



MYTHOS ODER WAHRHEIT: ALLHEILMITTEL XML

Ein Rückblick auf 20 Jahre Extensible Markup Language

| von TIM POMMERENING



„Mit einem Hammer sieht jedes Problem plötzlich wie ein Nagel aus.“ XML – als erweiterbare Auszeichnungssprache für Dokumente erfunden – hat sich in der Frühzeit des World Wide Webs seit Ende der 1990er-Jahre schnell in die schmalsten Winkel der IT ausgebreitet. Wo aber ist der Einsatz wirklich sinnvoll, wo hingegen nicht und was blieb vom damaligen Hype übrig?

XML ist heute allgegenwärtig. Spätestens seit den frühen 2000er-Jahren hat das Datenformat in allen Winkeln der Softwareentwicklung Einzug gehalten. Zur flexiblen Speicherung von Dokumenten etwa ist diese Auszeichnungssprache eine Bereicherung. XML hat sich jedoch auch in viele andere Bereiche der IT vorgewagt, die rückblickend betrachtet besser davor verschont geblieben wären. Auch das ist Teil des (vergangenen) XML-Hypes – und wir leiden oft noch heute darunter. Die folgenden Beispiele zeigen aus heutiger Sicht die Success-Stories und Failed States von XML, einer Technologie, die natürlich doch kein Allheilmittel ist.

XML IST FLEXIBEL UND PASST FAST IMMER

Vor XML wurden häufig Datenformate wie CSV oder FixedRecord verwendet. Im Fall von CSV wird ein Trennzeichen festgelegt, um Datenfelder abzugrenzen, während bei FixedRecord das folgende Datenfeld nach der vorher festgelegten Feldlänge ohne jeglichen Trenner beginnt. Beide Formate benötigen Sonderbehandlungen bei Zeilenumbrüchen. Vor allem FixedRecord verzeiht fast keine nachträglichen Formatkorrekturen, da sich alle Folgefelder dadurch verschieben. Dann kam XML und bot einen viel flexibleren Ansatz. Zeilenumbrüche sind kein Problem. Nachträgliche Erweiterungen sind prinzipiell ebenso möglich. Außerdem dürfte es bisher sehr selten vorgekommen sein, dass sich eine spontan zugunsten des XML-Formats ausgefallene Entscheidung nachträglich als grundsätzlich falsch herausgestellt hätte.

Neben zeichenbasierten Formaten existieren auch jede Menge Binärformate, wie etwa für Bilder. Selbst da bietet XML mit SVG ein Datenformat zum Speichern von Vektorgrafiken an. Im Office-Bereich hat XML inzwischen auch das vorher vorherrschende Binärformat ersetzt.

XML KANN BAUMSTRUKTUREN ABBILDEN

Häufig besteht die Anforderung an Datenformate, Bäume – also 1:n-Beziehungen zwischen Objekten oder Knoten – abbilden zu können. Einfache Standards wie zum Beispiel CSV eignen sich dazu nur mittels „Tricksereien“. Dagegen unterstützt XML beliebige Baumstrukturen durch die Verschachtelung von XML-Elementen, sodass keine Workarounds anderer Formate nötig sind, die in jedem Fall individuell abgestimmt werden müssen. Wie die Schachtelung in einem XML-Dokument funktioniert, ist für jedermann praktisch sofort ersichtlich.

XML ZUR ANWENDUNGSKONFIGURATION

Sind XML-Dateien schon für Programmierer kein effizientes Medium in der Erstellung, so sind sie es für Administratoren in der Unix-Kommandozeile noch viel weniger. Die Philosophie der zeilenbasiert arbeitenden Unix-Shell ist eine ganz andere. Mitte der 2000er-Jahre hatte sich, insbesondere durch den SOA-Hype getragen, eine Philosophie entwickelt, Programmierung durch Konfiguration zur Laufzeit zu ersetzen. Es sollten Komponenten mit so allgemeinen Schnittstellen entwickelt werden, dass sie, wenn einmal vorhanden, in verschiedenen Kontexten zur Laufzeit eingebunden werden können. Durch deren Kombination sollten zur Laufzeit neue Anwendungen erstellt oder Komponenten durch andere ausgetauscht werden können. Für die dafür notwendigen

immer komplexer werdenden Konfigurationsdateien wurde XML verwendet. Mit Frameworks wie Spring ging das so weit, dass komplexe Aufrufhierarchien über mehrere XML-Dateien hinweg „konfiguriert“ wurden und außerhalb der Entwicklung niemand auf die Idee gekommen wäre, daran Hand anzulegen. Gleichzeitig keimte in der Entwicklung die Erkenntnis auf, dass ein Compiler vielleicht doch eine ganz sinnvolle Sache war. Heute findet man XML-Höllen aus dieser Zeit zwar noch immer vor, aber die Tendenz geht seit einigen Jahren wieder ganz klar zur Konfiguration zur Compile-Zeit – also Programmcode – zurück. Die Entscheidung darüber hängt im Wesentlichen von den Qualitätsanforderungen an die Anwendung ab, die bei jeder Anwendung neu diskutiert werden müssen. Grundsätzlich bietet es sich jedoch eher bei Produkten als bei der Individualentwicklung an, Ersetzung und Umkonfiguration ganzer Komponenten zur Laufzeit vornehmen zu können, als bei der Individualentwicklung.

