Industrialisierte Oberflächenentwicklung mit Oracle ADF

Ralf Ernst, Bundesagentur für Arbeit, und Andreas Koop, enpit consulting OHG

Während sich in den letzten Jahren im Bereich des Backend eine immer effizientere und industrialisierte Entwicklung von Komponenten auf Basis diverser etablierter Frameworks wie Spring, EJB3, JPA oder Web-Services etabliert hat, folgt die Entwicklung von graphischen Benutzeroberflächen (GUIs) immer noch vielfach einem eher individuellen „Arts and Craft"-Anspruch. Eine Industrialisierung der Entwicklung von GUIs ist aber ebenso notwendig wie machbar.

Voraussetzung für die Industrialisierung der Entwicklung von GUIs sind weitgehend die Standardisierung der Benutzeroberfläche, eine verbindliche Referenzarchitektur und die Automatisierung wiederkehrender Gestaltungsaufgaben. Dieser Artikel beschreibt Motivation, Voraussetzungen und Vorgehen für die Einführung einer industrialisierten Oberflächenentwicklung bei der Bundesagentur für Arbeit (BA) auf Basis von Oracle ADF.

Das IT-Systemhaus der BA betreibt eine der größten IT-Systemlandschaften Deutschlands. Neben Standardsoftware für Client-zentrierte Office-Funktionalitäten, einem ERP-System für Auszahlungen und Verwaltung sowie einer Business- Intelligence-Plattform für statistische Auswertungen zählen hierzu rund 100 zentralisierte Applikationen mit zusammen mehr als 10.000 Masken auf Basis von Java EE beziehungsweise C++/Corba, die die spezielle Geschäftslogik der BA und somit ihr eigentliches Kerngeschäft implementieren. Die mehr als 100.000 Benutzer müssen zur Erledigung ihrer Geschäftsprozesse oft eine Vielzahl von Anwendungen nutzen.

Neben der Neuentwicklung sogenannter „Rollen-basierter Oberflächen" für spezielle Geschäftsprozesse in Service-Centern und Eingangszonen mussten verschiedene geschäftskritische ältere Verfahren auf Basis von C++/Corba und Java EE modernisiert und aktuelle Anforderungen der Fachseite mit Neuentwicklungen realisiert werden. Eine Analyse der bestehenden GUI-Architekturen und Entwicklungsprozesse ergab folgende Ergebnisse:

- Auch durch den Einsatz heterogener Technologien und Frameworks (C++, Swing, JSP, JSF) bedingt, existierten weder ein einheitliches „Look & Feel" noch einheitliche Bedienmuster der BA-Anwendungen. Dies erschwerte den Benutzern die Aufgabenerledigung erheblich und forderte von ihnen einen teilweise erheblichen Einarbeitungsaufwand.
- Auf Entwicklungsseite existierte im Gegensatz zum Backend für die Entwicklung von GUIs weder eine gültige Referenz-Architektur noch ein definierter Entwicklungsprozess. Entwickler von webbasierten GUIs mussten oft eine Vielzahl von Technologien und Frameworks beherrschen und integrieren (wie Struts, JSP, JavaScript, CSS, HTML).
- Die Ziel-Architektur aktueller Web-Frameworks unterscheidet sich in hohem Maße voneinander. Vergleicht man die Architektur von beispielswei-

se JavaScript-basiertem HTML5, JavaFX oder JSF, findet man kaum Gemeinsamkeiten. Im Gegensatz zu tieferen Applikationsschichten hat sich auch noch kein definierter Industriestandard entwickelt, was schon die schier unüberblickbare Vielfalt von GUI-Frameworks und Technologien verdeutlicht. Daher fehlen vielfach auch Entwicklungsmuster und Best Practices zu Themen wie Data-Binding, State-Handling oder Event-Processing.

Erwartungen an Benutzeroberflächen und GUI-Technologien

Die Erwartungen von Benutzern, IT-Architekten und dem Management an Benutzeroberflächen und deren Technologien sind sehr unterschiedlich. Benutzer vergleichen heutzutage die Oberfläche ihrer Unternehmensanwendung mit Anwendungen, die sie tagtäglich im Internet benutzen. Im Zuge dieser „Consumerization of Information Technology" definiert das Internet also mittlerweile, wie eine moderne Benutzeroberfläche auszusehen hat. Ein populäres Beispiel hierfür ist die einfache Google-Suche und deren Auto-Vervollständigungsmechanismus.

