Wolken-verhangen

Bernhard Steppan

Anfang des Jahres kündigte die Eclipse Foundation überraschend eine Cloud-Entwicklungsumgebung namens Eclipse Che an. Angepriesen wird sie als Open-Source-Alternative zu JetBrains' IntelliJ.

Cloud-Anwendungen liegen voll im Trend. Dass der sich nun auf Entwicklungsumgebungen ausdehnt, ist neu und war nicht unbedingt zu erwarten. Denn Desktop-IDEs bieten eine enorme Fülle an Funktionen und arbeiten mit vielen Programmiersprachen und Frameworks zusammen. Für Tyler Jewell, Projektleiter von Eclipse Che, scheint das keine Hürde zu sein, er hält Cloud-IDEs für die Zukunft der Softwareentwicklung. Jewell ist CEO des kalifornischen Start-ups Codenvy mit Sitz in San Francisco, das schon im Jahr 2009 mit der gleichnamigen Cloud-IDE einen ersten Versuch in diese Richtung gestartet hatte. Die Firma will Entwickler mit sofort gebrauchsfähigen Programmierwerkzeugen versorgen. Ganz neu ist die Idee nicht, denn auch von der Eclipse IDE lassen sich fertig konfigurierte Images verteilen. Nur befinden sich die nun in der Cloud und nicht mehr auf einer Workstation.

Bei der Namensgebung ihrer Projekte hat die Eclipse Community nicht immer eine glückliche Hand bewiesen. Schon bei der Einführung der klassischen IDE gab es Irritationen, weil manche Eclipse (Sonnenfinsternis) als eindeutige Anspielung auf den Untergang von Sun Microsystems verstanden. „Che" erscheint ebenfalls fragwürdig, da man diesen Namen automatisch mit dem berühmt-berüchtigten Revolutionär Che Guevara assoziiert. Laut Jewell leitet sich die Bezeichnung jedoch von der ukrainischen Stadt Cherkasy (dt. Umschrift „Tscherkassy") ab, wo die Cloud-IDE entwickelt wird (siehe „Alle Links"). Che ist ein neues Top-Level-Projekt der Eclipse Community, das derzeit als Release 4.01 vorliegt.

Nach der Installation (rund 280 MByte) liegen auf der Festplatte neben einer Reihe neuer Che-Bibliotheken alte Bekannte wie die Apache Commons Libraries, die Java Development Tools (JDT), einige Shell-Skripte und Batch-Dateien sowie ein kompletter Tomcat-Applikationsserver. Bevor sich die IDE starten lässt, möchte noch das Container-Tool Docker installiert werden. Che benötigt es zum automatisierten Verteilen der Anwendungen. Unter Linux lässt sich Docker nativ starten, unter Mac OS X und Windows gelingt es nur mithilfe einer virtuellen Maschine, da die Software einige Linux-Kernelfunktionen verwendet.

Von Grund auf neu entwickelt

Hat man die Installation erfolgreich hinter sich gebracht, präsentiert sich Che als eine moderne Single-Page-Webanwendung, die auf den ersten Blick nur wenige Gemeinsamkeiten mit der klassischen Eclipse IDE aufweist. Wer genauer hinschaut, dürfte erkennen, dass es sich um eine Applikation handelt, die auf bewährten Apache-Techniken aufsetzt: dem Build-Manager Maven und Tomcat als Laufzeitumgebung. Aufgrund zahlreich vorhandener Erweiterungen kommt Che mit allen gängigen Programmiersprachen klar (C++, Go, Java, PHP, Python und Ruby sowie mit vielen Frameworks wie Angular, Spring und Vaadin). Projektleiter Jewell betont, dass Che von Grund auf neu entwickelt wurde – und zwar von einem anderen Team als dem für die Eclipse IDE verantwortlichen. Entsprechend radikal ist der Bruch mit den Konzepten des Klassikers (siehe Kasten „Eclipse IDE versus Eclipse Che").

