Hartkodierten Code verringern S. 10

Besser entwickeln mit devonfw S. 76

Dungeons & Dragons und Metriken S. 82

JAVA Mag

Java magazin

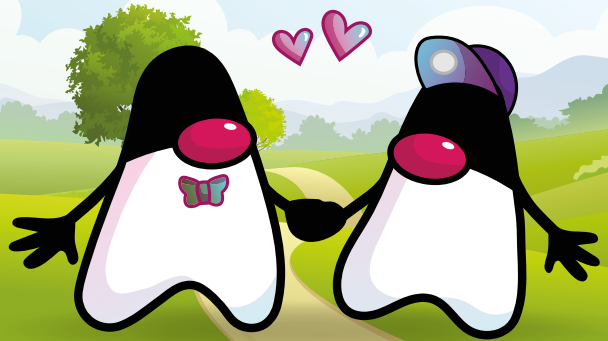
Java | Architektur | Software-Innovation

Spring

Frühlingsgefühle in der Java-Welt

Sonderdruck für Capgemini

www.capgemini.com



www.javamagazin.de

Ausgabe 9.2020

Brückenbauer in der Open-Source-Welt – Teil 1

Besser entwickeln mit devonfw

Ist die IT-Branche eigentlich richtig professionell? In Branchen wie Architektur und Bauwesen gibt es für wiederkehrende Probleme etablierte Standards und fertige Lösungen – von der Schraube zum Fertighaus. In der IT wird hingegen das Rad sehr oft neu erfunden. So gibt es beispielsweise unzählige APIs und Implementierungen für einen Logger. Der Mehrwert eines individuellen Loggers ist für den Nutzer der Lösung jedoch gleich Null.

von Jörg Hohwiller

Glücklicherweise gibt es auch in der IT gute Standards, und gerade Java ist mit der JEE bzw. Jarkarta EE hier eigentlich gut aufgestellt. Auch wenn dies beim Logging in Java SE nicht so geglückt ist, hat sich doch mit slf4j ein guter De-facto-Standard etabliert. Also ist doch alles hoch professionell – oder nicht?

Artikelserie

Teil 1: Besser entwickeln mit devonfw
Teil 2: Bessere Entwicklungsumgebung mit devonfw-ide
Teil 3: Inkrementelle Codegenerierung mit CobiGen
Teil 4: Architekturvalidierung mit sonar-devon4j-plugin

Vor vielen Jahren haben wir einmal analysiert, wie die interne Architektur und die Struktur von unterschiedlichen Businessanwendungen aussehen, die alle mit Java implementiert sind. Dabei konnten wir zwar durch Standards wie das Java Persistence API einige Gemeinsamkeiten erkennen, dennoch war das Ergebnis ernüchternd: Selbst bei Anwendungen mit ähnlicher Problemstellung vom gleichen IT-Dienstleister wurden identische Probleme anders gelöst, Package-Konventionen anders gewählt und „Komponenten“ ganz unterschiedlich im Code abgebildet – vom Vergleich unterschiedlicher Dienstleister ganz zu schweigen.

Im schlimmsten Fall gibt es diese fehlende Homogenität sogar innerhalb der gleichen Anwendung. Bei Audits von Softwareprojekten in Schieflage habe ich im Härtefall schon Systeme vorgefunden, die bereits

Manche Hypes führen zu immer kürzeren Lebenszyklen, und in der Welt der JavaScript-Bibliotheken scheint die Wegwerfgesellschaft auch in der IT angekommen zu sein.

vor dem Go-live nicht mehr zu retten waren, da Struktur und Architektur überhaupt nicht vorhanden waren. Zudem führen manche Hypes zu immer kürzeren Lebenszyklen, und in der Welt der JavaScript-Bibliotheken scheint die Wegwerfgesellschaft auch in der IT angekommen zu sein. Die IT-Branche zeigt hier zumindest großes Verbesserungspotenzial in den folgenden Punkten:

- *Qualität*: Durch klare Richtlinien und deren Überprüfung können Risiken deutlich reduziert und die Qualität kann verbessert werden – der Go-live wird entspannt.
- *Effizienz*: Durch Homogenität und Standardisierung kann der Aufwand für Entwicklung und Wartung deutlich gesenkt werden – Kunde und Dienstleister profitieren gleichermaßen.
- *Zukunftssicherheit*: Durch Verwendung etablierter und erprobter Technologien und Frameworks wird Software langlebig – die Software läuft Jahrzehnte.