Ein weiterer Begriff, der in diesem Zusammenhang gebraucht wird, ist „joy of use". Platt ausgedrückt soll der Anwender bei der Bedienung einer Benutzeroberfläche Freude haben. Die Bedienung einer Anwendung soll also intuitiv und vorhersehbar sein. Anstelle spezieller Lösungen und Tricks sind eine Beschränkung der Anzahl möglicher Bedienmuster (Master-Detail, Autocomplete, etc.) auf das unbedingt Notwendige und eine leicht durchschaubare einheitliche Navigationsstruktur, die sich konsistent in allen Anwendungen wiederfindet, ein Schlüssel für benutzerfreundliche Oberflächen. Schließlich muss eine GUI in der BA den aktuellen Erfordernissen und Gesetzen zur Barrierefreiheit genügen, also auch von (seh-)behinderten Benutzern problemlos bedienbar sein.

IT-Architekten erwarten auch von der GUI-Technologie ein hohes Maß an Wiederverwendbarkeit im Kontext einer ansonsten serviceorientierten Architektur. Dies beinhaltet die Integration mit bestehenden Architekturen ebenso wie die Unterstützung künftig mehr prozessorientierter Applikationen. Eine zukunftsorientierte GUI-Technologie muss im Intranet wie im Internetportal auf verschiedensten Endgeräten lauffähig sein. Eine saubere Trennung und damit Entkopplung der einzelnen Schichten einer Applikation („separation of concerns") sowie ein komponentenorientierter Ansatz sind hierfür Voraussetzungen.

Aus Sicht des Managements bemisst sich der Geschäftswert einer Applikation sehr stark an der Benutzeroberfläche. Aufgrund der in der BA üblichen langen Produktlebenszyklen (ca. 10 Jahre) muss eine GUI-Technologie zukunftssicher und über diesen langen Produktzyklus unterstützbar sein. In dieser Zeit müssen Provisionierung der Anwendung sowie Weiterentwicklung und Wartung kostengünstig möglich sein.

Ein industrialisierter Ansatz für die Oberflächenentwicklung

Industrialisierung bedeutet in der IT nichts anderes als die möglichst weitgehende Standardisierung von Lösungen und die

