„Wir sind jetzt unabhängig vom Zeitplan für Java EE 8 …“

MVC 1.0 sollte eine Ergänzung für Java EE neben dem existierenden Web-Framework Java Server Faces werden. Bei der Neuausrichtung von Java EE 8 hat Oracle den JSR über Bord geworfen. Starkes Feedback aus der Community hat den Konzern aber dazu bewegt, ihn zumindest außerhalb von Java EE 8 weiterzuführen – mit Ivar Grimstad als einem von Oracle (und den anderen großen Herstellern) unabhängigen Specification Lead: Sozusagen als „Community JSR“. Andreas Badelt, stellvertretender Leiter der DOAG Java Community, wollte wissen, wie es um den JSR steht.

Oracle hat den MVC 1.0 JSR an die Community übergeben. Du bist jetzt gemeinsam mit Ivar Grimstad Specification Lead. Was ist dabei deine Rolle?

Die Aufgaben des Spec Lead sind im Java Community Process (JCP) sehr genau definiert. Er ist verantwortlich dafür, dass der JSR im zeitlich vorgegebenen Rahmen die entsprechenden Ergebnisse liefert, beispielsweise die ersten Entwürfe veröffentlicht bis hin zum Release der finalen Spezifikation. Rechtlich gesehen trifft er alle Entscheidungen und könnte sich sogar über die Meinung der Expert Group hinwegsetzen. In vielen JSRs wird der Spec Lead als der inhaltliche Treiber gesehen und die Expert Group hat lediglich eine beratende Funktion. Unser Ziel für JSR 371 ist es, die Expert Group in einer möglichst demokratischen Art zu führen und die Spezifikation gemeinsam zu erarbeiten. Dazu gehört beispielsweise, dass wir viele Abstimmungen abhalten, die dann nach Mehrheit entschieden werden. Als Spec Leads versuchen wir, den notwendigen Rahmen zu liefern, damit die Expert Group gemeinschaftlich zum bestmöglichen Ergebnis kommen kann.

Die Übergabe des MVC durch Oracle hat länger gedauert als geplant. Was war der Hintergrund und kannst du Tipps für mögliche zukünftige Community-JSRs geben?

Die besondere Schwierigkeit für MVC war sicherlich, dass die Übergabe des JSR mitten im Prozess stattfand. Dadurch wurde sie aus rechtlicher Sicht sehr viel komplizierter. Im JCP hat der Spec Lead auch immer alle Rechte am geistigen Eigentum der Spezifikation. Bei einem Wechsel des Spec Lead müssen diese Rechte auf den neuen Leiter übertragen werden. Wie man sich vorstellen kann, ist das ein komplexer und zeitaufwändiger Prozess. Glücklicherweise handelte es sich bei dieser Übergabe um einen Sonderfall, der gerade deshalb sehr kompliziert war. Ich denke, dass sich die Situation für zukünftige Community JSRs deutlich einfacher gestaltet, wenn der entsprechende JSR von Anfang an von der Community geleitet wird. In diesem Fall ist die rechtliche Situation von Beginn an klar geregelt und die Expert Group kann sich auf die eigentliche Arbeit konzentrieren.

Wie sind jetzt die rechtlichen Rahmenbedingungen bezüglich des Quellcodes und insbesondere des TCK?

Nach Übernahme des JSR haben wir uns schnell für eine Änderung der Lizenz entschieden. Sowohl die Spezifikation als auch die Referenz-Implementierung und das TCK stehen inzwischen unter der freien Apache-Lizenz. Wir sind der Meinung, dass eine freie Lizenz ganz wesentlich für den Erfolg des JSR ist. Nur sie erlaubt Herstellern und Endnutzern einen sehr flexiblen Umgang mit der Spezifikation und öffnet die Tür für Beiträge aus der Community. Bei der Entscheidung bezüglich der Lizenzierung waren besonders CDI und Bean Validation unsere Vorbilder, die von Red Hat seit jeher in sehr offenerer Art geführt wurden.