Entstehung und Ziele von devonfw

Aus dieser Erkenntnis heraus haben wir 2014 die Open Application Standard Platform (OASP) gegründet, aus der wenig später devonfw [1] hervorging, das Ende 2018 die OASP vollständig abgelöst hat. Es handelt sich hierbei um ein Open-Source-Projekt, das sich weniger auf Code, sondern vor allem auf Standardisierung und Dokumentation fokussiert. Zudem betrachtet devonfw das Ganze, also die Anwendung bis hin zur Anwendungslandschaft, und baut dazu Brücken zwischen vielen bereits etablierten Open-Source-Lösungen, um daraus eine ganzheitliche Lösung zu definieren.

Während Hibernate, Jackson, Spring usw. gute Lösungen bereitstellen und auch ordentliche Dokumentationen liefern, erklären sie dem Entwickler nicht ausreichend, wie er eine funktionierende Anwendung baut. Im Gegenteil konfrontieren sie ihn sehr häufig mit Entscheidungen und Alternativen (z. B. Annotation an Private Field vs. Getter vs. Konstruktor oder gar XML-Deskriptoren), wobei jede Alternative ihre Vor- und Nachteile hat. Was davon letztlich als veralteter Irrweg gilt, ist für den Entwickler ohne tiefgreifende Erfahrung oft schwer zu sagen. Das führt zu langwierigen Entscheidungen im Design sowie inhomogenen und teilweise schlechten Ergebnissen.

Die Dokumentation in devonfw hat dagegen das Ziel, Entwickler und Architekten auf den etablierten und sicheren Weg zu führen und dazu fast vollständig auf Alternativen zur Lösung des gleichen Problems zu verzichten. So gibt sie dem Entwickler auf Augenhöhe die jeweils richtigen Antworten und erklärt anschaulich an konkreten Beispielen, wie ein Problem gelöst wird. Auf diese Weise ist die Erfahrung aus Projekten tausender Bearbeitertage zu einem konsolidierten Wissensschatz gereift. Dieser wird mit einer großen Community kontinuierlich weiterentwickelt und angepasst, denn keine Branche ist stärker von ständigem Wandel geprägt als die IT. Zu diesem Zweck wurde in devonfw sehr viel Energie in die Prozesse und Werkzeuge zur Pflege der Dokumentation investiert. Neben GitHub und AsciiDoc haben wir viele Details durch jahrelange Arbeit aufgebaut, sodass jeder mit einem GitHub-Account ganz einfach Verbesserungen an der Dokumentation als Pull Request beitragen kann. Das ist sogar direkt im Browser mit wenigen Klicks möglich und damit ähnlich einfach wie bei Wikipedia. Die Dokumentation einzelner Releases wird gleichzeitig als PDF aufbewahrt und erlaubt es auch Projekten, die in der Wartung nicht auf die neueste Version aktualisieren können oder wollen, noch den passenden Stand der Dokumentation vorzufinden. Inzwischen ist sogar ein YouTube Channel für Videotutorials hinzugekommen, um den Einstieg noch weiter zu vereinfachen.

Gegründet und betrieben wird devonfw übrigens von Capgemini. Es beteiligen sich inzwischen jedoch immer mehr externe Kontributoren und weitere Firmen. Hier sind insbesondere SAP für die umfangreiche Unterstützung der SAP-Hana-Datenbank bis hin zur Demonstration in der devonfw-Demoanwendung My Thai Star (MTS) mit Predictive Analysis und Geocoding sowie Red Hat für OpenShift-Support und Zertifizierung zu nennen. Umgekehrt bringt die devonfw-Community auch einige Issues, Bugfixes und Pull Requests in die verwendeten Open-Source-Produkte wie Spring, Maven usw. ein. Bei devonfw kann jeder mitmachen , der Ahnung von Software-Engineering und -Entwicklung hat – also auch du.

devonfw im Überblick

Die Entwicklung von Anwendungen geht heutzutage über Programmiersprachen hinaus. Hierzu ist devonfw in Stacks unterteilt:

- devon4j bietet alles zum Bau einer Geschäftsanwendung (Backend) mit Java.
- devon4net liefert diese Antworten für C#/.NET.
- devon4ng hilft Clients, mit Angular zu bauen.

Ein immer wieder diskutierter Punkt bezüglich devonfw ist die Kompatibilität des Ansatzes mit neuesten Architekturmustern wie Microservices, SOA oder sogar FaaS.

- devon4node erklärt, wie man Backends mit NestJS baut und mit Node.js betreibt.
- devon4x zeigt, wie mobile Frontends mit Xamarin gebaut werden.