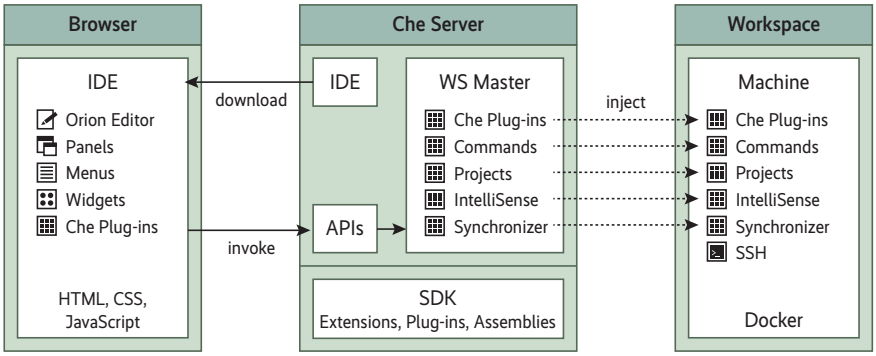
Che besteht aus drei Teilen: Im Browser präsentiert sich eine IDE, die aus

Eclipse IDE versus Eclipse Che

Eclipse Che und die klassische Eclipse IDE bilden verschiedene Strömungen innerhalb derselben Open-Source-Gemeinschaft. Beide Projekte verfolgen ein ähnliches Ziel, aber mit verschiedenen Mitteln: Die Desktop-IDE ist relativ statisch und entwickelt sich nur langsam weiter. Eclipse Che ist hingegen ein neues, dynamisches Projekt. Diese IDE bricht in vielerlei Hinsicht mit den Konzepten klassischer Desktop-Entwicklungsumgebungen.

Das zentrale Element von Che ist der Workspace Server. Durch ihn kann der Entwickler von praktisch überall auf seinen Arbeitsbereich samt Sourcecode sowie allen Einstellungen und Runtimes zugreifen. Er ist nicht mehr an einen physischen Computer gebunden, loggt sich einfach an einem beliebigen Rechner ein und ist sofort arbeitsfähig. Wer möchte, kann Che-Projekte via Maven-Import in den Arbeitsbereich einer Eclipse-IDE übernehmen.

Obwohl die Umgebungen unterschiedlich ausgelegt sind, richten sich doch beide an Anwendungs- und Tool-Entwickler. Eclipse Che ist nicht der Schlusspunkt für die klassische Eclipse IDE. Beide Umgebungen dürften über viele Jahre koexistieren.



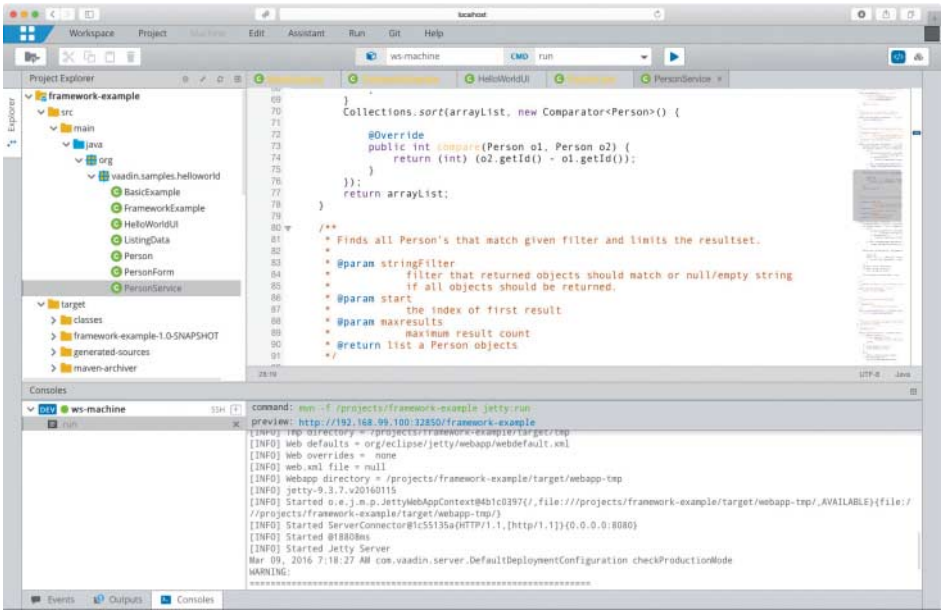
Die Architektur von Eclipse Che besteht aus drei Teilen: Browser-IDE, Server und Workspaces (Abb. 1).

