

Ein Überblick über die aktuellen IDEs und Werkzeuge für JSF

Gesichtsmasken

■ VON ANDY BOSCH

Seit etwas mehr als einem Jahr steht der Entwicklergemeinde der neue UI-Standard JavaServer Faces zur Verfügung. Nach einer ersten Hype-Phase, in der hauptsächlich „Freaks“ die Technik eingesetzt und getestet haben, scheint sich JSF mittlerweile als allgemein akzeptierter Standard in der UI-Entwicklung zu etablieren. Eine Spezifikation bzw. eine Implementierung allein ist jedoch nicht lebensfähig. Es bedarf der breiten Unterstützung seitens der Industrie, die ergänzende Produkte und Werkzeuge liefert. Welche Unterstützung vonseiten der Tool-Hersteller bereits heute gegeben ist, zeigen wir in unserem Artikel.

JavaServer Faces (JSF) wurde im Rahmen des Java Community Process entwickelt. Dabei wurde ein Expertengremium aufgestellt, das im Falle von JSF über drei Jahre an der Spezifikation gearbeitet hatte. Es setzte sich beinahe aus sämtlichen namhaften Herstellern zusammen (u.a. IBM, Borland, HP, BEA, Oracle). Dies war auch schon früh ein Indiz dafür, dass JSF sicherlich stark von den Herstellern in ihren Produkten unterstützt werden würde. Nach heutigem Stand hat sich dies auch bewährt. Alle großen Anbieter (natürlich auch viele kleinere) haben eine entsprechende JSF-Unterstützung in ihren IDEs vorzuweisen. Im Folgenden stellen wir daher verschiedene Umgebungen vor und zeigen die Funktionalitäten auf. Natürlich ist die Auswahl der Werkzeuge, die hier vorgestellt wird, nicht als vollständig anzusehen. Wichtig ist jedoch zu erkennen, welche Formen einer JSF-Unterstützung möglich sind und wie einem Anwendungsentwickler hier eine Hilfestellung an die Hand gegeben werden kann.

Sun Java Studio Creator

Der Java Studio Creator von Sun war eine der ersten IDEs, die eine umfangreiche JSF-Unterstützung anboten. Dies ist auch nicht weiter verwunderlich, da Sun quasi direkt an der Informationsquelle sitzt. Der Studio Creator ist fast ausschließlich für eine JSF-Entwicklung ausgelegt. Es bietet Unterstützung für die Seitenerstellung, das

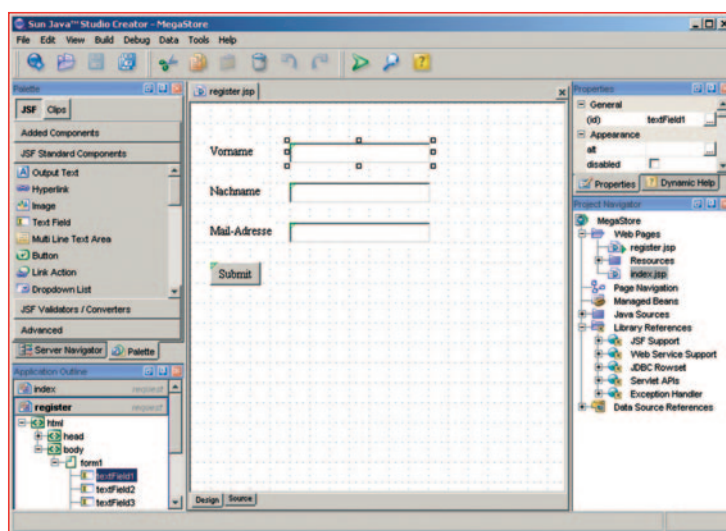
Bean Management und Eigenschaftsdia-
logue für sämtliche JSF-Komponenten sowie eine grafische Darstellung der Seitennavigation. Auch preislich liegt der Studio Creator mit knapp 100 US-Dollar gut im Rennen.