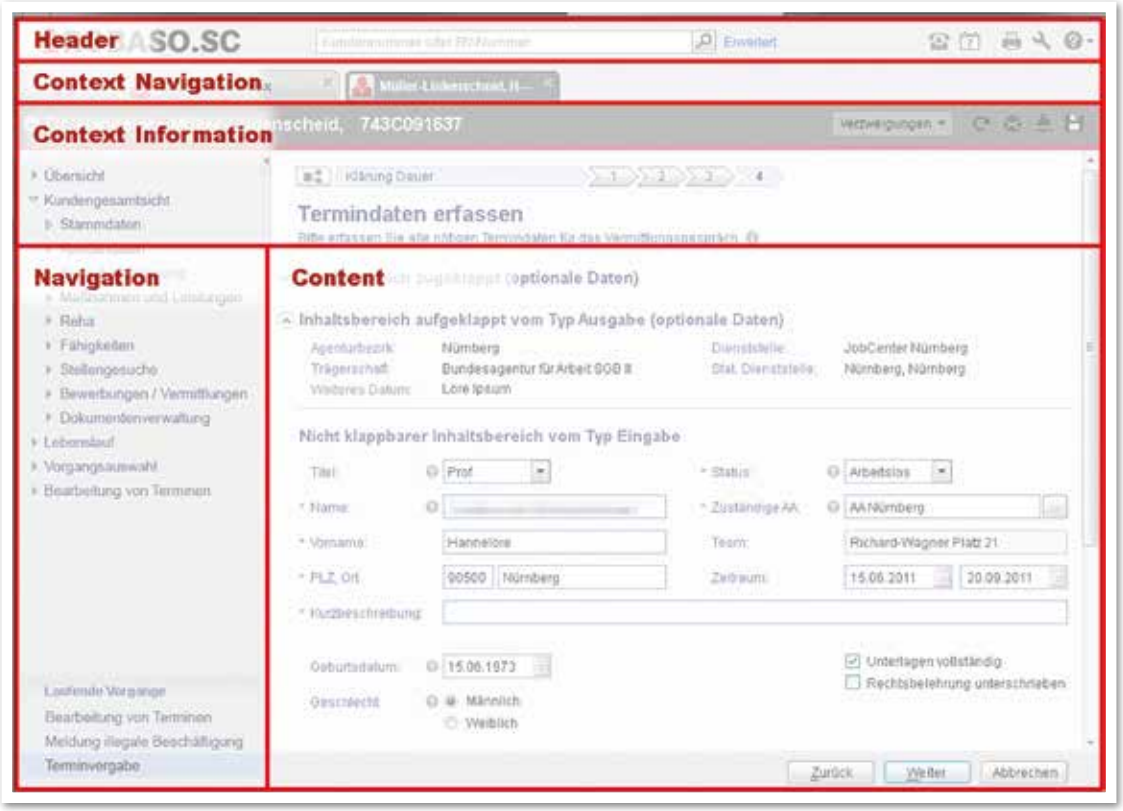

Abbildung 1: Standardisierte Seitenbereiche des BA-Design-Guides

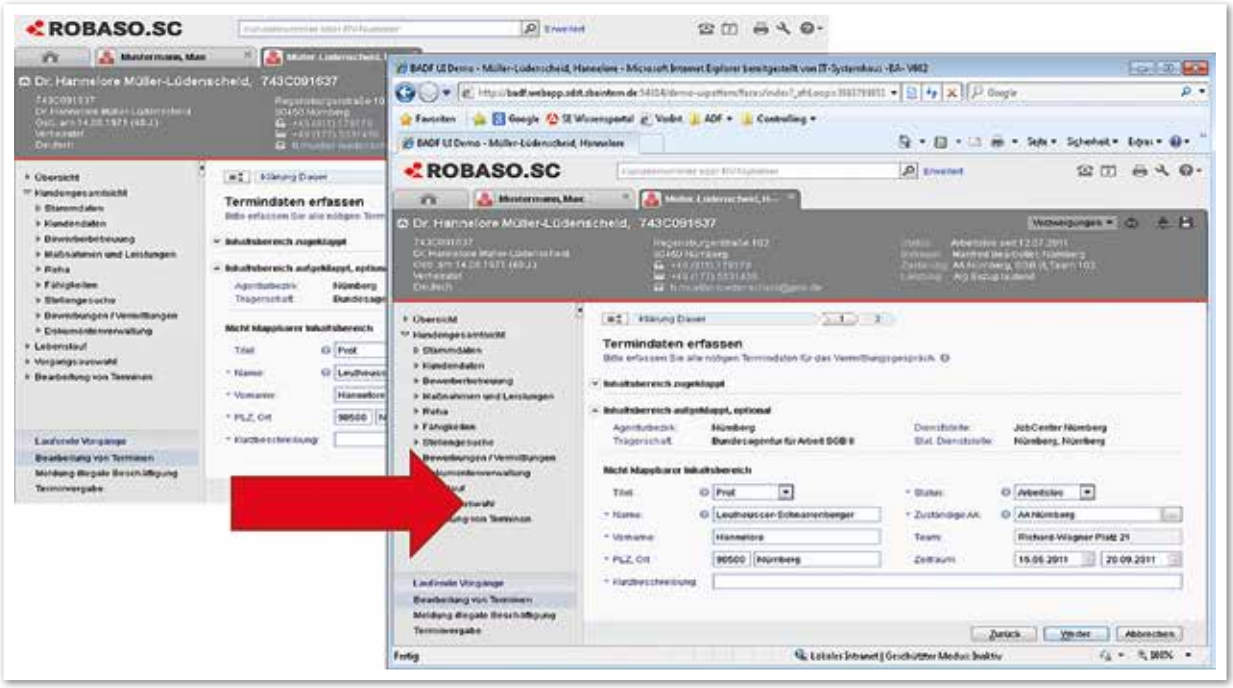

Abbildung 2: Vom Photoshop Screendesign zur ADF-Anwendung

Automatisierung wiederkehrender Aufgaben. Im Kontext der UI-Entwicklung erfolgte die Standardisierung durch die Definition eines sowohl für die IT als auch den Bedarfsträger verbindlichen Design-Guides, der das Aussehen, den Aufbau, die Navigationsstruktur und die möglichen Interaktionselemente festlegte. Daneben erfolgte die Definition einer Referenzarchitektur für UIs und die Produktauswahl des zukünftig einzigen Frameworks für die Implementierung – Oracle ADF.