Um alles live in Action sehen zu können, gibt es Beispiele bis hin zur Demoanwendung MTS, die ein digitalisiertes Restaurant mit Web- und Mobile-Frontend über alle Stacks demonstriert. Wichtig für eine standardisierte und effiziente Entwicklung sowie für einen schnellen Projektstart ist auch die Nutzung der richtigen Tools:

- devonfw-ide bietet eine Lösung zur vollautomatischen Einrichtung und Konfiguration der Entwicklungsumgebung in einer Sandbox, ohne den Overhead einer VM.
- CobiGen ist ein inkrementeller Codegenerator, der über Templates frei konfigurierbar ist. Fertige Templates für devon4j, devon4net und devon4ng erlauben es, beispielsweise CRUD-Funktionalität vollständig aus dem Datenmodell über alle Schichten hinweg zu generieren.
- sonar-devon4j-plugin kann über den SonarQube Marketplace installiert werden und die Architektur einer devon4j-Anwendung automatisch validieren sowie Verletzungen der devonfw-Architektur aufdecken.
- solicitor scannt alle direkten und transitiven Dependencies der eigenen Anwendung samt ihren Lizenzen und deckt vermutliche Lizenzinkompatibilitäten auf bzw. zeigt, wo Legal-Compliance-Klärungen notwendig sind. Im Gegensatz zum – mit GitHub vollintegrierten – Service Fossa [2] erlaubt solicitor eine firmenspezifische Konfiguration zur automatisierten Lizenzprüfung und Konfliktlösung sowie die Erstellung von Lizenzreports.
- MrChecker [3] ist eine JUnit-integrierte Lösung zur Implementierung von End-to-End-Tests (E2E-Tests). Aktuell erlauben sogenannte „Treiber“ für Selenium [4] und Appium [5] das Testen von browserbasierten und mobilen Applikationen über JUnit-Tests.

Neugierig geworden? Dann ist [1] der beste Einstieg, um tiefer in alle Stacks und Werkzeuge von devonfw einzutauchen. Alle Inhalte werden ausschließlich auf GitHub verwaltet. Um sich dort zurechtzufinden, ist es wichtig zu verstehen, dass devonfw auf GitHub in folgende Organisationen unterteilt ist:

- devonfw [6] ist die offizielle Organisation mit allem, was reif für den produktiven Einsatz ist und auch entsprechende Wartung und Weiterentwicklung bietet. Die genannten Stacks und Werkzeuge sind alle hier zu finden.
- devonfw-forge [7] ist der Incubator, in dem neue Projekte entstehen. Hier finden sich experimentelle Lösungen, die später in die offizielle Organisation überführt werden oder ggf. auch wieder verschwinden.
- devonfw-sample [8] bietet Showcases, Demos und Beispiele als lauffähige Projekte, um spezielle Features aus dem offiziellen devonfw zeigen.
- devonfw-training [9] ist die neue Organisation für Trainings zu devonfw.

Architektur und Struktur

Für die technische Architektur einer Anwendung liefert devonfw eine fertige Blaupause (**Abb. 1**). Diese konsolidiert gängige Best Practices der Three-Tier-Applikation (Client, Server, DB) sowie der Strukturierung in technische Schichten (Layer) und fachliche Komponenten (Slices).

Die grundsätzliche Struktur sollte keinen Architekten überraschen: Im Data Access Layer wird der Datenbankzugriff gekapselt (via Spring Data, bei klassischem RDBMS via JPA), der Logic Layer konzentriert sich auf die fachliche Aufgabe der Anwendung und damit die eigentliche Problemstellung des Kunden. Diese Geschäftslogik wird im Service Layer als Schnittstellen (empfohlen als REST mit JSON) exportiert. Die Anwendungsarchitektur ist zustandslos und skaliert damit optimal. Ebenfalls ermöglicht die Architektur einen Multi-Channel-Ansatz, bei dem GUI-Clients für verschiedene Kanäle bzw. Endgeräte (Desktop, Web, Mobile) mit unterschiedlichen Technologien angebunden werden können.

Das Besondere an devonfw ist, dass diese Architektur genau bis auf den Code gemappt wird. Am Beispiel von Java definiert devon4j ein Schema, wie Layer und Komponenten in Java-Package-Namen abgebildet werden und gibt Namenskonventionen für typische Konstrukte wie Entitäten, Use Cases, Services, Transportobjekte etc. an. Dadurch können Werkzeuge wie CobiGen oder das sonar-devon4j-plugin den Code „verstehen“ und damit große Mehrwerte und Effizienzgewinne bringen. Hierzu wird es Folgeartikel geben, die das genauer erklären. Als Technologiestack für Java empfiehlt devonfw im Wesentlichen:

