Oracle ADF Web-GUI für Legacy Systeme mit Reuse der bestehenden Business Logik

Peter Gübeli, Helsana Versicherungen AG
Timo Hahn, virtual7 GmbH

Schlüsselworte
Oracle ADF, Pillar-Architektur , State-Sharing, Error-Handling, Coherence

Einleitung
Die Helsana-Gruppe ist mit knapp 1,9 Millionen Versicherten der führende Schweizer Kranken- und Unfallversicherer. Sie steht Privaten und Unternehmen bei Gesundheit und Vorsorge sowie im Fall von Krankheit und Unfall umfassend zur Seite. Mit Prämieneinnahmen von 5,7 Milliarden Franken belegt das Unternehmen eine Spitzenposition im Schweizer Versicherungsmarkt.
Zur Verarbeitung der Kernprozesse steht im Bereich der Leistungsabrechnung eine Standardsoftware zur Verfügung. Die restlichen Verarbeitungen wie z.B. die Produktgestaltung und die Policierung basieren auf einer Cobol Eigenentwicklung welche auf einem IBM zOS Mainframe betrieben wird. Mittelfristig sollen alle Kernprozesse in einer Standardsoftware abgewickelt werden.
Bis dahin gilt es, die in der Eigenentwicklung integrierten und optimierten Businessprozesse kostengünstig zu betreiben und dem Endkunden darauf basierend ein modernes und komfortables Benutzerinterface zu bieten. Ein rascher ROI ist also bei allen weiteren Entwicklungen unabdingbar. Mit der Kombination neues Web-GUI auf Basis der Oracle ADF-Technologie angebunden an die existierende Businesslogik und Daten auf dem Mainframe System erreichen wir dies optimal.
Die Umsetzung findet unter Verwendung der Pillar-Architektur statt, die es ermöglicht einzelne Anwendungen separat zu entwickeln, diese aber für den Anwender als eine gemeinsame Anwendung darzustellen.
Die Pillar-Architekur gestattet den einzelnen Entwicklerteams die gemeinsame Nutzung von Kernkomponenten wie z.b. ExceptionHandling, Logging, Zugriffschicht auf das Legacy-System oder Security, erlaubt aber eine flexible Entwicklung der Anwendungslogic und Web GUIs.
Der Vortrag zeigt die gewählte Architektur und gibt Aufschluss über deren Implementierung anhand einer kurzen Demonstration.

Grundlage: Die Pillar-Architetkur von Oracle
Bei der Architektur von eSAS (Erklärung siehe Glossar) handelt es sich um eine Abwandlung der Pillar-Architektur (auf Deutsch Säulen-Architektur), welche vom Architektur-Team von Oracle ADF entwickelt wurde.

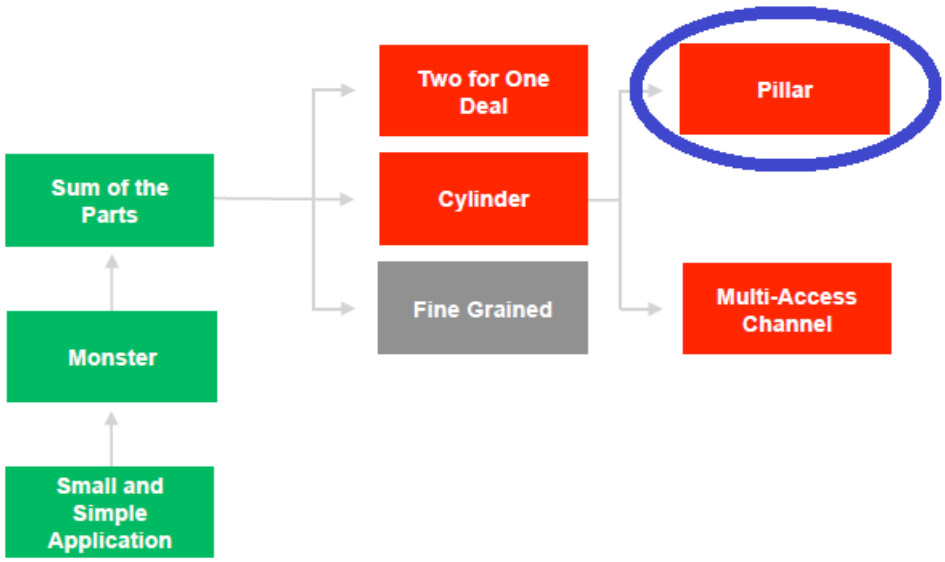

Abb. 1: Schematische Darstellung der Pillar-Architektur

Das Projekt eSAS baut auf den Konzepten der Pillar-Architektur auf und erweitert sie um fehlende Funktionalitäten. Das nachfolgende Diagramm zeigt den grundsätzlichen Aufbau:

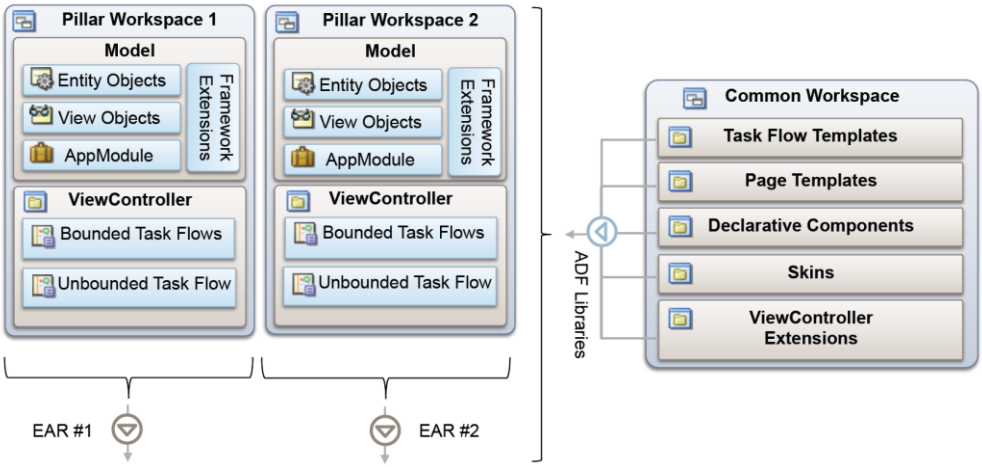

Abb. 2: Schematische Darstellung der Pillar-Architektur

Diese Architektur basiert darauf, dass es einzelne ADF Applikationen für verschiedene Use Cases oder ganze Anwendungsbereiche existieren. Eine solche Applikation ist dabei in einem Workspace zusammengefasst (Pillar Workspace X). Jede dieser Applikationen wird dabei separat auf dem Server deployed. Wiederverwendbare allgemeine Komponenten werden dabei aus dem Common Workspace als ADF Libraries geladen. Dabei handelt es sich um Skins für das Coporate Design, Utility-Klassen, Templates und Framework-Erweiterungen von z.B. ViewObjectImpl oder EntityImpl. Durch das Konzept dieser Wiederverwendbarkeit können neue Applikationen (Pillars) mit einem geringeren Aufwand hinzugefügt werden und haben das gleiche Erscheinungsbild. Die Anwender können somit zwischen den einzelnen Applikationen wechseln, ohne grosse optische Veränderungen wahr zunehmen.
Neben der Wiederverwendbarkeit ist die Unabhängigkeit der einzelnen Pillars voneinander der grösste Vorteil dieser Architektur. In einer ‚traditionellen' ADF-Architektur (z.B. Sum-of-all-Parts Architektur) befinden sich die Business Logik zwar in unterschiedlichen Projekten, die aber alle gemeinsam in einem EAR File deployed werden. Dies führt dazu, dass bei einer Änderung der Business-Logik in einem Bereich, alles neu getestet und deployd werden muss. Dieses Problem ist mit der Verwendung der Pillars, die unabhängig voneinander hinzugefügt, entfernt und geändert werden können, nicht vorhanden.
Aufgrund der Trennung bestehen allerdings auch besondere Anforderungen wie

- Einheitliche UI
- 'Transaction Handling' / 'State Sharing'
- Single Sign On (SSO)
- Session-Timeout Handling

die bei Verwendung der Pillar-Architektur berücksichtigt werden müssen.

