02-2016 | Sommer | www.ijug.eu

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

Aus der Community – für die Community

Java aktuell

D: 4,90 EUR A: 5,60 EUR CH: 9,80 CHF Benelux: 5,80 EUR ISSN 2191-6977

Java ist Programmiersprache des Jahres

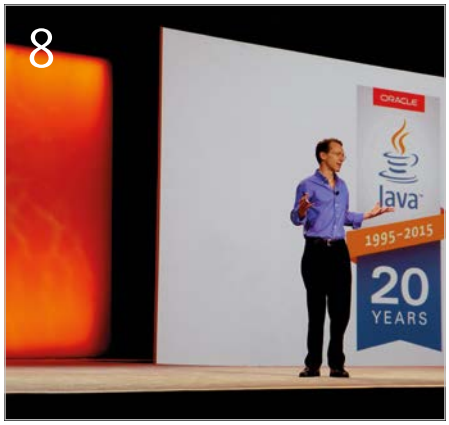
Sicherheit
- Social Login
- Single Sign-on

Richtig testen
Statische Code-Analyse

Richtig entwickeln
Lösung mit JavaFX und JVx

4 191978 304903 02

iJUG Verbund



Neues von der JavaOne

Die gängigsten Open-Source-Tools zur Code-Analyse im Praxis-Einsatz

Reaktive Anwendungen mit asynchronen, Event-getriebenen Architekturen gewinnen stark an Bedeutung

Groovy und Grails – quo vadis?

Falk Sippach, OIO Orientation in Objects GmbH

Das Jahr 2015 begann turbulent für die beiden bekanntesten Projekte des Groovy-Universums. Die daraus entstandenen Unsicherheiten haben die neuen Funktionen der Releases 2.4 (Groovy) beziehungsweise 3.0 (Grails) ziemlich in den Hintergrund gedrängt. Dabei sind die Projekte lebendiger denn je und vor allem schon längst reif für den produktiven Einsatz. Der Artikel analysiert den aktuellen Stand und wagt einen Ausblick in die Zukunft. In der nächsten Ausgabe folgt in einem zweiten Teil ein Blick auf die wichtigsten und interessantesten Neuerungen der vergangenen Releases.

Der bisherige Herausgeber Pivotal hatte im Januar 2015 angekündigt, sowohl Groovy als auch Grails nicht mehr zu unterstützen. Viele haben sich damals gefragt, wie und ob es überhaupt weitergeht. Dem Team rund um Groovy wurden durch die Schließung der Codehaus-Plattform zudem weitere Steine in den Weg gelegt. Dank einer starken Community und einer gewissen Reife haben sich beide Projekte durch diese Probleme aber nicht entmutigen lassen und konnten sogar gestärkt aus der Situation hervorgehen.

Ruhig geworden

In den vergangenen Jahren war es bereits deutlich ruhiger um Groovy und Grails geworden. Die Gründe dafür sind vielfältig. Einerseits sind beide Projekte mittlerweile mehr als zehn Jahre alt. Die Welt ist jedoch nicht stehen geblieben und andere interessante Sprachen und Frameworks haben die Aufmerksamkeit in Fachmagazinen oder auf Konferenzen auf sich gezogen. Hinzu kam eine scheinbar fehlende Mainstream-Tauglichkeit im Falle von Groovy (Stichwort: „dynamische Typisierung"), wodurch ein Großteil der Java-Entwickler unnötigerweise abgeschreckt wurde. Die vielen Vorteile von Groovy und Grails gingen da zuletzt leider unter.

Am Beispiel der Gardner-Hype-Kurve *[1]* kann man erkennen, dass es vielen Technologien so ergeht. Nach dem Gipfel der Erwartungen in den Anfangsjahren folgt meist das Tal der Ernüchterung, wenn die breite Masse merkt, dass man nicht alle Vorteile haben kann, ohne auch die einen oder anderen Konsequenzen in Kauf zu nehmen. Typischerweise erholt sich die Kurve schließlich, wenn die ernsthaft Interessierten den Pfad der Erleuchtung beschreiten und sich die Technologie auf einem gewissen Niveau (der Produktivität) etabliert. Bei Groovy ist es schwer, diesen Verlauf mit Jahreszahlen zu versehen. Aber darauf kommt es auch nicht wirklich an. Die Tendenz lässt sich auf jeden Fall nachvollziehen.

