JAVA Mag

Java magazin

Java | Architektur | Software-Innovation

DIE NEUE GENERATION BIG DATA

SMACK

Sonderdruck für ARS

www.ars.de

IBM Liberty im Schnellcheck

Doch nicht tot

Wollen Sie jetzt wirklich einen Artikel über einen Java EE Application Server lesen? Wenn Sie zustimmen, dass der Java-EE-Stack noch lange eine Säule moderner IT ist und meinen, dass Application Server gut, zuverlässig, standardkonform, aber gleichzeitig auch innovativ und cool sein sollten, dann hat – für manche unerwartet – IBM etwas zu bieten. Der vorliegende Artikel gibt einen Überblick über IBM Liberty, was diesen Application Server besonders macht und warum man sich damit beschäftigen sollte.

von Reinhard Hohberger und Joachim Gucker

IBM WebSphere Liberty – mancher muss da seine Antipathie dem Namen gegenüber wohl überwinden – ist ein vollwertiger und für das Java EE 7 Full Profile zertifizierter Server, der voll für den Produktionseinsatz tauglich ist und auch durch den Hersteller Support erhält. Der Einstieg sieht so aus: Nach einem Download von 50 bis 100 MB, je nach Variante, wird Liberty mit einem Unzip installiert, nach einem Kommando kann der Server gestartet werden. Die Konfiguration verläuft auch in wenigen Schritten: Wenn gewünscht, genügt ein XML-File. Wer Tomcat nutzt, kennt es nicht anders. Wer die Konfiguration bei IBM gewohnt war, dem war bis dato derartige Schlichtheit fremd. Die Konfiguration folgt dem Configuration-by-Exception-Konzept und erlaubt eine Strukturierung sowie Standardisierung durch *Includes*. Auch wenn Liberty weit weniger bekannt ist als Tomcat oder JBoss/WildFly, halten sich doch einige Mythen dazu hartnäckig. Einige davon wollen wir erörtern.

Ein Mythos zur Konfiguration ist, dass es kein grafisches Konfigurationswerkzeug gäbe. Wer ein solches möchte, darf sich des WebSphere Liberty Server Configuration Tools [1] bedienen. Die Konfiguration per XML-File oder UI dient aber noch einem weiteren Zweck, den man von anderen Java EE Application Servern weniger oder gar nicht kennt: der Definition der durch den Application Server bereitzustellenden Features. Der Server ist vollkommen modular aufgebaut. Durch die Wahl der Module in der Konfiguration schneidert der Entwickler oder der Administrator sich seinen Server passgenau.

Modular aufgebaut

IBM hat das Liberty Profile als von Grund auf modularen Server mit feiner Granularität entwickelt. Zur Modularisierung wird standardisierte und erprobte OSGi-Technik verwendet, die bekanntlich auch die Themen Versionierung und Abhängigkeiten zwischen Komponenten sauber abbildet. Ein Kernel des Servers lädt die vorgegebenen Module beim Start und gegebenenfalls automatisch auch weitere Module, die benötigt werden. Mit der internen Modularisierung mittels OSGi hatte IBM bereits im traditionellen WebSphere Application Server V6.1 im Jahr 2006 begonnen, aber hier konnte man den nicht verwendeten Portlet-Container oder JCA-Container aus dem Hauptspeicher verbannen. Die Granularität der Modularisierung von Liberty ist wesentlich feiner und so unserer Ansicht nach einmalig bei Java-EE-Servern. Um ein nahe liegendes Missverständnis gleich auszuräumen: Für den eignen Java-EE-Code lässt sich OSGi natürlich auch verwenden – wie bei vielen Application Servern. Die Liste der Features und ihrer Module finden Sie hier [2]. Ein Beispiel daraus: Die von Ihnen eingesetzte Java-EE-Applikation benötigt JSF. Durch entsprechende Konfiguration wird das JSF-Feature in Liberty geladen und besorgt sich – durch OSGi

Über die Neuerung Liberty Repository stellt IBM kontinuierlich und ohne fest getakteten Zeitplan neue Funktionen wie zusätzliche APIs, Features, Tools, Skripte oder Snippets zur Konfiguration des Servers bereit.

– automatisch das Servlet- und JSP-Feature oder deren Module. Nutzt die Applikation kein JSF- oder JSP-basiertes UI, sondern reine Servlet-Aufrufe, dann wird nur das Servlet-Feature geladen. So wird kein CPU-Zyklus oder Hauptspeicher für die JSP- oder gar JSF-Funktionalität verschwendet.