dem Orion Editor, verschiedenen Panels, Menüs, Widgets und Plug-ins besteht. Sie kommuniziert mit dem Che Server über eine Reihe von APIs. Projekte und Plug-ins befinden sich in der dritten Komponente, dem auf Docker basierenden Workspace (Abbildung 1). Wie in der bekannten Eclipse IDE enthält ein Workspace Projekte der gleichen Art inklusive Bibliotheken und Grundeinstellungen. Bei Che wären beispielsweise Java-, JavaScript-, PHP-/WordPress- sowie ASP-Projekte zu nennen. Beispiele lassen sich ausgesprochen einfach per GitHub-Repository importieren und ohne die üblichen Konfigurationsorgien umgehend in Betrieb nehmen. Die Workspace-Verwaltung präsentiert sich zweigeteilt: Im linken Bereich befindet sich eine Baumstruktur mit Dashboard, Workspace und Projekten, rechts zeigen sich zugehörige Informationen. Hier legt der Entwickler auch neue Workspaces und Projekte an.

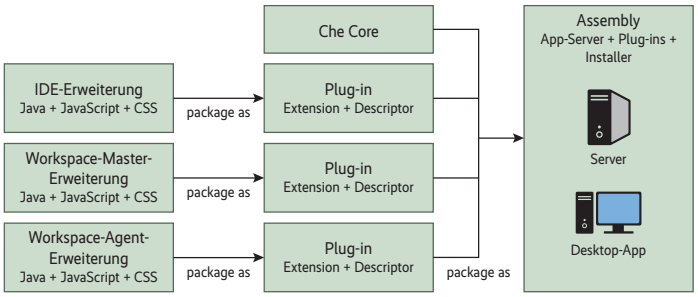
Markiert er in der Workspace-Verwaltung ein Projekt und klickt auf „Open in IDE“, erscheint es in einer IDE-Ansicht. Che kann mehrere Projekte gleichzeitig öffnen und mehrere Anwendungen parallel starten. Die Oberfläche der Entwicklungsumgebung ist klassisch aufgebaut und besteht aus einem Projekt-Explorer auf der linken und einem Editorbereich auf der rechten Seite. Das Standardfarbschema nach dem ersten Start erinnert an die Gestaltung der IntelliJ-UI. Wer es lieber heller und ergonomischer mag, wählt ein anderes Farbschema. Kern der IDE-Ansicht ist der erwähnte Orion Editor, ebenfalls ein Eclipse-Projekt, das speziell in Cloud-Umgebungen seinen Dienst verrichten soll.

Bedienung streckenweise umständlich

Der Code-Editor beherrscht die gängigen Grundfunktionen, darunter Zeilennummerierung, Syntaxhervorhebung, Syntaxüberprüfung und faltbare Codeabschnitte (Abbildung 2). Zudem besitzt er einen Programmierassistenten und gibt Empfehlungen zur Lösung von Problemen. Sowohl Geschwindigkeit als auch die Handhabung zum Beispiel mit Drag and Drop ist zufriedenstellend. An der Bedienung gäbe es dennoch einiges



Die Che IDE zeigt die klassische Aufteilung in Projekt-Explorer, Konsolen-Fenster und Editor (Abb. 2).



Über eine Java-API erweitert der Entwickler die IDE. Er kann sowohl ihr Erscheinungsbild als auch ihre APIs modifizieren (Abb. 3).

zu verbessern: So erschließen sich manche Tastenkombinationen erst nach dem Studium des Dialogs „Key Bindings“. Sie erscheinen nicht wie gewohnt im Menü.

Schlimmer sieht es bei den von den jeweiligen Programmiersprachen abhängigen Funktionen aus. Wer etwa die heute üblichen Refactoring-Funktionen erwartet, dürfte herb enttäuscht werden. Es gibt derzeit lediglich *Move* und *Rename*, was angesichts der knapp 30 Refactoring-Funktionen der Eclipse IDE, auf die man hätte zugreifen können, nicht dem Stand der Technik entspricht. Andere von klassischen IDEs bekannte Konfigurationsoptionen, etwa ein variabler Codeformatierer, fehlen ebenso.