Aktuell bekommen wir mit dem Thema „Microservices" wieder ein schönes Beispiel für einen solchen Verlauf geliefert. Obwohl der Ansatz bereits einige Jahre alt war, begann mit dem Artikel von Martin Fowler und James Lewis Anfang 2014 ein regelrechter Hype (technologischer Auslöser). Aber bereits ein Jahr später machte sich Ernüchterung breit. Auch bei Microservices gibt es keinen „Free Lunch" und die Umsetzung passt nicht zu jeder Unternehmensform („Conway's Law"). Zudem sind die Nachteile nicht trivial und müssen irgendwie behandelt werden, insbesondere die verteilte Natur der Services und die daraus resultierenden Folgen. Trotzdem werden sich Microservices für ausgewählte Szenarien als adäquater Architektur-Stil durchsetzen. Überprüfen können wir diese These allerdings frühestens in ein paar Monaten, wenn nicht sogar Jahren. Übrigens, wer jetzt schon auf den Microservices-Zug aufspringen möchte, findet mit Grails ein bereits vollkommenes Framework für die Umsetzung vor.

Dass es um Groovy ruhiger geworden ist, kann man auch gut an ReGina sehen. Es handelt sich hier nicht um die Schwiegermutter eines Groovy-Core-Entwicklers. Vielmehr kam im Januar 2007 (kurz nach Groovy 1.0) ein Buch heraus, das bis heute die Bibel der Groovy-Enthusiasten ist: „Groovy in Action". Im Zeitalter der maximal 140 Zeichen bei Twitter musste eine Abkürzung gefunden werden, was zu „GinA" führte. Zwei Jahre später überlegten sich die Autoren, die Neuerungen aus zwei Minor-Releases in eine zweite Auflage einfließen zu lassen. Anfangs war man optimistisch, aber erst sechs Jahre und viele Twitter-Nachrichten später (Suche nach „#regina", „#groovy" beziehungsweise „#groovylang") konnte man im Juli 2015 diese zweite Auflage (Wortspiel: „ReGinA") in den Händen halten.

Oberflächlich betrachtet, schienen die Autoren nicht mehr genug Enthusiasmus an den Tag zu legen. Aber weit gefehlt, vielmehr war Groovy als moderne, sich stetig weiterentwickelnde Sprache der eigentliche Grund für die lange Verzögerung. Durch die hohe Häufigkeit der Releases ergab sich ständig neuer Input, den man auch noch im Buch verarbeiten wollte. Man kämpfte also tapfer dagegen an, dass das Buch bei der Veröffentlichung nicht schon wieder veraltet war. Das Pivotal-Debakel Anfang 2015 kam den Autoren um Dierk König somit gerade recht, ergab sich dadurch doch endlich mal eine Zwangspause. Dank des Early-Access-Programms des Verlags konnte man zudem bereits lange eine vorläufige Fassung kaufen und bekam außerdem die Aktualisierungen regelmäßig frei Haus geliefert.

Historie

Ein Blick in die Geschichte zeigt, dass beide Projekte immer eng verbunden waren. In den Jahren 2003 respektive 2005 erblickten sie das Licht der Welt. Während sich bei Groovy die beiden maßgeblichen Initiatoren James Strachan und Bob McWhirter relativ schnell wieder zurückzogen, sind bei Grails die Gründer bis heute aktiv. Und auch bei Groovy fanden sich rasch neue Mitstreiter, die auch heute noch die Geschicke des Projekts mitbestimmen.

Es wurde früh versucht, die Open-Source-Entwicklung über eine Firma (G2One) abzusichern, die sich wiederum mit Schulungen und Beratungsdienstleistungen finanzieren sollte. SpringSource, die Firma hinter dem Spring-Framework, übernahm aber schon kurz darauf G2One und damit Groovy und Grails in sein Open-Source-Portfolio. Wenig später wurde SpringSource dann Teil der Firma VMware, die vor vier Jahren seine Open-Source-Entwicklungsabteilung in die Firma Pivotal ausgründete. Groovy und Grails haben also schon eine bewegte Geschichte hinter sich. Trotz aller Probleme und Widerstände konnten über all die Jahre immerhin mehrere Entwickler in Vollzeit an den Projekten arbeiten. Jochen Theodorou hat die Geschehnisse aus seiner Sicht in einem nicht unkritischen Blog-Beitrag zusammengefasst [2].

Cedric Champeau hat in seinem Blog-Post „Who is Groovy?" [3] die Geschichte aus einem anderen Blickwinkel betrachtet und anhand der Commits im Sourcecode-Repository nachvollzogen. Interessanterweise ist der fleißigste Entwickler, Paul King, gar nicht von Pivotal bezahlt worden – wiederum ein Zeichen für eine starke Community.