Jedoch überwiegen die Vorteile

- Falls eine Applikation hinzugefügt oder geändert wird, hat das keinen Einfluss auf die bestehenden.
- Die Komponenten aus dem Common Workspace müssen nicht für jede ADF Applikation neu programmiert werden.
- Die Applikationen können unabhängig voneinander deployed werden (unterschiedliche Versionen möglich). Zudem können die Applikationen auch in unterschiedlichen WLS-Clustern laufen.
- Die Integration von SSO in den einzelnen Applikationen ist aufgrund einer bestehenden Lösung kein Problem.

Die eSAS Architektur
Im Vorfeld der Entwicklung wurde eine Benutzeroberfläche entworfen, die allen Pillar-Anwendungen zugrunde liegt. Im Bild unten ist diese skizziert:

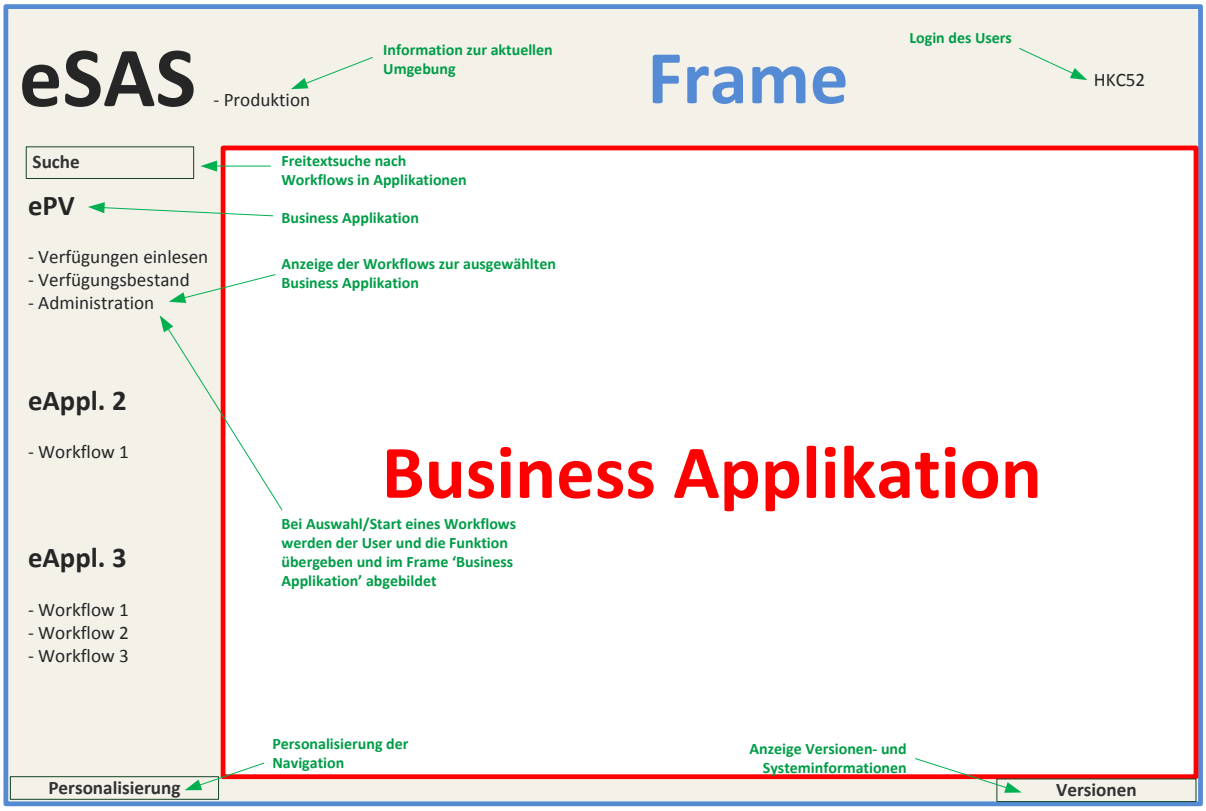

Abb. 3: Schematische Darstellung der Benutzeroberfläche

Wie auf der Abbildung zu sehen ist, besteht das Layout aus einem äußerem Frame, in welchem der Helsana Header und eine Navigation abgebildet ist. Je nach dem welcher Punkt auf der Navigation ausgewählt ist, ändert sich die Anzeige im inneren Frame, im Bereich „Business Applikation“. Dieser beschreibt den dynamischen Teil der Struktur und dient den Pillar-Anwendungen als Rahmen. Für diesen Rahmen wurde unter dem Begriff eSAS eine passende Architektur definiert, welche nicht nur den sichtbaren Teil, sondern auch die dahinter liegenden Strukturen und Abläufe vereinheitlicht und somit einen technischen Rahmen für zukünftige ADF Projekte im SAS Umfeld bereitstellt. Als Basis für diese Architektur dienen dabei folgende Kriterien:

- Einheitliche Erscheinung der ADF Applikationen im Frontend
- Auf ADF Best Practices basieren
- Flexible Erweiterung der Architektur um weitere Applikationen und Use Cases
- Wiederverwendbarkeit von allgemeinen Komponenten

Umsetzung der eSAS Architektur
Die Umsetzung der entworfenen Architektur ist im Bild unten schematisch dargestellt.