Das Erzeugen der JSF-Seiten geschieht mittels Drag & Drop. Aus der Komponentenpalette können die entsprechenden UI-Komponenten ausgewählt werden und mit der Maus entsprechend auf der Seite positioniert werden. Standardmäßig bietet der Studio Creator ein GridLayout an. Dieses arbeitet über Style-Angaben, sodass die Elemente pixelgenau auf einer Seite angeordnet werden können. Wem dies nicht gefällt, kann auf FlowLayout umstellen,

bei dem auf eine Pixelangabe der Elemente verzichtet wird. Hier kann dann klassisch z.B. mit Tabellen ein Aufbau des Seitenlayouts erreicht werden.

Es ist auch möglich, neben den UI-Komponenten verschiedene Validatoren und Konverter aus der Palette auszuwählen und mit der Maus auf die Seite zu integrieren. Auch hier existieren Eigenschaftsfenster, die eine Eingabe der möglichen Parameter abbilden. Allerdings ist eine Bearbeitung der Seiten in der Source-Ansicht nicht ohne Probleme möglich. Wird beispielsweise ein Textfeld aus dem Quelltext einer Seite gelöscht, ist es nach einem Wechseln in die Design-Ansicht plötzlich wieder sichtbar. Hier empfiehlt sich somit,

Abb. 1: Seitenbearbeitung im Sun Java Studio Creator

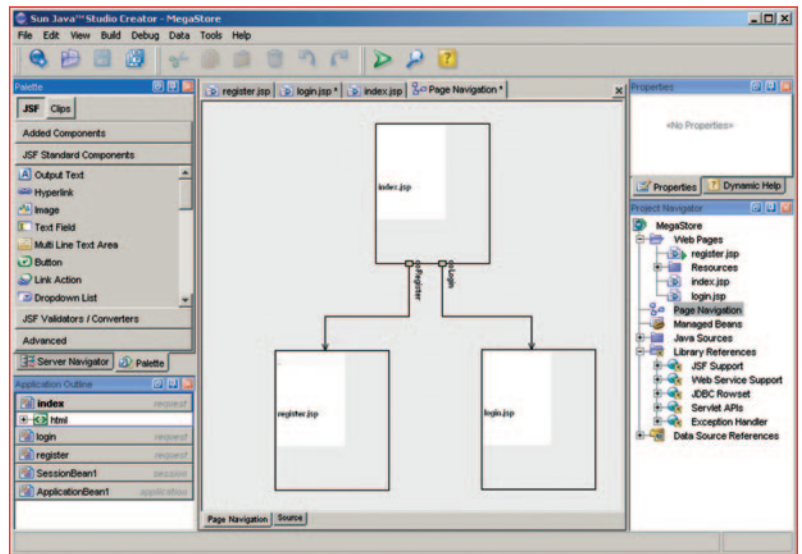


größere Änderungen ausschließlich im Design-Modus vorzunehmen oder Elemente aus dem Application Outline Panel zu entfernen. Doch gerade bei komplexeren User Interfaces müssen Änderungen direkt auf Source-Ebene vorgenommen werden.

Die Erstellung der Navigationsregeln und -fälle kann im Studio Creator ebenfalls ohne manuelle Pflege der Faces-Konfigurationsdatei durchgeführt werden. Er erkennt automatisch, welche JSF-Seiten vorhanden sind. In einer speziellen Ansicht für die Seitennavigation werden die vorhandenen Seiten als Symbole dargestellt und können dann per Drag & Drop miteinander verbunden werden. Bei gedrückter SHIFT-Taste lässt sich die Position der Symbole für die einzelnen Seiten entsprechend verschieben.

Bei JSF müssen sowohl Bean-Klassen erstellt als auch Eintragungen in der Faces-Konfigurationsdatei hinterlegt werden. Beides wird in einem Arbeitsschritt vom Studio Creator übernommen. Nach Angabe des Bean-Namens und der Klasse wird die Bean-Klasse erstellt und die Bean

Abb. 2:
Studio Creator - Darstellung der Navigation



als Managed Bean in der Konfigurationsdatei eingetragen. Natürlich bietet der Studio Creator die gewohnten Wizards zum Erstellen von *Getter*- und *Setter*-Methoden von Eigenschaften ebenfalls an. Die Zuweisung einzelner Eigenschaften von

Managed Beans (Value Binding) geschieht ebenfalls dialoggestützt. So kann zum Beispiel eine Eigenschaft einem Textfeld zugeordnet werden, ohne dass ein Eingriff in die Source-Ansicht der JSF-Seite notwendig ist.