Beide Projekte entwickelten sich ständig weiter und insbesondere Grails wurde für Groovy der Innovationstreiber. Auch nach den Versionen 1.0 Ende 2006 beziehungsweise 2008 kam es jährlich mindestens zu Minor-Releases und damit zu stetigen Verbesserungen und natürlich auch jeder Menge neuer Funktionen.

Jähes Ende

Anfang des Jahres 2015 kam nun der gravierende Einschnitt. Das Management von Pivotal suchte Kosten-Einsparpotenzial, man wollte sich auf die Kern-Kompetenzen (Cloud, Virtualisierung) konzentrieren. Dabei gerieten Groovy und Grails auf die Streichliste, möglicherweise als Alibi aus Ermangelung an anderen Alternativen. Vermutlich dürfte sich die Einsparung dieser sechs Gehälter im Gesamtbudget kaum bemerkbar gemacht haben. Stattdessen hat man nun den Einfluss über Groovy und Grails verloren, obwohl beide Projekte mittlerweile stark in der Spring(-Boot)-Welt verankert sind.

Groovy 2.4 und Grails 3.0 sind die letzten Major-Releases unter dem Sponsorship von Pivotal: „The decision … is part of Pivotal's larger strategy to concentrate resources on … its growing traction in Platform-as-a-Service, Data, and Agile development … Pivotal has determined that the time is right to let further development … be led by other interested parties … who can best serve the goals …" [4]

Im Nachhinein betrachtet war der Zeitpunkt aber gar nicht schlecht gewählt. Immerhin waren beide Projekte mittlerweile sehr ausgereift und haben seit Langem einen festen Platz in der Open-Source-Welt eingenommen. Pivotal hatte außerdem eine Gnadenfrist von etwa drei Monaten zugestanden. Die bereits angekündigten Releases konnten so noch fertiggestellt werden. Für die bereits gekündigten Entwickler trotzdem keine leichte Zeit, da man nebenher nach einem anderen Sponsor oder einer neuen Arbeitsstelle suchen musste.

Wechsel zu Apache

Im Fall von Groovy verschärfte sich die Situation letztlich noch. Der bisherige Infrastruktur-Anbieter Codehaus (Mailinglisten, Jira, Homepage etc.) musste sich der erdrückenden Marktmacht von GitHub beugen und schaltete seine Systeme im Frühjahr 2015 ab. Auch wenn sich das schon längere Zeit angekündigt und man bereits Vorkehrungen getroffen hatte, diese Ereignisse fielen einfach unglücklich zusammen. Als Alternative wurde sehr ausführlich über den Anschluss an eine Open Source Foundation debattiert. Apache, Eclipse und Software Freedom Conservancy standen zur Auswahl [5]. Nach Abwägen der Vor- und Nachteile entschied man sich für die Apache Software Foundation. Mit der Aufnahme in den dortigen Inkubator mussten allerdings bestimmte Anforderungen und Konventionen erfüllt werden. Das bedeutete neben dem Umzug der Infrastruktur (Issue Tracker, Code-Repo, Mailinglisten) auch Umstrukturierungen und Anpassungen an den Prozessen und am Projekt selbst.

Im Sommer konnten mit 2.4.4 und 2.4.5 zwar schon erste kleinere Bugfix-Releases unter der neuen Schirmherrschaft herausgebracht werden, wirklich neue Funktionalität gab es aber seit 2.4 verständlicherweise nicht mehr. Immerhin ist mittlerweile wieder ein Großteil der ursprünglichen Committer an Bord. Auch da gibt es bei Apache gewisse Prozesse und Formalitäten, die es einzuhalten galt. Ende 2015 standen die Aussichten bereits gut, den Inkubator bald verlassen zu können. Als vollwertiges Apache-Projekt wird Groovy vermutlich nochmal mehr Aufmerksamkeit bekommen. Die steigenden Download-Zahlen im Jahr 2015 deuten schon darauf hin. Der Wechsel zu Apache hat dem Projekt also definitiv gutgetan.