Voll Java-EE-7-zertifiziert

Ein weiterer Mythos betrifft die Spezifikation: Liberty unterstütze nur das limitierte JEE Web Profile. Aus Entwicklersicht erfreulich ist bei dem bereits genannten Featurekonzept, dass Liberty seit der Version 8.5.5.6 als vollzertifizierter Java EE 7 Application Server zählt [3]. Damit ist er, was dieses aktuelle Programmiermodell angeht, schon deutlich weiter als der klassische WAS. Er unterstützt außerdem die Verwendung von Java SE 1.6, 7.0 und 8.0, und zwar nicht nur in der hauseigenen Implementierung von IBM, sondern beispielsweise auch die von Oracle. Produktionstauglicher Support für Liberty wird über die Standardverträge von IBM angeboten. Es ist dabei zu differenzieren, welche Edition man erwirbt. Die Bandbreite reicht hier von einer einfachen, lediglich Java-EE-7-Web-Profile-zertifizierten Variante bis hin zu der voll clusterfähigen Edition, die mehr als nur Java EE 7 bietet.

Zero Migration

Die beschriebene Modularität erlaubt es, mit Liberty auch neue Wege zu beschreiten, um den Einsatz von Java EE effizient und agil zu gestalten. Bei einer Migration der Laufzeitumgebung auf eine höhere Version entfällt erheblicher Aufwand darauf, die bestehenden Anwendungen für die neue Version zu testen und gegebenenfalls anzupassen. Aufgrund der Rückwärtskompatibilität vieler Spezifikationen ist der Umfang der Anpassungen für die einzelne Anwendung meist gering, er ließ sich jedoch bisher nicht vermeiden. Zudem gibt es gerade in größeren Umgebungen häufig hundert und mehr Anwendungen, die hier betroffen sein können. Und erst wenn auch die letzte Anwendung angepasst wurde, kann der Migrationsprozess abgeschlossen werden. Die Folge sind vorab schwer zu kalkulierende Projektlaufzeiten, die sich nicht selten über viele Monate hinziehen.

Was bedeutet ein Update einer Application-Server-Landschaft üblicherweise? Die neue Version des Servers bringt ein Update in vielen Teilen der Java-EE-Implementierung, d. h. eine Servlet-3.0-Implementierung weicht beispielsweise einer Servlet-3.1-Implementierung. Bei IBM Liberty sind das nur Features, und die neue Version der Features ersetzt nicht zwingend die ältere Version oder die älteren Versionen. Unterschiedliche Versionen des Features stehen damit zur Wahl. Der Kunde behält die Wahl auf Featureebene, wo er Technologie ändert und modernisiert und wo er der Devise „Never touch a running system“ folgen möchte. Wenn der Kunde alle Features, die eine Applikation benötigt, unangetastet lassen möchte, dann entfallen beim Update auf eine neue Version des Applikation Servers im Prinzip sämtliche Migrationsaufwände auf der Anwendungsseite.

Die OSGi-basierte Architektur sorgt zudem dafür, dass Inkompatibilitäten zwischen Features bzw. bestimmten Versionen von Features schon beim Serverstart gemeldet werden und nicht erst zur Laufzeit. Eine gleichzeitige Verwendung von beispielsweise *servlet-3.0* und *websocket-1.1* würde somit schon von Anfang blockiert werden.

Continuous Delivery neuer Features

Neue Wege beschreitet IBM auch bei der Weiterentwicklung und Pflege von Liberty. Die traditionelle Art, zusätzliche Funktionen über neue Releases und Refresh Packs einzubringen, hat sich schon in vielen Projekten als Fortschrittsbremse erwiesen. Zu selten gibt es Updates. Hier gibt es von IBM zwei Ansätze zur Lösung, die man unter dem Schlagwort Continuous Delivery Model vermarktet. Zum einen ist es nun üblich, dass neue Funktionalität auch über die quartalsweise herausgegebenen Fix Packs zur Verfügung gestellt wird – etwas, das es bisher nur in eingeschränkter Form gab, meist in Form von zusätzlichen Properties zur Feinkonfiguration einer einzelnen Komponente des Application Servers. Nun aber werden auf diesem Weg komplett neue Features ausgerollt: *ejb-3.2*, *batch-1.0*, *jacc-1.5* und zahlreiche weitere wurden mit Fix Pack 8.5.5.6 vorgestellt, sodass schließlich die Java-EE-7-Zertifizierung ab diesem Level möglich war. Die eigentliche Neuerung ist jedoch das Liberty Repository. Hierüber stellt IBM kontinuierlich neue Funktionen bereit – ohne fest getakteten Zeitplan und ohne langwierige Ankündigung. Es kann sich dabei um zusätzliche APIs, Features oder Tools handeln,

aber auch um Skripte oder Snippets zur Konfiguration des Servers. Wer Unterstützung bei der Integration von Open-Source-Komponenten wie Struts, RichFaces, Hibernate und anderen mit Liberty sucht, wird dort ebenfalls fündig.