Um sicherzustellen, dass die entsprechenden Design-Vorgaben auch umgesetzt und nicht im Zuge der Entwicklung neuer Applikationen konterkariert würden, wurden Anreize in Form eines zentralen BA-Skins geschaffen, zentrale Vorlagen und vorgefertigte UI-Komponenten für wiederkehrende Gestaltungsmuster der Masken bereitgestellt. Diese Maßnahmen sollten ein „Baukasten-ADF" (BADF) und eine Referenzarchitektur schaffen, um eine Standardisierung grundlegender Aufgabenstellungen wie Data-Binding, State-Handling und Event-Processing über verschiedene IT-Projekte hinweg zu ermöglichen. Darüber hinaus war ein einheitliches Tooling zu entwickeln, das Deployment, Staging und automatisierte GUI-Tests unterstützt.

Der BA-Design-Guide

Der BA-Design-Guide wurde unter Führung des Bedarfsträgers, also der Fachabteilung der Zentrale der BA, entwickelt. Die Fachabteilung definierte, wie die zukünftigen Benutzeroberflächen – bereitgestellt durch das IT-Systemhaus – aussehen und was diese leisten. Insoweit fungiert der BA-Design-Guide auch als wichtiges Kommunikationsmedium zwischen IT und Fachabteilung für künftige Anwendungen, da er die möglichen UI-Elemente und Interaktionsmuster aller BA-Anwendungen enthält und von der Fachabteilung quasi als Katalog für künftige Anwendungen genutzt werden kann, was die Formulierung von Fachkonzepten und Anforderungen an die IT wesentlich vereinfacht.

Das Team zur Definition des BA-Design-Guides setzte sich neben einem UX-Experten größtenteils aus fachlichen Analysten und Anwendern zusammen. Ziel war die Definition eines allgemein verwendeten

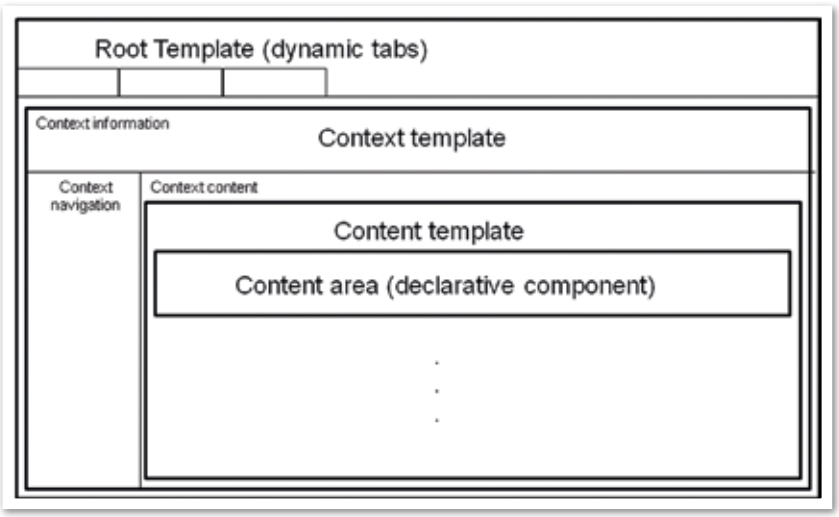

Abbildung 3: Mapping der Seiten-Fragmente auf korrespondierende, wiederverwendbare Templates