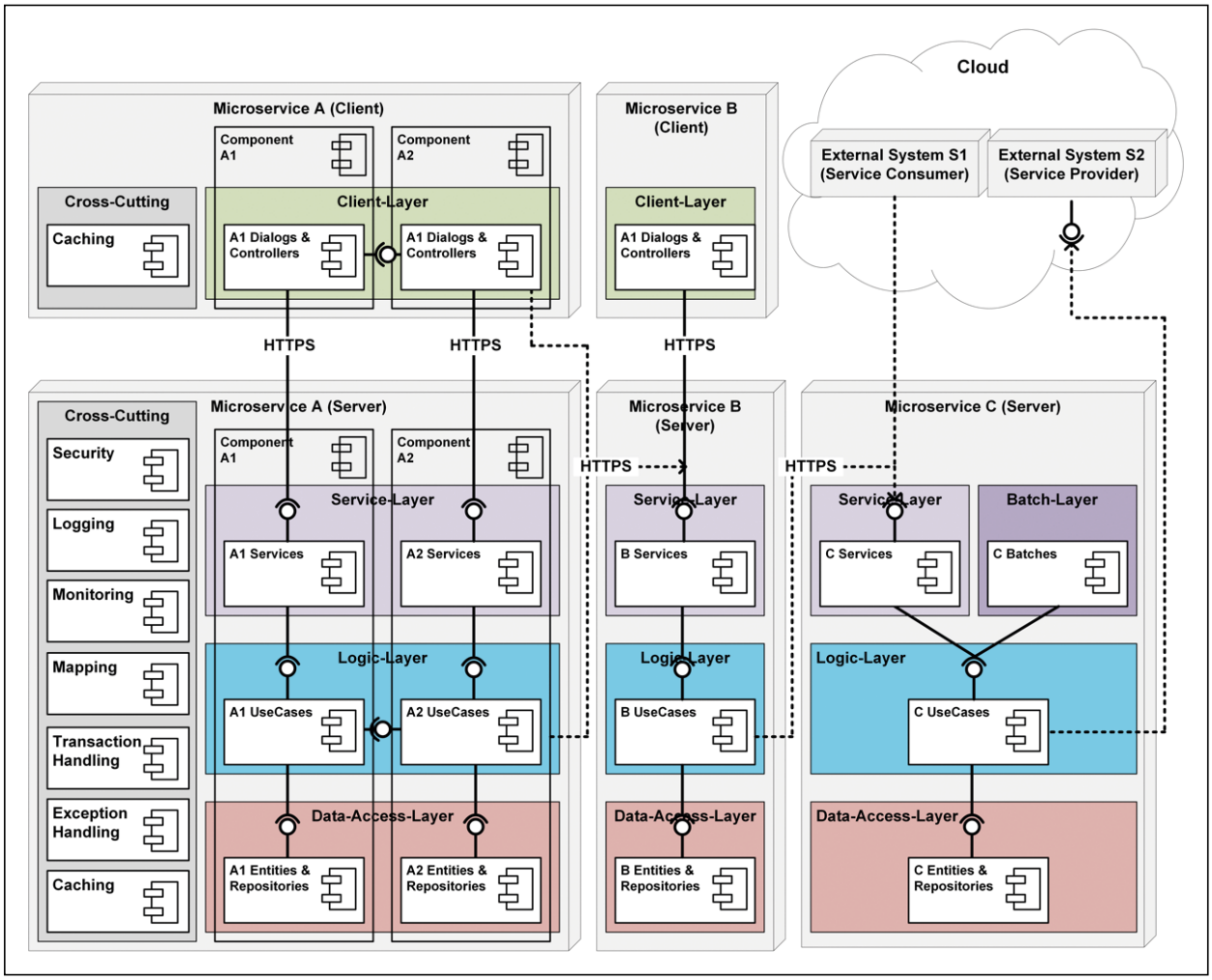


Abb. 1: Blaupause von devonfw

- Backends mit Java (devon4j) auf Basis von Spring und Spring Boot, Jackson, Hibernate, QueryDSL, Apache CXF, JUnit, AssertJ u. v. m.
- Frontends mit Angular (devon4ng) auf Basis von Angular, TypeScript, Jasmine, Protractor und ggf. NgRx oder auch Ionic

Wer diese Technologien bereits kennt und nutzt, kann direkt durchstarten. Andernfalls bietet die ausführliche und verständliche Dokumentation zu jedem Thema eine gute Erklärung mit praktischen Codebeispielen, um den Entwickler auf Augenhöhe abzuholen.

Gerüstet für die Zukunft?