Grails zu OCI

Das Grails-Team, das bis kurz vor dem Ende der Zeit bei Pivotal hart an dem großen 3.0er-Release-Sprung arbeitete, gönnte sich zunächst einige Wochen der Ruhe. Aber bereits im Mai konnte mit OCI Inc. ein neuer Hauptsponsor präsentiert werden, der direkt zwei der Haupt-Entwickler auf seine Lohnliste gesetzt hatte. Zudem verstärkte OCI seine Bemühungen im zweiten Halbjahr und stellte weitere Grails-Committer an. Man ist also sehr stark an einer Weiterentwicklung interessiert und versucht, hauptsächlich über Beratungsdienstleistungen und Projektgeschäft mit Grails die Ausgaben zu finanzieren.

Auch wenn sich für die Projekte letztlich alles zum Guten gewendet hat, so haben die Umstrukturierungen auch menschliche Schicksale betroffen. Natürlich sind mittlerweile alle untergekommen. Aber ein Teil musste sich umorientieren. Von den ursprünglich sechs Pivotal-Angestellten können nur Graeme Rocher und Jeff Scott Brown weiter Vollzeit an ihrem Projekt (Grails) entwickeln. Cedric Champeau (Groovy) und Lari Hotari (Grails) sind zu Gradle gewechselt und bleiben so immerhin dem Groovy-Universum treu. Ihre Zeit für Groovy beziehungsweise Grails wird sich jedoch aufgrund der großen Aufgaben bei Gradle in Grenzen halten. Der ehemalige Groovy-Projektleiter Guillaume Laforge wird auch weiterhin als Botschafter und nun Apache PMC Chair zur Verfügung stehen, er hat sich beruflich aber mittlerweile der Firma Restlet zugewandt. Die längste Aus- und Bedenkzeit hatte sich Jochen Theodorou genommen. Seit Kurzem arbeitet er nun bei der Schweizer Firma Canoo, die sich in den vergangenen Jahren im Umfeld von Groovy und Grails bereits einen Namen gemacht hat.

Im zweiten Teil dieses Artikels werden wir einen genaueren Blick auf die Eigenschaften und Neuerungen der letzten Groovy- und Grails-Versionen werfen. Für die Eiligen und die Entscheider wollen wir an dieser Stelle aber schon mal ein Fazit ziehen und den Ausblick in die Zukunft wagen.

Ausblick

Nach den politischen Querelen waren viele zunächst verunsichert. Aber beide Projekte sind definitiv noch nicht reif für den Framework-Friedhof. Vielmehr sind sie gestärkt aus den Geschehnissen hervorgegangen, dank der bereits stabilen Codebasis und der großen Community. Dazu passt auch die Aussage von Guillaume Laforge: „Here to stay. Independence. Community above all."

Dem Einsatz von Groovy und Grails steht also auch in Zukunft nichts im Wege. Aber wie bei allem ist es sinnvoll, die Stärken und Schwächen abschätzen zu können und entsprechend für die Einsatz-Szenarien abzuwägen. Man hat immer eine Wahl und sollte ganz bewusst entscheiden. Groovy eignet sich beispielsweise für Skripte, das Schreiben von Tests, die Verwendung als DSLs, Zugriffe über eine Admin-Konsole in Enterprise-Systemen und natürlich auch im Zusammenhang mit Grails, stellvertretend für viele andere Bibliotheken/Frameworks aus dem Groovy-Ökosystem.

Grails wurde immer als reines Web-Framework abgestempelt. Allerdings sollte man es nicht unterschätzen, es deckt schließlich den ganzen Bereich von der Persistenz bis zum Frontend ab und bietet durch seine Modularität viel Flexibilität. Man hat immer die Möglichkeit, auf bestehende Plug-ins zurückzugreifen oder eigene zu entwickeln. Grails eignet sich von jeher auch gut für das Prototyping, weil man in kurzer Zeit sehr viel erreicht. Firmen sollen schon Ausschreibungen gewonnen haben, weil sich der Kunde die Umsetzung aufgrund des Prototyps bereits sehr gut vorstellen konnte.

Als klassische Web-Anwendung empfehlen sich aus eigenen, etwas leidvollen Erfahrungen aber eher kleine Intranet-Anwendungen. Es gibt jedoch auch diverse Erfolgsgeschichten zu Umsetzungen von größeren Projekten mit Grails. Eine wichtige Voraussetzung dafür ist, dass man ein Team zusammenhat, das die Stärken und Schwächen von Groovy und Grails kennt und damit umgehen kann. Alle müssen am gleichen Strang ziehen, was bei einem Mischmasch aus Groovy-affinen und klassischen Java-Entwicklern meist nicht der Fall ist.