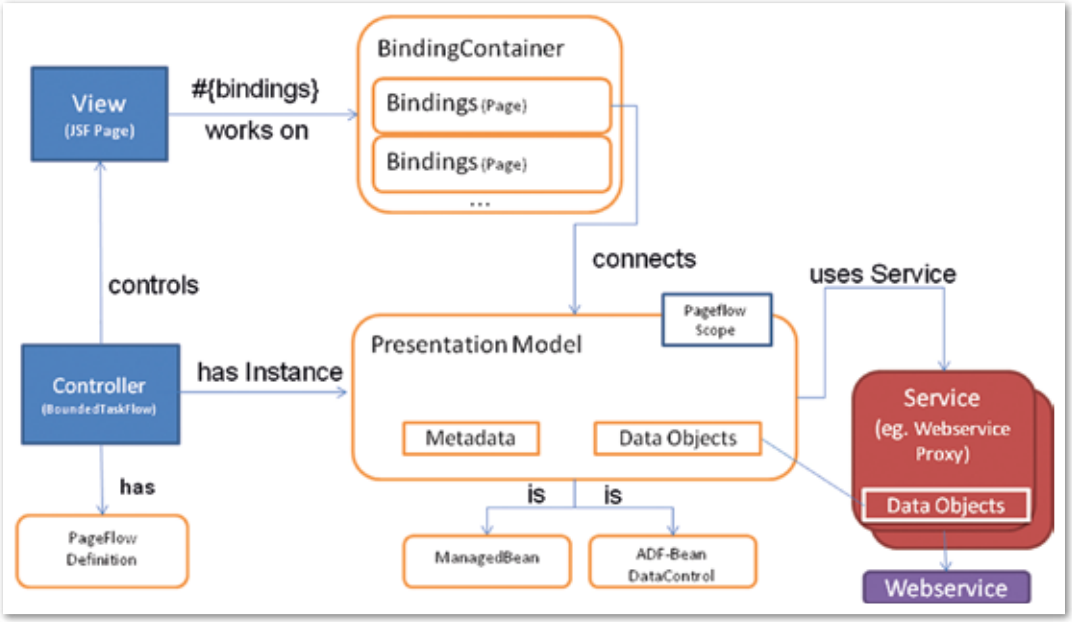

Abbildung 4: POJO-Binding

Anwendungsrahmens und einer generischen Navigationsstruktur sowie die Reduktion möglicher Interaktionsmuster auf das unbedingt notwendige Maß *(siehe Abbildung 1)*.

Begleitet wurde das Team von zwei technischen Experten, die zur generellen technischen Machbarkeit der Vorschläge und Entwürfe Stellung nahmen. Neben allgemeinen Regelungen zur Ergonomie und Barrierefreiheit definiert der BA-Design-Guide nunmehr einen standardisierten Seitenaufbau in Header, Kontext-Navigation, Kontext-Information, Navigationsbereich und Inhaltsbereich sowie ein einheitliches Navigationskonzept.

Die Standardisierung des Inhaltsbereichs erfolgte durch sogenannte „Gestaltungsmuster", die die gewünschte Funktionalität optimal unterstützen. Ziel hierbei war wiederum, so wenige Gestaltungsmuster wie möglich zu verwenden, um für den Anwender konsistente, wiedererkennbare Oberflächen auch über verschiedene Anwendungen hinweg zu garantieren. Gestaltungsmuster definieren die verschiedenen Formulare und Tabellen-Ansichten (wie Master-Detail, Command-Link, 2-spaltiges Formular etc.) innerhalb des Inhaltsbereichs einer Seite.

Darüber hinaus wurden Aussehen und Verhalten von Interaktionskomponenten (input-fields, lists, enumerations, choices, texts, buttons etc.) modifiziert, um die gewünschte Funktionalität zu unterstützen, und zuletzt die gewünschte Darstellung von Meldungs- und Fehlerdialogen zu spezifizieren. Die in Photoshop entworfenen Maskenfragmente wurden dann in korrespondierende ADF-Templates übernommen. Mittels ADF-Skinning war das gewünschte „Look & Feel" exakt realisierbar *(siehe Abbildung 2)*.

Abbildung 5: Bestandteile des BADF-Toolkits

Die Produktauswahl

Als GUI-Framework für die Implementierung des BA-Design-Guides wurde Oracle ADF ausgewählt. Für diese Lösung sprach insbesondere, dass mit Technologien wie wiederverwendbaren und zentral mittels ADF Library provisionierbaren Templates in Verbindung mit Skinning die Vorgaben des Design-Guides nahezu „1:1" umsetzbar waren. Daneben konnte mit dem Konzept der Dynamic-Tabs die Forderung der Fachseite erfüllt werden, verschiedene Kontexte (mehrere Kundendaten, Prozesse etc.) in verschiedenen Bereichen der Anwendung zugleich laufen zu lassen *(siehe Abbildung 3)*.