Nicht vergessen werden sollte unter dem Aspekt „Konfiguration“ auch das Hinterlegen von einfachen Schaltern in Konfigurationsdateien. Auch die bis dahin gängigen .properties-, .conf- oder .ini-Dateien wurden zwischenzeitlich durch XML-Dateien ersetzt. Das führt in den meisten Fällen zu einem Lesbarkeitsdesaster und ist nur dann sinnvoll, wenn Baumstrukturen abgebildet werden müssen. Nach Abebben des XML-Hypes setzten sich hier nach und nach wieder die ursprünglichen Formate durch. Heute wird bei Bedarf an Baumstrukturen eher das gut lesbare .yaml-Format¹ (sprich jammel, für yet another markup language) verwendet, das gegenüber .xml durchaus auch Nachteile hat, aber durch eine viel bessere Lesbarkeit punktet.

XML ALS PROGRAMMIERSPRACHE

Ruft man sich in Erinnerung, dass XML in einer Sprache zum Erstellen von Build-Prozessen (Apache ANT²) sogar als Syntax für eine Skriptsprache erhalten musste, so lässt sich das aus heutiger Sicht nur mit dem Hype ein Jahr nach der Veröffentlichung von XML erklären. ANT ist Programmierung. XML ist ausgesprochen schlecht lesbar und nicht minder schlecht zu schreiben. Trotzdem fiel die Entscheidung der Entwickler auf XML als Programmiersprache, die als Sprachelemente eine Vielzahl von XML-Elementen aufweist. Heute käme niemand mehr auf die Idee, bei einer für Programmierer gedachten Sprache eine XML-Syntax zu verwenden.

XML ZUR DATENÜBERTRAGUNG

Wenn es etwas gibt, für das sich XML überhaupt nicht eignet, ist es Datenübertragung. Datenübertragung – eine kontroverse Aussage! Schließlich ist es in Java heute nach wie vor der Standard

dafür. XML-Dokumente sind wegen ihrer ausufernden Datenstrukturen in absoluten Zahlen riesig. Gleichzeitig ist der Informationsgehalt in Relation zum Rahmen gering. Daher versenden Anwendungen über Netze unglaublich große Dokumente mit unglaublich geringem Informationsgehalt. Das Gegenargument dazu ist, dass sich XML gerade wegen des geringen Informationsgehalts fantastisch komprimieren lässt. Ein Sender erzeugt also zuerst, unter hohen CPU- und Speicherkosten, riesige lesbare XML-Strukturen, um sie dann – wieder mit hohen CPU-Kosten – zu komprimieren, also in ein Binärformat zu konvertieren. Nachdem sie daraufhin effizient über Netze versendet wurden, muss der Empfänger erneut mit hohen CPU- und Speicherkosten zuerst dekomprimieren und dann parsen. Warum also nicht besser gleich ein genauso offenes spezifiziertes Binärformat nutzen, das auf eine effiziente Datenübertragung ausgelegt ist? Dieser verschwenderische Umgang mit Ressourcen ist sowieso nur möglich, weil Rechenleistung in der Vergangenheit immer erschwinglicher wurde und einfach erhöht werden konnte. Aber Green-IT geht anders. Dennoch ist gerade in der Java-Welt XML heute der Datenaustauschstandard. Letztlich besteht auch keine Not, von diesem Standard abzuweichen, und eine solche Abweichung müsste gut begründet werden, denn mit dem Standard ist man auf der sicheren Seite. Anders sieht es hingegen im Mobilbereich aus. Da sowohl aus historischen als auch immer noch aus realen Gründen mit geringeren Datenübertragungskapazitäten und geringer CPU-Leistung (Akku-Verbrauch) auf Effizienz geachtet werden muss, kommen hier häufiger flexiblere und leichtgewichtige Datenformate, wie zum Beispiel JSON³, zum Einsatz.

FAZIT

Ganz klar: XML ist nicht der Heilsbringer der Daten- und Informationskodierung, und wir müssen uns in bestimmten Bereichen nur deshalb heute damit auseinandersetzen und oft auch quälen, weil eigentlich sinnvolle Technologien zu einer Zeit entstanden sind, als XML im Hype-Zyklus ganz oben war. Solche Fehlentscheidungen werden heute, nachdem der Hype abgeflaut ist, nach und nach revidiert. Aber einiges hat nach wie vor Bestand, etwa die Möglichkeit, mit Stylesheets das XML-Schema inklusive Datentypen so eng zu definieren, dass sich ein Nutzer auf deren Einhaltung verlassen kann. Gerade in dem Bereich, für den XML ursprünglich angetreten ist – also als Dokumentenauszeichnungssprache – konnte es sich im Internet nie durchsetzen. Zwar wurde das mit XHTML versucht, doch „XHTML5“ sucht man vergebens. Statt Markup schreibt man Publikationen heute in leichtgewichtigen Markdown-Derivaten – Formate, die auch in ihrem Quelltext recht angenehm von Menschen zu lesen und schreiben sind, während beides mit XML zur Qual wird. Im Web findet es sich nach wie vor bei abonnierbaren Feeds und wird uns auch darüber hinaus als Datenaustauschformat sicher noch lange erhalten bleiben. ●

1 <https://de.wikipedia.org/wiki/YAML> (abgerufen am 10.09.2019)

2 https://de.wikipedia.org/wiki/Apache_Ant (abgerufen am 10.09.2019)

3 https://de.wikipedia.org/wiki/JavaScript_Object_Notation (abgerufen am 10.09.2019)