Anzeige

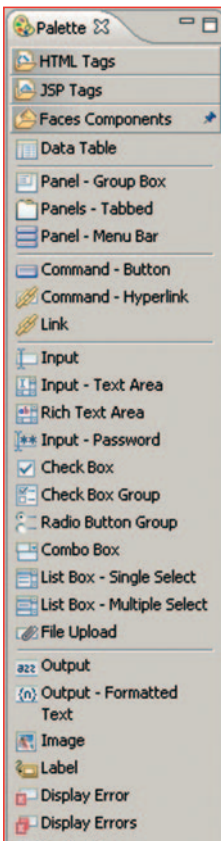


Abb. 3: Rational Application Developer – Komponentenpalette

Die Entwicklung insgesamt ist stark an eine JSF-Entwicklung gekoppelt. Hat man sich jedoch erst einmal an die Dialoge für JSF-Eigenschaften gewöhnt, ist durchaus sehr komfortables Arbeiten möglich. Dass die gesamte Arbeit an JSF-Komponenten durch entsprechende Oberflächen gekapselt ist, hat jedoch auch Nachteile. So ist der generierte Code nicht immer optimal. Es wird beispielsweise standardmäßig zu jeder JSF-Seite eine zusätzliche Page Bean erzeugt, in der die UI-Komponenten direkt ansprechbar sind (Component Binding). Dies kann zwar durchaus nützlich sein, bei größeren Anwendungen jedoch verliert man hier schnell den Überblick.

Insgesamt weist der Studio Creator jedoch genau das auf, was man von einer JSF IDE erwartet, und zwar zahlreiche Wizards, eine dialogunterstützte Entwicklung sowie ein grafischer Editor für die Seitenerstellung.

IBM Rational Application Developer

IBM hat bereits sehr früh eine entsprechende JSF-Unterstützung angeboten. Bereits im Websphere Application Developer

(WSAD) gab es die Möglichkeit, über einen grafischen Editor JSF-Seiten zu gestalten. Aktuell ist die JSF-Unterstützung im Rational Application Developer 6 weiter überarbeitet worden.

IBM hat zusätzlich zu den Standardkomponenten weitere Komponenten entwickelt, die auch im grafischen Editor verwendet werden können (z.B. eine Datei-Upload-Komponente, einen Rich-Text-Editor usw.). Die JSF-Implementierung von IBM ist jedoch nicht die Referenzimplementierung, sondern eine IBM-eigene. Diese ist zwar standardkonform, jedoch existieren IBM-spezifische Erweiterungen. Diese sind zum Teil durchaus nützlich, man macht sich bei Verwendung dieser Funktionen jedoch von der IBM-Implementierung abhängig.

Der grafische Editor ist bereits sehr ausgereift. Es können wieder die einzelnen Komponenten aus einer einzigen Palette ausgewählt und auf der JSF-Seite positioniert werden. Es wird zusätzlich eine größere Anzahl an weiteren Komponenten angeboten (Abb. 3). Für die Seitenansicht stehen insgesamt drei Ansichten zur Verfügung: der Design-Modus, in dem eine grafische Seitenerstellung per Drag & Drop möglich ist, die Source- und zusätzlich noch eine Preview-Ansicht. Die Synchronisierung zwischen den Sichten verlief in den Tests problemlos. Änderungen in der Source-Ansicht waren sofort in der Design-Ansicht und im Preview vorhanden.

Auch im Rational Application Developer (RAD) existiert die Möglichkeit, die

JSF-Navigation grafisch abzubilden und Navigationspfade mittels Drag & Drop festzulegen. Dies wird durch den Web Diagram Editor übernommen. Zusätzlich existiert ein Web Site Designer, der die Navigationspfade einer kompletten Webseite grafisch darstellt. Es können somit auch Navigationen auf externe Seiten visualisiert werden. Man kann sich den Web Site Designer auch als Sitemap vorstellen.