MVC ist aus dem Java-EE-8-Umbrella herausgenommen worden und läuft jetzt als unabhängiger JSR. Wie bewertest du die Konsequenzen?

Nachdem bekannt wurde, dass MVC 1.0 nicht Bestandteil von Java EE 8 sein wird, ist die Spezifikation aus vielen Köpfen verschwunden. Viele haben diese Entscheidung als Todesurteil für MVC interpretiert. Faktisch heißt das aber lediglich, dass Java-EE-8-Applikationsserver nicht zwingend MVC unterstützen müssen. Aber natürlich dürfen sie mehr Funktionalität bieten, als es die Java-EE-8-Spezifikation vorschreibt. Unterstützt der Applikationsserver MVC trotzdem nicht, muss man in diesem Fall lediglich eine MVC-Implementierung mit in seinem WAR-Archiv verpacken. Auch wenn die MVC-1.0-Spezifikation nicht in Java EE enthalten ist, bleibt sie natürlich trotzdem ein Standard des Java Community Process – mit allen positiven Effekten wie beispielsweise der Langlebigkeit und Stabilität des API. Außerdem ist die aktuelle Situation natürlich noch mehr ein Ansporn, besonderen Wert darauf zu legen, dass die Referenz-Implementierung auch in allen Applikationsservern und in Java SE funktioniert, etwa in einem Apache Tomcat oder Jetty. Natürlich sind wir jetzt unabhängig vom Zeitplan für Java EE 8, der aufgrund der genannten Verzögerungen für MVC sowieso kaum einzuhalten gewesen wäre.

War es im Nachhinein die richtige Entscheidung, auf der JAX-RS-Spezifikation als Basis für MVC aufzusetzen?

Aus meiner Sicht war es definitiv eine gute Idee. JAX-RS bietet bereits viel Funktionalität, die auch für MVC in ähnlicher Form benötigt wird. Wäre JAX-RS nicht Basis gewesen, hätten viele Konzepte dupliziert werden müssen, was für die Einheitlichkeit der Plattform sicherlich nicht von Vorteil gewesen wäre. Außerdem ist das JAX-RS-API unter Entwicklern gut bekannt. Der Einstieg in MVC 1.0 fällt somit tendenziell leichter. Trotzdem war die Entscheidung in der Expert Group nicht unumstritten. Der Nachteil ist sicherlich, dass wir uns in gewisser Weise abhängig von JAX-RS und somit von dessen Funktionsumfang und Entwicklungsgeschwindigkeit gemacht haben.

In der Java-EE-Panel-Diskussion im JavaLand kamen ein paar Abhängigkeiten gegenüber JAX-RS zutage, die im Rahmen des Java-EE-8-Prozesses ignoriert worden waren. Wie ist da der aktuelle Stand?

Es war in der Tat so, dass die JAX-RS-Spezifikation einige Lücken aufwies, die für uns zum Problem wurden. Die JavaLand-Konferenz bot eine sehr gute Gelegenheit, diese Thematik im persönlichen Gespräch zu erörtern und nach Lösungswegen zu suchen. Glücklicherweise hat die JAX-RS-Expert-Group inzwischen reagiert und die Spezifikation entsprechend erweitert – trotz des Termindrucks hinter EE 8. Die genannten Änderungen werden Teil von JAX-RS 2.1 sein, sodass wir auf der neuen Funktionalität aufbauen können.

Warum braucht das Java-Ökosystem MVC?

Die Frage nach dem besten Web-Framework wird seit jeher kontrovers diskutiert. Es gibt verschiedene technische Ansätze, die je nach Anwendungsfall mehr oder weniger gut funktionieren. Mit Java Server Faces ist in Java EE seit mehr als zehn Jahren ein komponentenorientiertes Web-Framework enthalten. In vielen Fällen sind aktionsbasierte Frameworks besser geeignet, da sie leichtgewichtiger sind und üblicherweise mehr Kontrolle über technische Details bieten. Für Java EE soll MVC 1.0 diese Lücke schließen. Natürlich ist beispielsweise Spring MVC ein sehr populärer Vertreter dieser Kategorie und in vielerlei Hinsicht auch Vorbild für MVC 1.0. Trotzdem wäre es in den meisten Projekten nicht sinnvoll, Spring einzig und allein für das Web-Framework einzubinden, obwohl man eigentlich auf Java EE setzen möchte. Daher denke ich, dass MVC 1.0 ein wichtiger Standard für die Plattform ist.