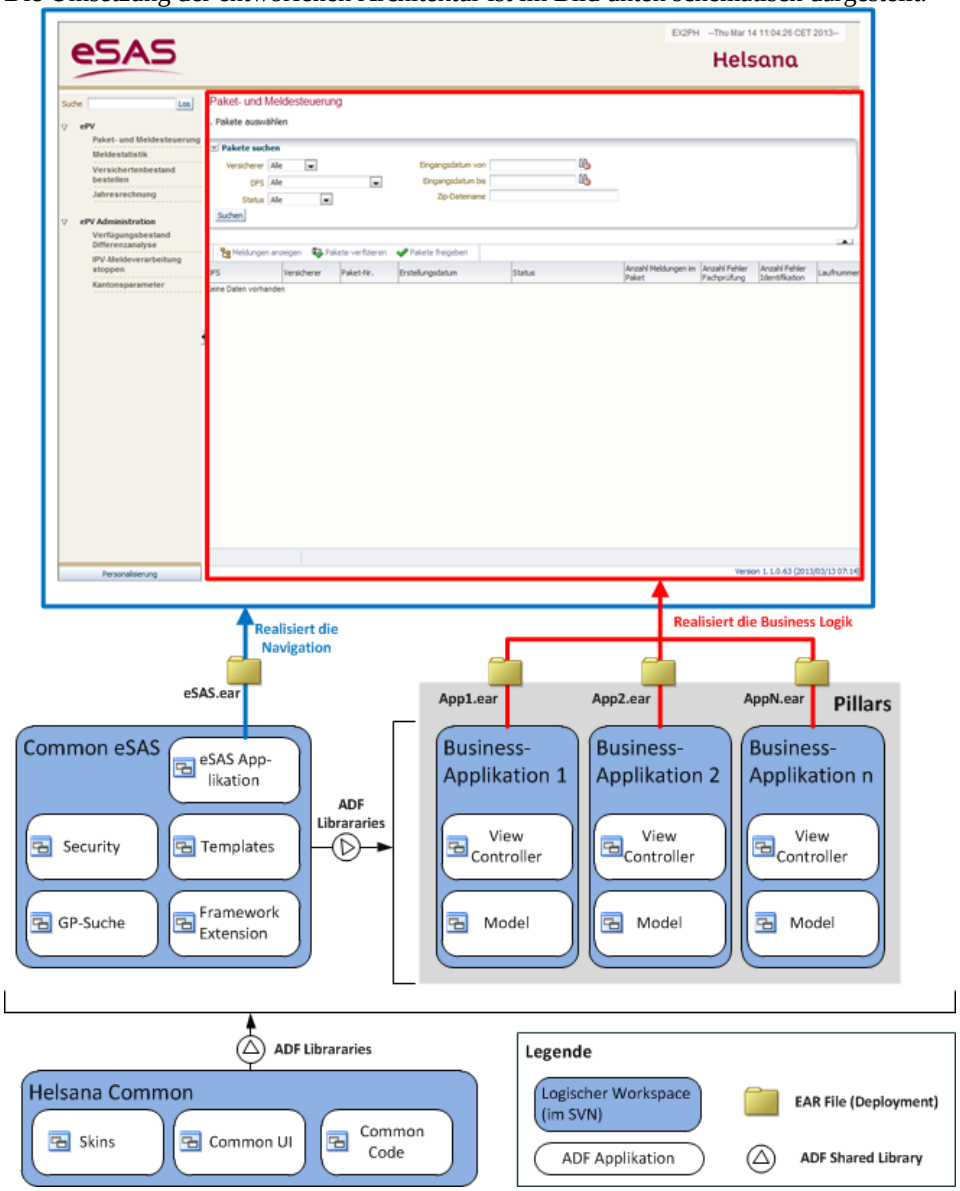

Abb. 4:Umsetzung der Schematische Darstellung der Benutzeroberfläche in einer Anwendung

Auf der Grafik ist oben zu sehen, dass die im Browser ersichtliche Seite aus zwei Bereichen besteht: Blau markiert ist die eSAS Applikation, die als Rahmenanwendung für die Navigation zwischen den Pillar-Anwendungen und als 'Single Point Of Entry' dient. Der rot markiert Bereich ist den einzelnen Business-Applikationen (realisiert in den einzelnen Pillar Workspaces - Business Applikation 1 - n) vorbehalten.
Bei der eSAS Application handelt es sich um eine eigenständige Anwendung, die einen Header, eine Navigation und ein I-Frame (roter Bereich) enthält. Die Navigation wird dabei dynamisch, personalisiert aus einer Datenbank geladen. Hinter jedem Navigationspunkt verbirgt sich eine URL welche als Sprungziel auf die unterschiedlichen Business-Applikationen dient. Wird nun ein Navigationspunkt angeklickt, dann erscheint im I-Frame (Rot markiert) eine der Pillar-Applikationen. Dadurch wird das Hauptziel der Pillar-Architektur erreicht: Der Anwender wechselt zwischen unterschiedlichen Pillar-Applikationen ohne dies zu merken, da nur der Inhalt im inneren Frame getauscht wird.
