframework Wicket, so sollte zumindest eine der folgenden Eigenschaften erfüllt sein:

- Ein typischer Struts-Action-Flow muss sich wie eine abgeschlossene Wicket-Komponente in die neue Anwendung einbetten lassen

- Eine neue Wicket-Komponente muss sich aus einem existierenden Struts-Action-Flow heraus aufrufen und einbetten lassen

LEICHTGEWICHTIGE INTEGRATION MIT ZONING

Zoning ist ein leichtgewichtiges Webframework, das die Integration von Frontendfunktionen, die mit unterschiedlichen Frameworks erstellt wurden, vereinfachen soll.

Der zentrale Einsatzzweck von Zoning ist die risikoarme Restrukturierung und Modernisierung klassischer JEE Webanwendungen. Deswegen wurde bei der Konzeption der größte Wert auf die Technologieneutralität gelegt. Zoning steht deshalb in keiner Konkurrenz zu anderen Webframeworks, sondern soll diese lediglich um eine einheitliche Integrationsmöglichkeit mit anderen Webframeworks ergänzen.

Zoning ermöglicht die schritt- und teilweise Modernisierung von Anwendungen. Durch die Unterstützung bei der Ausbildung wieder verwendbarer Komponenten aus monolithischen Anwendungen, können mit Zoning grob granulare Redundanzen verringert werden und so zu einer Konsolidierung kompletter Anwendungslandschaften beigetragen werden.

TECHNISCHE FUNKTIONSWEISE UND FEATURES

Zoning basiert komplett auf gängigen Standards in der JEE Entwicklung. Die Zoning-Core-Komponente basiert ausschließlich auf der Servletspezifikation und ist somit auf allen gängigen Servletengines einsetzbar. Die Features im Überblick:

- Multiversionsfähigkeit: Komponenten können zur Laufzeit in mehrere Versionen deployed werden, so dass deren Austausch ohne Downtime möglich wird
- Komponentisierung: Unterstützung für die Erstellung komponensierter Web-Anwendungen auch mit Web-Frameworks, die diese Funktionalität nicht selbst mitbringen (z B. Struts 2)
- Einfache Integration von Web-Applikationen: Bestehende und neue Web-Applikationen können zu einer virtuellen Applikation integriert werden
- Clusterfähigkeit der virtuellen Applikationen wird unterstützt
- Technologieunabhängigkeit: Unterschiedliche, alte und neuere Web-Frameworks können in virtuellen Applikationen miteinander integriert werden. Diese technologische Kompatibilität von unterschiedlichen Web-Frameworks ermöglicht eine schrittweise Migration von Alt-Anwendungen (z. B. durch Entkernung Funktion für Funktion)
- Content Aggregation: Inhalte/Use-Cases einer Seite können, ähnlich Portalen, auf einer Seite zusammengefasst werden
- Geschäftsprozesse/Dialogverschaltung: Dialoge können über mehrere Web-Applikationen hinweg miteinander verschaltet werden. Auch bestehende Use-Cases können auf diese Weise integriert werden
- Global Session Scope: Der globale Session Scope oder Sitzungskontext umspannt die integrierten Web-Applikationen und ermöglicht auf diese Weise ein Single-Sign-On (SSO)