Ein immer wieder diskutierter Punkt bezüglich devonfw ist die Kompatibilität des Ansatzes mit neuesten Architekturmustern wie Microservices, serviceorientierten Architekturen (SOA) oder sogar Serverless (FaaS). Die Diskussion ist berechtigt, denn der erste Anschein, wie auch hier vorgestellt, suggeriert die etablierte Umsetzung einer Architektur für Monolithen oder Modulithen. Während einige Architekturmuster wie die Versionierung von Schnittstellen, Circuit Breaker, Two-Phase Commit, ggf. sogar reaktives Programmieren und viele mehr auch in monolithischen Architekturen immer wieder ihre Anwendung fanden, verschiebt sich die Wichtigkeit einzelner Architekturmuster mit der Einführung beispielsweise von Microservices.

Dennoch bleibt das Grundwerkzeug eines Softwarearchitekten auf Mikroebene das gleiche. Die Frage zur Kompatibilität von devonfw mit Microservices bezieht sich also vielmehr darauf, weitere Architekturmuster zu definieren und einzuordnen, und auf die Betrachtung der Architektur der Infrastruktur von Softwaresystemen. Denn gerade für Microservices verschiebt sich ein Fokus der Systemarchitektur auf die Architektur der Infrastruktur zur Orchestrierung und zum Betrieb von vielen unabhängigen eigenständigen Kleinstapplikationen.

Insbesondere für die Architektur der Infrastruktur definieren Frameworks wie Istio [10] bereits eine klare Richtung für die Open-Source-Welt, die wir derzeit in der devonfw-Community evaluieren. Denn am wichtigsten für die Auswahl von Frameworks, Bibliotheken und Werkzeugen für devonfw sind und bleiben für uns die Erfahrungen, die in weltweiten Projekten gesammelt werden konnten. Hierzu zählen insbesondere der Bedarf und die Umsetzbarkeit von neuen Technologien für un-

Insbesondere für die Architektur der Infrastruktur definieren Frameworks wie Istio bereits eine klare Richtung für die Open-Source-Welt, die wir in der devonfw-Community evaluieren.

sere Kunden und ihre möglichst langanhaltende Tragfähigkeit im Produktiveinsatz. Breites Feedback und Diskussionen sind hier sehr willkommen.

Fazit und Ausblick

Jeder Java-Entwickler kennt vermutlich die Story, wie Maven durch Standardisierung aus Ant-Chaos homogene Projektstrukturen und Build-Schritte etabliert hat. Genau das gleiche macht devonfw für das, was unterhalb von Ordnern wie *src/main/java* liegt. Seit wir mit devonfw professionell Projekte durchführen, finden sich die Entwickler sofort in Projekten zurecht und lokalisieren bei jeder Story sofort die Stelle im Code, wo die notwendige Änderung hingehört. Die Architektur hat sich in den unterschiedlichsten Kundenkontexten als tragfähig und skalierbar erwiesen. Von Horror-Stories (wie in den eingangs erwähnten Studien und Audits) bleibt man mit devonfw verschont. Bereits am ersten Tag des Projekts stehen die Referenzarchitektur, Technologie, Entwicklungsumgebung, Dokumentation u. v. m. schon fast vollständig zur Verfügung, während andere Projekte ohne devonfw mit der Definition des technischen Projektrahmens Monate verbringen können.

Lust auf mehr?

Dieser Artikel hat nur einen ersten groben Überblick gegeben. Weitere detaillierte Artikel zu jeweils einzelnen Tools und Stacks werden folgen:

- bessere Entwicklungsumgebung mit devonfw-ide
- beschleunigte Entwicklung durch inkrementelle Codegenerierung mit CobiGen
- bessere Qualität durch Architekturvalidierung mit sonar-devon4j-plugin
- keine Risiken durch Lizenzverstöße mit solicitor
- End-to-End-Testing mit MrChecker

Gebt uns Feedback, wenn ihr die Themen bessere Softwareentwicklung und devonfw spannend findet und noch mehr erfahren möchtet.

Jörg Hohwiller ist seit 2002 für Capgemini als Architekt und Berater tätig. Privat entwickelt er aktiv an vielen Open-Source-Projekten mit. Sein Ziel ist es, gemeinsam mit vielen anderen erfahrenen Entwicklern die IT kontinuierlich besser zu machen.

Links & Literatur

[1] https://devonfw.com

[2] https://fossa.com

[3] https://devonfw.com/website/pages/docs/master-mrchecker.asciidoc.html

[4] https://www.selenium.dev

[5] http://appium.io

[6] https://github.com/devonfw

[7] https://github.com/devonfw-forge

[8] https://github.com/devonfw-sample

[9] https://github.com/devonfw-training

[10] https://istio.io/docs/concepts/what-is-istio/