Wird nun eine Applikation geändert (z.B. um einen weiteren Navigationspunkt ergänzt) oder neu erstellt, müssen die eSAS Applikation und die anderen Applikationen nicht geändert bzw. neu deployed werden. Es genügt die Datenbank-Tabelle, die die Navigation beinhaltet, um die neuen Navigationspunkte und URL's zu ergänzen.

Neben der Rahmenanwendung werden auch allgemeine Abläufe und Strukturen als ADF Libraries in eSAS bereitgestellt. Dazu zählen

- Security: speziell angepasste mandantenfähige Security Lösung welche auch die Berechtigungen des Altsystems berücksichtigt
- Task-Flow-Templates: Stellen ein einheitliches ErrorHandling bereit
- Page-Templates: einheitliches Seitenlayout und Session-Timeout Handling
- Framework-Erweiterungen: spezielle Erweiterungen des ModelLayers für den Zugriff auf IBM DB2 mittels WebServices
- State-Sharing und Transaction-Handling: Coherence

Error-Handling
Das Error-Handling innerhalb der Pillar-Architektur unterscheidet sich vom normalen Error-Handling, da im Prinzip mehrere Anwendungen gleichzeitig laufen und Exceptions in allen von diesen auftreten können. So kann z.B. eine Exeption in einer Pillar-Anwendung auftreten, die nur in dieser Anwendung behandelt werden kann und soll. Es ist aber auch möglich, dass eine Exception zu einem Fatalen Fehler führt, der die Gesamte Pillar-Architektur, also auch die Rahmen-Anwendung betrifft. In diesem Fall muss die Exeption an die Rahmen-Anwendung weitergegeben werden, damit diese entsprechend reagieren kann.