Daneben war aus Sicht der gewünschten Standardisierung wichtig, dass ADF eines der wenigen GUI-Frameworks am Markt ist, dass in sich komplett und vollständig ist, also ohne zusätzliche Erwei-

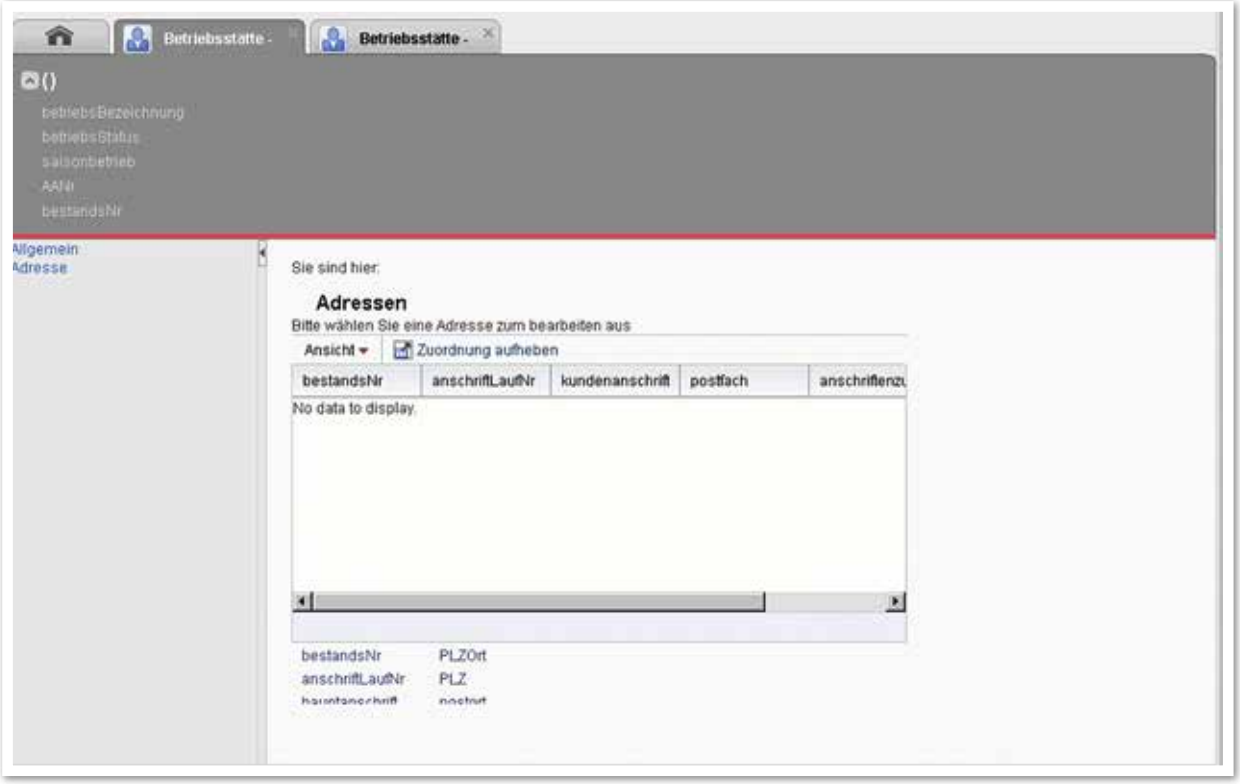

Abbildung 6: Screenshot eines Entwicklungsprojekts mit BADF in einer sehr frühen Phase

terungen, Funktionalitäten wie Ajax, Server-Push und ein mit nativen Clients vergleichbares Dialog-Framework bietet. Es wird kein zusätzlicher, manuell zu implementierender JavaScript-Code für diese Funktionalitäten benötigt. Von den in ADF enthaltenen JavaScript-Funktionalitäten wird vollständig abstrahiert.

Ein modularer Aufbau der GUI wird durch wiederverwendbare Controller mit integrierter Zustandsverwaltung und definierten Schnittstellen und den sogenannten „Contextual-Events" zur losen Kopplung einzelner Seitenregionen ermöglicht. Vorteilhaft war auch die native Integration von ADF Taskflows in das WebCenter-Portal, auf dessen Basis der Internetauftritt der BA *(siehe „http://arbeitsagentur.de")* realisiert ist. Die Zukunftsfähigkeit von Oracle ADF ist durch die JSF-Renderer gegeben, da ausgehend vom selben Sourcecode völlig unterschiedliche Codes je nach Endgerät und Browser erzeugt werden können. So wird mittlerweile – ohne Änderung des Quellcodes – bei Zugriff von einem Tablet HTML5 gerendert, wo vor wenigen Jahren noch Flash-Artefakte erzeugt wurden. Außerdem existieren spezielle Renderer zur Unterstützung von Tools für Sehbehinderte wie Lunar und Jaws.