Für den Einsteiger ist die Arbeit mit dem RAD sehr komfortabel. Die IDE übernimmt bereits viele Arbeiten. So werden u.a. beim Anlegen von Managed Beans automatisch Aktionsmethoden als Stubs erzeugt oder beim Erzeugen neuer JSF-Seiten entsprechende Page-Bean-Klassen mit angelegt. Für den erfahrenen JSF-Entwickler, der z.B. bislang mit Eclipse „pur“ gearbeitet hat, ist es anfangs sehr verwirrend, wenn aufgrund weniger Aktionen bereits das Projekt um extrem viel generierten Code erweitert wird. Dennoch weiß man den Komfort nach einer gewissen Einarbeitungszeit durchaus zu schätzen.

Oracle JDeveloper

Eine gute JSF-Unterstützung ist bei Oracle mit dem JDeveloper 10g (10.1.3) gegeben. Die Angaben in diesem Artikel beruhen jedoch noch auf der Developer Preview, sodass sich bis zur finalen Version kleinere Unterschiede ergeben können. Zwar konnte JSF bereits in der Version 10.1.2 in den JDeveloper integriert werden, jedoch war hierbei noch so gut wie keine Entwicklungsunterstützung für JSF

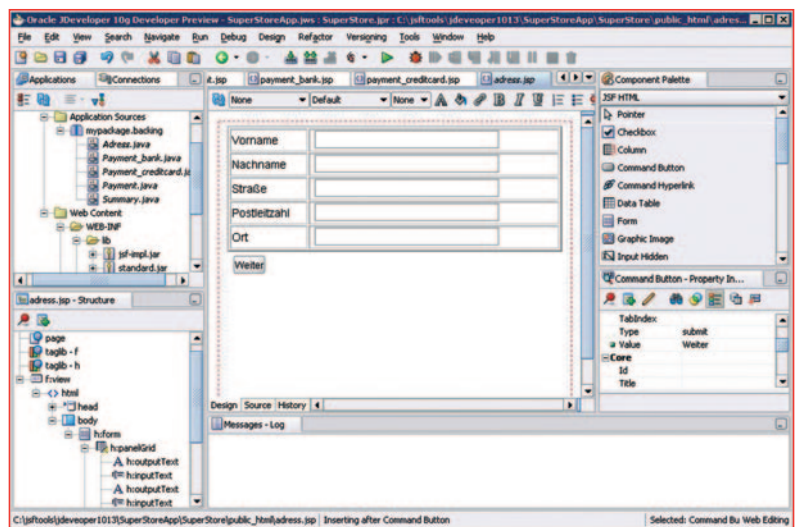


Abb. 4: Seitenbearbeitung im JDeveloper

vorhanden. Zusätzlich zur reinen Unterstützung von JSF bietet Oracle weitere UI-Komponenten mit an. Diese sind unter ADF Faces-Komponenten zusammengefasst.

Oracle liefert den JDeveloper schon seit einiger Zeit mit dem Oracle-eigenen Framework ADF (Application Development Framework) aus. Dieses bietet Unterstützung bei der Softwareentwicklung generell, angefangen von Standard-Applikations-Frameworks, die vollständig integriert sind und automatisch entsprechenden Sourcecode generieren, bis hin zu erprobten Design-Patterns, die im Framework umgesetzt wurden. Mit ADF Faces wurde eine Erweiterung aufgebaut, mit der zusätzliche Komponenten für JSF angeboten werden können.

Man merkt deutlich, dass Oracle an der JSF-Spezifikation mitgewirkt hat. Es existieren viele Wizards, die einem beim Aufsetzen eines JSF-Projektes stark unterstützen. So kann mittels weniger Mausklicks ein komplettes Projekt inklusive Verzeichnisstruktur, Konfigurationsdateien und Bibliotheken getrimmt auf JSF aufgesetzt werden. Sehr schön ist auch, dass zunächst auf Oracle-spezifische Erweiterungen verzichtet und lediglich Standard-JSF eingebunden wird.

Das Anlegen einer neuen JSF-Seite geschieht wiederum über einen entsprechenden Wizard. Anschließend kann die Seite im grafischen Editor per Drag & Drop bearbeitet werden. Der Editor liefert ebenfalls weitgehend JSF-Standard-konformen Quellcode. Positiv zu bemerken ist auch, dass der generierte Code durchweg strukturiert und übersichtlich ist. Die ADF Faces-Komponenten waren in der Developer Preview noch nicht eingebunden, dies wird allerdings bis zur finalen Version erfolgen.