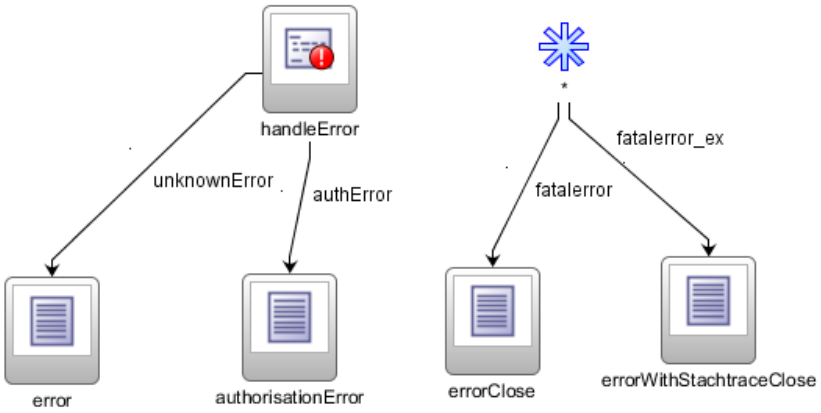

Abb. 5: Unbounded-Task-Flow Template

Das Bild oben beschreibt das Error-Handling der Rahmen-Anwendung, das Bild unten für Bounded-Task-Flows (BTF) die in einer Pillar-Anwendung verwendend werden. Sollte ein fataler Fehler in einem BTF auftreten, wird dieser abgefangen und als 'fatalerror' weitergereicht. Dies führt dann in der Rahmen-Anwendung dazu, dass eine Fehlerseite angezeigt wird, die nur noch das Schließen der Anwendung erlaubt.

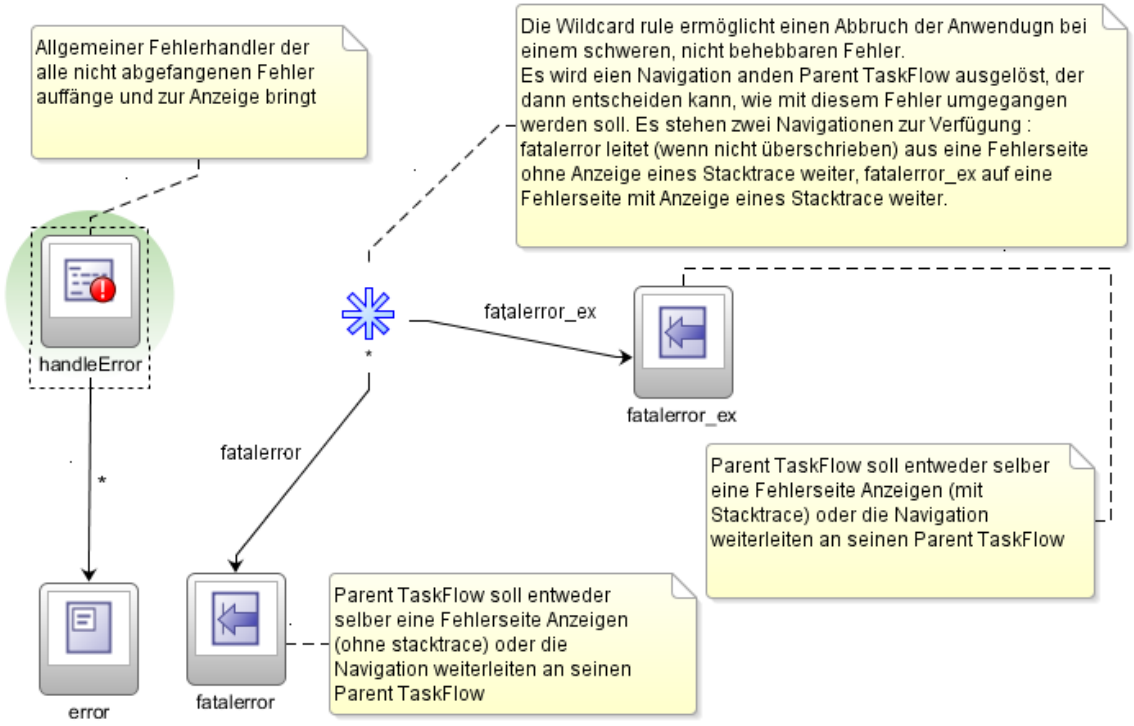

Abb. 6: Bounded-Task-Flow Template

Session-Timeout Handling

Da zu einem Zeitpunkt immer zwei Anwendungen aktiv sind, die jede ihre eigene Session hat, können diese Sessions auch zu unterschiedlichen Zeiten ablaufen. Dies führt zum unschönen Effekt, das die Session-Ablaufwarnung mit der Meldung, das die Session abgelaufen ist doppelt auftreten können. Dies ist nicht weiter schlimm, führt aber bei den Anwendern zu Verwirrung.