Mit Grails kann man beispielsweise genauso gut auch ein schlankes REST-Backend implementieren, auf dem dann SPA/RIA-JavaScript-Clients aufsetzen. Und da Grails seit Version 3.0 auf Spring Boot aufsetzt, eignet es sich auch wunderbar für die Entwicklung der verschiedenen Ausprägungen von Microservices.

Roadmaps

Ein Blick auf die aktuellen Roadmaps zeigt, dass sich Groovy und Grails erstmals in ihrer langen Geschichte auseinanderbewegen. Während man bei Grails mittlerweile wieder sehr positiv und mit vielen Ideen in die Zukunft blickt, versucht man sich beim Groovy-Projekt nach der Aufnahme bei Apache zunächst mal zu sammeln. Hier macht sich bemerkbar, dass das Grails-Team zum größten Teil zusammenbleiben konnte und weiterhin von einer Firma finanziert wird, die aus wirtschaftlichen Gründen auch an einer raschen Weiterentwicklung interessiert ist. Am Groovy-Projekt hingegen arbeitet derzeit niemand Vollzeit, die drei bisherigen Core-Entwickler sind bei unterschiedlichen neuen Arbeitgebern untergekommen.

Zum Glück ist der Bedarf bei Groovy aber auch nicht ganz so hoch. In den letzten Jahren ist bereits eine mächtige JVM-Sprache entstanden und es fehlen keine wichtigen Show Stopper. Die noch im Jahr 2014 ausgegebenen größeren Ziele, die Neukonzeption des Meta Object Protocol (MOP) basierend auf Invoke Dynamic und die Re-Implementierung der Sprachgrammatik in Antlr v4, wurden nach dem Abschied von Pivotal zunächst auf Eis gelegt. Ganz typisch Open Source wird natürlich trotzdem weiterentwickelt. Auf der Developer-Mailingliste herrscht reger Betrieb und für die kommende Version 2.4.6 wurden auch schon wieder etwa siebzig Tickets abgeschlossen.

Während man bei Groovy also zunächst kleinere Brötchen bäckt, gibt es bei Grails aufbauend auf dem letzten großen Release 3.0 noch einiges zu tun. Schon im Januar 2016 folgte Version 3.1 mit ausgebauten Profile-Support und zum Beispiel Profilen für REST und AngularJS. Außerdem wurde die Persistenz-Schicht GORM komplett überarbeitet und es werden jetzt neue Versionen von Hibernate, MongoDB, Neo4j und Cassandra unterstützt. Bereits im zweiten Quartal soll dann schon Version 3.2 mit weiteren Verbesserungen am REST bzw. AngularJS Support und am Grails Object Relational Mapping (GORM) folgen, unter anderem für nicht-blockierende Aufrufe. Anschließend will man wieder in den jährlichen Release-Prozess wechseln. Grails 3.3 ist für 2017 angekündigt, unter anderem mit Non Blocking IO, basierend auf Netty. Dass sich OCI viel vorgenommen hat, sieht man auch daran, dass weiterhin bezahlte Mitstreiter für das Grails-Projekt gesucht werden.

Fazit

Alles in allem scheinen die erzwungenen Änderungen beiden Projekten gutgetan zu haben, wenn auch auf unterschiedliche Art und Weise. Bei Grails ist es anscheinend optimal verlaufen – es gibt weiterhin eine Firma, die vor allem ein wirtschaftliches Interesse am Fortbestehen des Frameworks hat. Hoffen wir, dass sich OCI nicht verhebt und Grails lange begleiten wird.

Für Groovy sah es zunächst düsterer aus. Aber gerade der Wechsel zur Apache Foundation wird dem Projekt in Zukunft viele Türen öffnen und neue Nutzer bescheren. War man bisher zu stark von der Politik einer einzelnen Open-Source-Firma abhängig, hat man nun eine starke unabhängige Gesellschaft mit sehr gutem Ruf im Rücken. Die Popularität scheint damit jetzt schon gewachsen zu sein, wenn man Statistiken wie dem Tiobe-Index (von Platz 80+ mittlerweile in die Top 20 *[6]*) oder den offiziellen Downloadzahlen (2015 mehr als verdoppelt im Vergleich zum Vorjahr) Glauben schenken mag.

Positiv ist natürlich, dass neben Grails noch einige weitere, vor allem schon etablierte Bibliotheken und Frameworks im Groovy-Umfeld existieren (wie Spock, Gradle und Ratpack). Hinzu kommt, dass auch viele Java-Bibliotheken auf Groovy als DSL oder Skriptsprache setzen. Groovys besondere Features (etwa Closures oder Builder) und die ausdrucksstarke Syntax scheinen zudem Treiber für andere etablierte (Java 8) oder neue Sprachen (Swift) gewesen zu sein.