Natürlich existiert auch eine grafische Darstellung der Navigationsregeln. Zusätzlich können neben der grafischen Bearbeitung auch die Navigationsregeln und -fälle dialoggestützt eingegeben werden. Die Werte für die einzelnen Felder werden dabei meist als Auswahlliste vorgeschlagen, sodass mit dieser Lösung sehr gut gearbeitet werden kann.

Insgesamt macht der JDeveloper bereits in der Developer Preview einen sehr

guten Eindruck. Man darf gespannt sein, wie sich die finale Version darstellt.

Borland JBuilder

Zugegebenermaßen auf den ersten Blick etwas enttäuschend verlief der JSF-Test mit dem JBuilder 2005. Sicherlich hat JBuilder seine eigene große Fangemeinde, auch ist die IDE ohne Frage eine sehr mächtige und sehr leistungsfähige Entwicklungsplattform. Im Bezug auf JSF weist JBuilder jedoch einige Defizite auf. JBuilder bietet beim Anlegen eines Projektes die Möglichkeit, JSF-spezifische Konfigurationen automatisch vorzunehmen. Dies funktioniert auch ohne Probleme. Auch können JSF-Seiten per Wizard angelegt und gleich-

zeitig Managed Beans erzeugt werden. Für die Seitenbearbeitung jedoch existiert kein grafischer WYSIWYG-Editor, es ist dagegen eine textuelle Bearbeitung der JSF-Seite notwendig. Dabei gibt es durchaus eine entsprechende Unterstützung, indem per Mausklick ein JSF Tag ausgewählt und dieses dann im Textmodus auf die JSF-Seite platziert werden kann. Dies funktioniert auch sehr elegant. Wählt man z.B. einen Command Button aus, den man auf einer Seite platzieren möchte, muss dieser zwangsläufig vom Form Tag umgeben werden. Dies übernimmt der Editor. Wählt man somit eine Stelle aus, die bereits innerhalb eines Formulars liegt, wird lediglich das Command Button Tag eingesetzt.

Anzeige

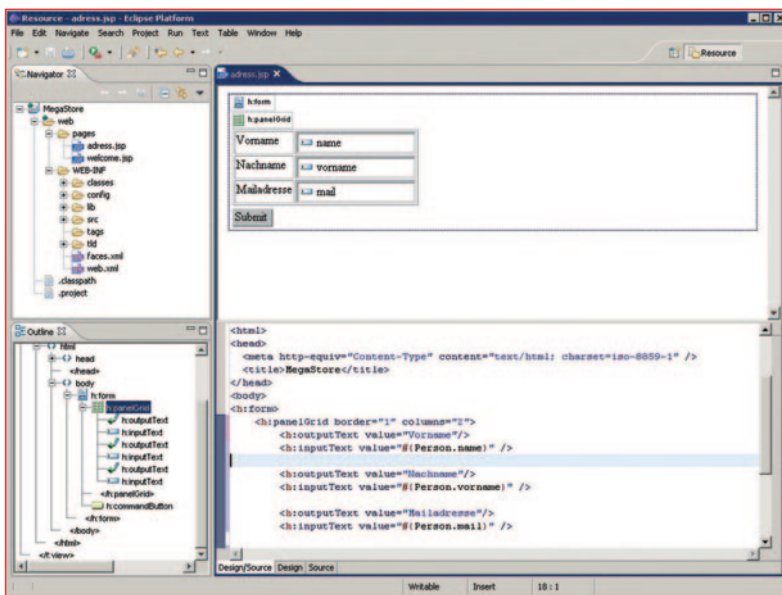


Abb. 5: Split-Pane-Darstellung in NitroX

Liegt die Platzierung des Buttons außerhalb eines Formulars, so wird automatisch das Formular mit angelegt.

Auch beim Thema Navigationsregeln und -fälle hat Borland einen etwas anderen Weg gewählt als die anderen Anbieter. Die Bearbeitung der Navigationsregeln erfolgt an der Seite selbst. Zudem kann über den Editor der Faces-Konfigurationsdatei zusätzlich eine Bearbeitung der Navigationsregeln erfolgen.