Weiterhin ist zu beachten, dass beim Umschalten auf eine andere Pillar-Anwendung eine weitere Session hinzukommen kann, falls zu dieser Anwendung noch keine Session bestand. Dies führt dann zu einem erhöhtem Bedarf an Sessions und den damit verbundenen Ressourcen (z.B. Connections). Um diese Problematik zu umgehen, werden die Timeouts der Pillars auf einen kurzen Zeitraum eingestellt (z.B. 5 min), die der Rahmenanwendung auf einen längeren Zeitraum (z.B. 60 min). Damit die Pillar-Anwendung nicht nach 5 min Untätigkeit abläuft, wird über eine Poll-Komponente kurz vor Ablauf ein Refresh ausgelöst. Erst wenn die Rahmen-Anwendung abläuft unterbleibt auch der Refresh für die Pillar-Anwendung, die dann nach ihrem eingestellten Timeout ebenfalls abläuft.

Wird die Pillar-Anwendung gewechselt, unterbleibt auch der Refresh und die Session läuft kurze Zeit später ab.

State-Sharing und Transaction-Handling

Besondere Aufmerksamkeit muss dem Start-Sharing und Transaction-Handling gewidmet werden. Im Prinzip laufen mehrere Anwendungen parallel, die sich aber keine Transaktion teilen, sondern jede Anwendung eine eigene Transaktion besitzt. Vorteil davon ist, dass die Anwendungen auf unterschiedlichen Schema bzw. DBs arbeiten können.

Problematisch wird es allerdings, wenn eine Pillar-Anwendung Daten geändert hat, dann aber über die Rahmen-Anwendung auf eine andere Pillar-Anwendung gewechselt werden soll.

Die normale 'isTransactionDirty' Warnung funktioniert hier nicht, da die Aktion in der Rahmen-Anwendung ausgelöst wurde, die ja keine Änderung gemacht hat. Es muss also eine Möglichkeit geschaffen werden, dass die Pillar-Anwendungen ihren Zustand der Rahmenanwendung mitteilen können. Zu diesem Zweck wird Coherence eingesetzt. Es wird in Coherence für jeden User eine globale Session gehalten, in der die Pillar-Anwendungen ihren Zustand hinterlegen. Vor einem Wechsel werden diese Informationen abgefragt und bei nicht gesicherten Daten eine Warnung an den Anwender ausgegeben. Dies wird durch den Einbau eines speziellen DCTransactionStateListener erreicht, den jede Pillar-Anwendung in ihrer Startseite einbaut. Die

Implementierung des DCTransactionStateListener ist in den Task-Flow-Templates hinterlegt und wird mit diesen vererbt.
Wegen der hohen Komplexität der Security der Altanwendungen wird auch der eSAS Security Context nicht von jeder Applikation separat geladen, sondern global zur Verfügung gestellt. Dies wird mit der Anmeldung an die Rahmen-Anwendung vollzogen.

Glossar

SAS SANAswiss
In Cobol entwickeltes Kernsystem der Helsana Gruppe zur Abwicklung der Versicherungsprozesse. Die Userinteraktion erfolgt via 32/70 Terminal Software

eSAS easy SANAswiss
bestehendes SANAswiss erweitert mit teilweiser Ablösung der Terminaluserinteraktion durch ein ADF-WEB-GUI

Referenzen

ADF Architectural Fundamentals - Angels in the Architecture
Task Flow Design Fundamentals
Coherence Guide.docx

Kontaktadressen

Peter Gübeli
Helsana Versicherungen AG
Postfach
CH-8081 Zürich

Telefon: +41 43 340 51 05
Fax: +41 430 340 01 11
E-Mail: peter.guebeli@helsana.ch
Internet : www.helsana.ch

Timo Hahn
Virtual7 GmbH
Zeppelin Strasse, 2
D-76185 Karlsruhe

Telefon: +49 (721) 619 017 59
Fax: +49 (721) 619 017 29
E-Mail hahn@virtual7.de
Internet: www.virtual7.de