Wichtiges Kriterium für die BA war auch die Tatsache, dass Oracle-Produkte wie der Enterprise Manager oder die Fusion Apps vom Hersteller selbst in ADF implementiert werden, da sich die BA hiervon eine kontinuierliche Weiterentwicklung und Langzeit-Support verspricht. Die zwischenzeitlich erfolgte Vorstellung von ADF Mobile bestätigt diese Annahme. Zuletzt steht mit dem JDeveloper auch eine IDE zur Verfügung, die eine WYSIWYG-Entwicklung in Verbindung mit den zentral bereitgestellten Komponenten erlaubt.

Die Referenz-Architektur

Das Framework Oracle ADF implementiert den JSF-Standard und basiert auf dem etablierten MVC-Pattern für GUIs. Die Model-Schicht innerhalb von ADF, die von den darunterliegenden Daten-Services abstrahiert, wird von Oracle auf die Zusammenarbeit mit dem proprietären, objektrelationalen Mapping-Framework ADF Business Components optimiert. Da in der BA jedoch standardmäßig POJOs als DTOs vom EJB- oder WebService-Backend zur GUI-Schicht verwendet werden, wurde das Binding entsprechend angepasst *(siehe Abbildung 4)*.

Alle Oberflächenabläufe werden als Bounded Taskflow auf Basis von JSF-Page-Fragmenten implementiert. Jeder Taskflow – mit definierten Ein- und Ausgabe-Parametern – ist dabei für die potenzielle Wiederverwendung in anderen Applikationen per ADF Library einsetzbar. Wiederverwendbare Teile einzelner Seitenfragmente werden wiederum als deklarative Komponenten implementiert.

Soweit notwendig, werden Referenzen auf UI-Komponenten im View-, Request- oder BackingBean-Scope zu ihren Backing-Beans gebunden. Der eigentliche UI-State wird je nach Anwendungsfall mittels Managed Beans im PageFlow- oder Session-Scope gehalten. Ein weiterer wichtiger Grundsatz ist, kein HTML-Markup in den JSF-Seiten zu verwenden, da ansonsten die durch den jeweiligen JSF-Renderer gegebene Plattform-Unabhängigkeit verloren gehen kann.

BADF

Das Baukasten-ADF (BADF) implementiert den BA-Design-Guide auf Basis der Refe-

renz-Architektur. Motivation für dieses Toolkit war, die Umsetzung der Design-Vorgaben zu vereinfachen und deren Implementierung zu vereinheitlichen.

Grundfunktionalitäten wie die Navigation (Reiter/Baum-Navigation) und auch der grundlegende Seitenaufbau werden durch JSF-Templates und deklarative UI-Komponenten geliefert. Das Exception-Handling wiederum wird durch sogenannte „Taskflow-Templates" zentral sichergestellt. Sämtliche Templates, deklarative UI-Komponenten und Skins sind zentral mit ADF-Libraries bereitgestellt und stehen den Entwicklern im JDeveloper sofort zur Verfügung.

Es gibt zwei Skins für interne BA-Anwendungen, einschließlich optimierter Ausgabe für Sehbehinderte (insbesondere im Hinblick auf Farbkontraste und Schriftgröße). BADF implementiert die Referenz-Architektur bezüglich Binding und Zustandsverwaltung von Java POJOs, EJBs und Web-Services, liefert Supportklassen für Contextual Events und enthält Basisklassen für immer wiederkehrende Fälle wie z.B. Exception Handling oder spezielle Konverter beziehungsweise Validatoren. Ferner existiert eine Security-Integration mit einem eigenen OPSS-Provider, basierend auf JAAS und einer Security-Bean für den komfortablen Zugriff per EL-Ausdrücken in JSF-Seiten. *Abbildung 5* zeigt einen Überblick über die wesentlichen Bestandteile des BADF-Toolkits.