Obwohl das Interesse bei Konferenzen und in Magazinen im Vergleich zur Anfangszeit natürlich spürbar zurückgegangen ist, existieren noch einige kleinere Veranstaltungen (GR8Conf, Greach, Spring 2GX), die sich nur dem Groovy-Ökosystem widmen. Und bei dem einzigen, aber hochgelobten Groovy-Vortrag auf der letztjährigen WJAX in München gab es über Twitter Anfragen/

Empfehlungen an die Organisatoren, das Thema statt im Keller mal wieder im Ballsaal anzubieten.

Die Webseite listet zudem einige namhafte Firmen auf, die Groovy verwenden. Natürlich weiß man nicht, in welchem Umfang. Aber genau das ist ja eine der Stärken: Groovys Universalität von Skripten bis hin zu ausgewachsenen Enterprise-Anwendungen. Von daher eine klare Empfehlung, Groovy und gegebenenfalls auch Grails mal (wieder) eine Chance zu geben.

Referenzen

[1] Gardners Hype-Kurve: *https://de.wikipedia.org/wiki/Hype-Zyklus*

[2] About being a paid OSS Developer for Groovy: *http://blackdragsview.blogspot.de/2015/04/about-being-paid-oss-developer-for.html*

[3] Who is Groovy: *http://melix.github.io/blog/2015/02/who-is-groovy.html*

[4] Groovy 2.4 And Grails 3.0 To Be Last Major Releases Under Pivotal Sponsorship: *http://blog.pivotal.io/pivotal/news-2/groovy-2-4-and-grails-3-0-to-be-last-major-releases-under-pivotal-sponsorship*

[5] Moving Groovy to a Foundation: *http://www.groovy-lang.org/mailing-lists.html#nabble-td5722483*

[6] Tiobe-Index: *http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html*

Falk Sippach
falk.sippach@oio.de

Falk Sippach hat mehr als fünfzehn Jahre Erfahrung mit Java und ist bei der Mannheimer Firma OIO Orientation in Objects GmbH als Trainer, Software-Entwickler und Projektleiter tätig. Er publiziert regelmäßig in Blogs, Fachartikeln und auf Konferenzen. In seiner Wahlheimat Darmstadt organisiert er mit Anderen die örtliche Java User Group. Falk Sippach twittert unter @sippsack.

Frontend-Entwicklung mit ClojureScript und React/Reacl

Michael Sperber, Active Group GmbH

Welches ist das beste Framework, um ein GUI-Frontend in Java zu entwickeln? Swing, SWT oder gleich JavaFX? Es geht auch ganz anders: Seit Java 8 wird ein kompletter Web-Browser mit HTML5 und JavaScript mitgeliefert. So können JavaScript-Anwendungen auch innerhalb von Java-Anwendungen laufen.

HTML5 ist eine echte Alternative zu den „Native"-Toolkits. Diese Option wird besonders attraktiv durch React, das GUI-Framework für HTML5 von Facebook. Es erschließt eine völlig neue Art der GUI-Programmierung. Noch besser lässt sich React mithilfe von ClojureScript und dem React-Wrapper Reacl programmieren.

Zunächst einige grundsätzliche Gedanken zur GUI-Programmierung: Eine vernünftige GUI-Architektur setzt auf die Trennung von „Modell" (oder „Business-Logik") und „View". Unterschiede zwischen Patterns wie MVC oder MVVM sind für die Zwecke dieses Artikels nebensächlich. Damit die GUI stets den korrekten Zustand des Modells anzeigt, muss das Modell sie über Änderungen informieren. Bei „Native"-Toolkits für Java (Swing, SWT und JavaFX) geschieht das in der Regel durch „Callbacks" im Modell, um die GUI zu verändern, in Java gern über das Observer-Pattern *(siehe Abbildung 1)*.

Eine solche Applikation steuert die GUI auf zwei verschiedene Arten:

- Eine Funktion generiert beim Start der Applikation die GUI aus dem Anfangs-Modell
- Callbacks am Modell machen aus einer Veränderung des Modells eine Veränderung der GUI

Der zweite Punkt hat es in sich, weil es oft gar nicht so einfach ist, eine Veränderung der GUI zu finden, die genau der Veränderung des Modells entspricht. Außerdem gibt es Callbacks nicht nur vom Modell in die GUI,