Dass man mit dem JBuilder eine mächtige und umfangreiche Entwicklungsumgebung an der Hand hat, steht außer Frage. JBuilder ist nicht zu unrecht eine der beliebtesten IDEs auf dem Markt. In Bezug auf JSF ist die Funktionalität jedoch noch nicht ausgereift genug. Hier wäre es wünschenswert, wenn Borland in künftigen Versionen nochmals nachlegen könnte.

Eclipse-Plug-ins und weitere Tools

Im Eclipse-Umfeld existieren bereits zahlreiche Plug-ins, die eine JSF-Unterstützung bieten. Allerdings sind diese vom Funktionsumfang nicht mit den kommerziellen IDEs der großen Hersteller zu vergleichen. Dennoch bieten sie nützliche Funktionen und Unterstützung an. Im Folgenden werden einige Werkzeuge kurz vorgestellt.

Amateras

Amateras ist ein Open-Source-Projekt, das unter der Apache Software License bei

Sourceforge zu finden ist. Innerhalb des Amateras-Projektes gibt es zahlreiche Unterprojekte. Eines davon, FacesIDE [10], bietet gute Unterstützung für JavaServer Faces. Es wird als Plug-in in Eclipse installiert.

Der Amateras-JSP-Editor bietet Syntax Highlighting, das auch die entsprechenden JSF Tags erkennt. Über eine Palette können JSF-Komponenten in die Seite eingebaut werden. Dies erfolgt jedoch nicht mittels einer grafischen Darstellung, sondern es wird z.B. beim Klick auf ein Eingabefeld in der Palette ein `<h:inputText>` Tag in der Source-Ansicht erzeugt. Die Preview-Ansicht funktioniert nur für manche Komponenten und auch nur dann, wenn mit der MyFaces-Implementierung gearbeitet wird.

Es ist allerdings anzumerken, dass die FacesIDE in der Version 0.1.7 vorliegt, daher wird an dieser Stelle in naher Zukunft sicherlich noch einiges passieren. Da die Dokumentation sehr spärlich ist (beziehungsweise überhaupt nicht vorhanden), sei an dieser Stelle noch darauf hingewiesen, dass Amateras das Vorhandensein des Graphical Editor Framework-(GEF)-Plug-ins in Eclipse voraussetzt.

M7 NitroX for JSF

NitroX for JSF ist ein weiteres Plug-in für Eclipse. Es ist zwar kommerziell, jedoch mit Preisen ab knapp 500 US-Dollar noch im günstigeren Bereich angesiedelt. Ni-

troX bietet in Ergänzung zum Standard Package Explorer einen eigenen Application Explorer. Das Erstellen der Verzeichnisstruktur inklusive des Einbindens der JSF-Bibliotheken erfolgt durch einen Wizard. Auch für die Seitenbearbeitung existiert ein grafischer Editor. Interessant ist hier der Ansatz, mittels einer Split-Pane-Darstellung die Design- und die Source-Ansicht gleichzeitig zu sehen. Dies hat in den Tests auch sehr gut funktioniert, beide Ansichten waren immer synchron. Das Prinzip mit der Split-Pane-Darstellung ist durchgängig in der IDE umgesetzt. So können auch Navigationsregeln oder Managed Beans grafisch aufgebaut werden, während man parallel die Auswirkungen im Source der Faces-Konfigurationsdatei verfolgen kann. Schade ist jedoch, dass ein Drag & Drop nicht umgesetzt wurde, so dass zwar eine grafische Unterstützung gegeben ist, jedoch ein Verschieben (im grafischen Editor, bei den Navigationsregeln etc.) mit der Maus (noch?) nicht unterstützt wird.

Exadel Studio

Exadel hat bereits sehr frühzeitig eine JSF-Unterstützung in der IDE angeboten. Genau genommen handelt es sich dabei auch wieder um Plug-ins, die auf Eclipse aufsetzen. Das Exadel Studio 2.5.2 gibt es in zwei Versionen: Die Basisversion (Exadel Studio) ist kostenfrei zu beziehen, der Preis für die Exadel Studio Pro-Version liegt bei knapp 100 US-Dollar. Die Unterscheidung hinsichtlich JSF ist jedoch nicht gravierend, sodass bereits mit der Standardversion eine sehr gute Unterstützung von JSF gegeben ist.