Insgesamt wirkt die IDE-Ansicht zwar übersichtlich, ist allerdings auch spartanisch ausgestattet. An die Funktionsfülle einer modernen Desktop-IDE reicht sie bei Weitem nicht heran. Bei der Projektverwaltung muss der Benutzer ebenfalls Abstriche machen: Che bietet derzeit lediglich akzeptable Unterstützung für Konsolen- und Webprojekte. Beispielsweise gibt es keinen GUI-Builder für Swing-, SWT-/JFace- und RCP-Projekte. Fehlanzeige auch bei Importfunktionen für mit der Desktop-IDE angelegte Workspaces. Vermutlich sind sie wegen der Neuausrichtung von Che nicht so einfach umzusetzen.

Angesichts der zahlreichen fehlenden Funktionen beschleicht einen das Gefühl, dass sich die Che-Projektleitung in den vergangenen Jahren nur mit Cloud- und Webentwicklung beschäftigt hat und darüber vergaß, wie viele Rich-Client-Projekte im Java-Umfeld existieren. Dafür hat man in Sachen Webprojekte einiges vorgelegt: So gibt es Codebeispiele und Tutorials für WordPress, Spring und Vaadin. Diese Projekte lassen sich über GitHub importieren und durch das eingebaute Maven ohne Klimmzüge ausführen. Hierzu muss der Entwickler lediglich entsprechende Maven-Skripte wie Build oder Run erstellen. Die Dokumentation dazu ist sehr gut. Zusätzlich bietet die IDE voreingestellte Datenquellen wie PostgreSQL, MySQL, Oracle, MS SQL Server und NuoDB.

Die Ausführungsgeschwindigkeit der IDE, etwa beim Laden und Bauen von Projekten, lässt sich nur als durchschnittlich bezeichnen. Eine gut konfigurierte Desktop-IDE erledigt so etwas in der Regel deutlich schneller. Über ein mitgeliefertes SDK ist die Cloud-IDE erweiterbar. Das geschieht über Plug-ins, die jedoch nicht dem Eclipse-RCP-Standard folgen. Drei Erweiterungstypen sind im Angebot: IDE, Workspace Master und Workspace Agent (Abbildung 3). Über die IDE-Plug-ins kann der Entwickler das Look-and-Feel sowie Editoren, Assistenten, Menüs und Symbolleisten anpassen. Über Master fügt er neue Core-APIs hinzu und verändert bestehende. Und mit den Agent-Plug-ins bearbeitet und erzeugt er projektspezifische Erweiterungen. Das betrifft beispielsweise das Verhalten von Codevorlagen, Befehlen oder der Programmierhilfe.

iX-Wertung

⊕ einfache Bedienung
⊕ gute Dokumentation
⊕ Open Source

⊖ geringer Funktionsumfang im Vergleich zu Desktop-IDEs

Fazit

Freunde klassischer IDEs wie Eclipse und IntelliJ IDEA dürfen aufatmen: Auch wenn Che nach Umsturz klingt, bleibt die Revolution des Entwicklerarbeitsplatzes (vorerst) aus. In der momentanen Verfassung ist die Cloud-Entwicklungsumgebung weder leistungsfähig noch schnell genug, um den Desktop-IDEs das Wasser abzugraben. Doch die Beschäftigung mit dem Cloud-Werkzeug zeigt, welches Potenzial in modernen Webanwendungen steckt. Die Zukunft der Softwareentwicklung könnte in einer perfekt eingerichteten IDE liegen, die sich ohne Konfigurationsorgien von jedem beliebigen Arbeitsplatz aus bedienen lässt. (jd)

Bernhard Steppan

arbeitet als Leading Consultant bei SYRACOM Consulting in Wiesbaden und ist Autor des Buchs „Eclipse Rich Clients und Plug-ins“ (Hanser, 2015).

Alle Links: www.ix.de/ix1605052