Angesprochen wird das Repository entweder aus dem IBM Installation Manager, direkt über die Webseite [4] oder aus den für Eclipse-Umgebungen erhältlichen Developer-Tools. Such- und Filterfunktionen helfen dabei, die geeigneten Downloads aus den über hundertfünzig Einträgen dort zu finden. Für bereits bestehende Liberty-Installationen kann der Zugriff auch über die Kommandozeile erfolgen, wobei hier automatisch auch Abhängigkeiten von Features berücksichtigt werden.

Sweet Spots in modernen Architekturen

Trotz all dieser Euphorie muss man natürlich die Realität in den Rechenzentren betrachten, um die besten Einsatzszenarien für Liberty ausfindig zu machen. Häufig existiert schon eine Java-EE-Application-Server-Umgebung mit eingespielten und womöglich automatisierten Abläufen hinsichtlich der Serverinstallation, dem Deployment oder dem Testen von Anwendungen. Ein neues Produkt hier einzufügen, will gut durchdacht sein. Da es die vom traditionellen WebSphere Application Server (WAS) bekannte Skriptsprache *wsadmin* für Liberty nicht gibt, sind solche Prozesse auch von einem WAS nicht ohne Aufwand auf Liberty übertragbar.

Noch ein Mythos zur Zielgruppe: Liberty ist nur für Entwickler gedacht. Dabei muss gesagt werden, dass es ohne Einschränkung in Produktionsumgebungen eingesetzt werden kann. Es gibt Clustermechanismen, die Ausfallsicherheit und Lastverteilung ermöglichen. Größere Umgebungen können zentral über eine Schnittstelle administriert werden – Stichwort Collectives. Das Skalierungsverhalten ist hier sogar besser als beim traditionellen WAS.

Somit wird es beim Einsatz von Liberty weniger darum gehen, eine vorhandene Infrastruktur auf einen Schlag abzulösen. Stattdessen ergibt es mehr Sinn, sich spezifisch die Projekte auszuwählen, bei denen die Stärken von Liberty besonders gut zum Tragen kommen. Dies sind beispielsweise dynamische, moderne Laufzeitarchitekturen wie Docker oder Cloud. IBM hat hier mit der Platform-as-a-Service-Lösung Bluemix bereits Voraussetzungen für Liberty geschaffen. Ebenso bietet sich Liberty für hoch skalierende Serverlandschaften mit möglicherweise tausenden von spezialisierten Servern an, da sich dort der schlanke und modulare Aufbau von Liberty besonders eignet. Ist man jedoch nicht darauf angewiesen, auf bestehende Java-EE-Landschaften Rücksicht nehmen zu müssen, so sprechen viele Argumente für Liberty: ein zertifiziertes Programmiermodell mit Java EE 7, Produktionstauglichkeit durch Skalierbarkeits- und Ausfallsicherheitsmechanismen und der bewährte Support durch IBM. Und: Die IBM-Fraktion im Unternehmen hat endlich mal wieder ein wirklich cooles Produkt vorzuweisen.

Reinhard Hohberger studierte Mathematik und Informatik an der Universität Bayreuth und arbeitete von 1995 bis 2014 für IBM. Zuletzt war er dort seit 1998 technischer Experte für die IBM WebSphere Application Server Platform. Bei ARS ist er seit 2014 als Berater und Trainer unter anderem im Bereich „JEE Application Server“ tätig.

Joachim Gucker studierte Informatik und Betriebswirtschaft an der TU München. Seit 1997 ist er bei der ARS Computer und Consulting GmbH. Mit Sitz in München ist ARS in IT-Beratung und Projekten aktiv. Der Schwerpunkt der ARS liegt auf Software Engineering, speziell mit IBM-Software und Open-Source-Software. 2006 wurde Joachim Gucker hier in die Geschäftsleitung berufen. Er ist bis heute auch als Spezialist für mehrere IBM-Softwareprodukte und im IBM-Lizenzmanagement tätig.

Links & Literatur

[1] WebSphere Liberty Server Configuration Tool: http://serverconfig.mybluemix.net/

[2] IBM WebSphere Liberty: www.ars.de/web/resources/Liberty.pdf

[3] Java EE Compatibility: http://www.oracle.com/technetwork/java/javaee/overview/compatibility-jsp-136984.html

[4] Liberty Repository: https://developer.ibm.com/wasdev/downloads/