Zuletzt wurde eine Werkzeugstraße für den zentralen Integrationsprozess auf Basis von Hudson, ojdeploy, Sonar und automatisierte Tests bis hin zum Deployment auf WebLogic Servern implementiert. Bei Bedarf kann eine textuelle DSL verwendet werden, um GUI-Masken zu spezifizieren. Der generierte Code nutzt die BADF-Artefakte, um Design-Guide-konforme Oberflächen zu erzeugen.

Fazit

Mittlerweile sind drei Projekte mithilfe des Design-Guides und BADF erfolgreich realisiert und in Produktion. Einige weitere Projekte sind aktuell kurz vor dem „Go-Live". Oracle ADF „als Enabler" hierfür hat in den letzten beiden Jahren erheblich an Reife gewonnen und die Benutzerakzeptanz ist gut.

Die Vereinheitlichung der Benutzeroberflächen unterschiedlicher Verfahren mit dem Design-Guide und BADF hat sich bewährt. Änderungen am Design-Guide können zentral durch Anpassungen der Page-Templates und des Skins vorgenommen werden und gelten für alle Produkte und Projekte, die BADF nutzen.

Die gewählte Referenz-Architektur unterstützt dabei sowohl prozessunterstützende Anwendungen sowie traditionelle CRUD-Anwendungen. Schon zu Beginn der Entwicklung ist das spätere Zieldesign klar erkennbar, wie *Abbildung 6* verdeutlicht.

Zusammenfassend bleibt festzuhalten, dass eine industrialisierte GUI-Entwicklung durchaus möglich ist. Voraussetzungen hierfür sind eine größtmögliche Standardisierung der Benutzeroberfläche, die Einhaltung einer strikten Referenzarchitektur und die weitgehende Automatisierung der Implementierung von wiederkehrenden Gestaltungsmustern.

Der gewünschte Standardisierungsgrad kann nur zusammen mit der die IT beauftragenden Fachabteilung gelingen, da hierdurch natürlich die Gestaltungsmöglichkeiten einer Anwendung – gewollt – eingeschränkt werden. Dies nahm die Fachabteilung der BA angesichts der Vorteile wie massiver Kostenersparnis in der Entwicklung und einer durch die Vereinheitlichung optimierten Benutzerfreundlichkeit (intuitiv, sauber, leicht erlernbar) in Kauf. Die IT erreicht außer einer signifikanten Verringerung des Entwicklungsaufwands bei der Entwicklung von GUIs auch die Vereinfachung von Test und Betrieb sowie verbesserte Supportmöglichkeiten im Fehlerfall und erheblich verringerte Aufwände bei Weiterentwicklung und Pflege im zukünftigen Applikations-Lifecycle.

Oracle ADF fungierte bei diesem Vorgehen als Enabler, die Konzepte des BA-Design-Guides waren hiermit vollständig und exakt umsetzbar. Eine Übertragbarkeit des in der BA gewählten Ansatzes hängt allerdings insbesondere von der Kundenbasis und der Anzahl und Größe der zu unterstützenden Applikationen ab und sollte keinesfalls als allgemeingültig verstanden werden.

Ralf Ernst
ralf.ernst@arbeitsagentur.de

Dipl.-Inf. Andreas Koop
andreas.koop@enpit.de

Bring your own Device soll mit neuer Oracle Mobile Security Suite sicherer werden

Oracle hat auf dem Mobile World Congress in Barcelona seine neue Oracle Mobile Security Suite vorgestellt. Damit können Organisationen dem Trend „Bring you own Device" (BYOD) nachgehen, ohne Zugeständnisse bei der Sicherheit machen zu müssen. Die Oracle Mobile Security Suite ermöglicht den sicheren Zugriff auf sensible Unternehmensanwendungen und Daten auf gängigen mobilen Endgeräten und bietet eine integrierte Plattform, von der aus Organisationen die Zugriffsverwaltung von allen Endgeräten, darunter Laptops, Desktop-Rechner und andere mobile Geräte, auf alle Anwendungen vornehmen können. Darüber hinaus stellt der Arbeitsbereich eine komplette Lösung für die mobile Anwendung bereit. Darin sind Single Sign-on, eigenes Netzwerk-Tunneling für jede Anwendung, Verschlüsselung der gespeicherten Daten, native Integration mit Microsoft Active Directory für eine gemeinsame Nutzung, ein Katalog mit Unternehmensanwendungen und ein Transformations-Tool enthalten.