Wie ist der aktuelle Stand der Spezifikation und wie sieht der Zeitplan für das erste Release aus?

Leider haben die Übergabe der Spezifikation und die Aktualisierung der Lizenz sehr viel Zeit gekostet. Zusätzlich hat Oracle die *„java.net"*-Plattform eingestellt, auf der bisher unsere komplette Infrastruktur bereitstand. Somit mussten wir zu allem Überfluss auch noch ein neues Zuhause für die Spezifikation finden. Das hat dafür gesorgt, dass wir bezüglich des Zeitplans umdenken mussten. Die Spezifikation selbst ist bereits auf einem sehr guten Stand. Trotzdem wollen wir in den kommenden Monaten alle Aspekte noch einmal durchgehen und genau prüfen. Es gibt auch durchaus einige Punkte, die noch komplett offen sind. Dazu gehört beispielsweise, dass der Mechanismus für das Data Binding die Sprache korrekt berücksichtigen muss, damit die Verarbeitung von Dezimalzahlen und Datumsangaben korrekt funktioniert. Aktuell planen wir, im Herbst in das „Public Review" zu gehen. Danach folgt dann Anfang 2018 der „Proposed Final Draft" und – wenn alles gut geht – Mitte 2018 das „Final Release".

Was sind die nächsten Schritte und wie kann euch die Community dabei am besten unterstützen?

Wie bereits erwähnt, planen wir, die Spezifikation in den nächsten Monaten zu stabilisieren. Wir müssen uns dann bei jedem Aspekt fragen: Ist die Formulierung genau genug? Sind alle Spezialfälle abgedeckt? Gibt es Widersprüche? Ist noch unerwünschter Interpretationsspielraum vorhanden? Deckt die Spezifikation die Anforderungen realer Projekte ab? Besonders der letzte Punkt ist natürlich ganz wesentlich. Die Expert Group kann stets nur aufgrund der eigenen Erfahrung urteilen. Vielleicht hat der normale Entwickler ja eine ganz andere Sichtweise. Ist es aktuell einfach genug, bestimmte Standardfälle zu implementieren? Diese Fragen können uns natürlich am besten Entwickler beantworten, die sich im täglichen Job mit Web-Applikationen beschäftigen. Jeder ist eingeladen, an den Diskussionen auf der Mailingliste teilzunehmen.

Links

[1] *Homepage: https://www.mvc-spec.org*
[2] *Mailing List: https://groups.google.com/forum/#!forum/jsr371-users*
[3] *Issue Tracker: https://github.com/mvc-spec/mvc-spec/issues*
[4] *Reference-Implementation: https://www.mvc-spec.org/ozark*
[5] *Twitter: https://twitter.com/mvc_spec*

Hinweis: Die DOAG Java Community möchte den MVC JSR im Rahmen des „Adopt a JSR"-Programms unterstützen. Hierzu wird es in der zweiten Jahreshälfte 2017 Veranstaltungen geben, die unter anderem über Twitter („@DOAGJava" beziehungsweise „#DJC") und im iJUG-Newsletter angekündigt werden.

Zur Person:
Christian Kaltepoth

Christian Kaltepoth *(„christian@kaltepoth.de")* arbeitet als Senior Developer bei der ingenit GmbH & Co. KG in Dortmund. Sein Schwerpunkt ist die Entwicklung von webbasierten Unternehmensanwendungen auf Basis von Java EE. Seit Mai 2017 ist er zusammen mit Ivar Grimstad MVC 1.0 Specification Lead. Twitter: „@chkal"