Es gibt bereits beim Anlegen eines Projektes verschiedene Wizards, die eine entsprechende Infrastruktur für JSF-Projekte aufsetzen (Verzeichnisse, Konfigurationsdateien usw.). Schön ist, dass beim Anlegen eines Projektes die Auswahl besteht, mit welcher Implementierung das Projekt umgesetzt werden soll (Referenzimplementierung oder MyFaces). Es existieren auch wieder Oberflächen, die eine Arbeit mit der Faces-Konfigurationsdatei vereinfachen, also z.B. die Navigation grafisch darstellen oder Managed Beans verwalten. Ein grafischer Editor für das Bearbeiten der JSF-Seiten mittels Drag & Drop

fehlt jedoch. Insgesamt betrachtet wirkt das Exadel Studio sehr durchdacht und bietet einem Entwickler zahlreiche Unterstützung bei der Arbeit mit JSF-Anwendungen.

War das alles?

Nein, natürlich nicht! Es gibt noch weitere kommerzielle und auch freie Tools, die das Arbeiten mit JSF unterstützen. Kurz erwähnt sei noch die Faces Console von James Holmes. Das Werkzeug ist eine Oberfläche für die Faces-Konfigurationsdatei. Technisch ist es als Swing-Anwendung aufgebaut, kann aber auch als Plug-in in Eclipse integriert werden. Das Tool ist kostenfrei, jedoch kein Open Source.

Ein weiteres kommerzielles Plug-in ist MyEclipse von Genuitec. Auch hier besteht die Möglichkeit, Bean Management, Seitennavigation und allgemeine Konfigurationseinstellungen der Faces-Konfigurationsdatei oberflächengestützt vornehmen zu können. Leider fehlt auch hier ein WYSIWYG-Editor.

Diese Auflistung ließe sich noch lange fortsetzen. Die gezeigten Beispiele sollen zumindest helfen, Tools für JSF miteinander vergleichen zu können und aufzeigen, was heutige Tools an Unterstützung für JSF bereits bieten.

Fazit

Als sehr erfreulich aus JSF-Sicht ist zu nächst zu vermerken, dass insgesamt die

JSF-Unterstützung in fast allen IDEs und Werkzeugen gegeben ist. Wenn auch zum Teil noch nicht komplett ausgereift, ist die Verwendung von JSF in den IDEs deutlich komfortabler als pur in einem Editor mit JSF zu arbeiten. In fast allen IDEs kann mithilfe der entsprechenden Unterstützung die Entwicklung einer JSF-Anwendung gut bis sehr gut durchgeführt werden. Der Fortschritt der letzten Monate lässt jedoch erwarten, dass sich in diesem Bereich mit den nächsten Versionen der IDEs noch weitere Verbesserungen ergeben. Für welche Entwicklungsumgebung und welches Tool man sich letztlich entscheidet, hängt daher nicht unbedingt von den hier vorgestellten Eigenschaften ab. Vielmehr ist es entscheidend, ob mit der Gesamtheit der Funktionalität eines Werkzeuges eine verbesserte und beschleunigte Entwicklung möglich ist.

Dennoch soll abschließend noch einmal darauf hingewiesen werden, dass zwischen einer Entwicklungsumgebung, einer JSF-Implementierung und einer JSF-Komponentenbibliotheken unterschieden werden muss. So kann man sich hier auch das Beste aus allen Lagern zusammenbauen. Der Autor hat beispielsweise für einen Prototyp mit Eclipse gearbeitet, die Sun-Referenzimplementierung eingesetzt und exemplarisch anschließend MyFaces- sowie ADF Faces-Komponenten integriert.

Andy Bosch ist selbstständiger IT-Berater und Projektleiter. Er beschäftigt sich überwiegend mit der Konzeption und Realisierung von Webanwendungen, seit einiger Zeit fast ausschließlich im JSF-Umfeld. Zudem ist er Betreiber der Plattform www.jsf-forum.de.

■ Links & Literatur

- [1] Java Studio Creator: developers.sun.com/prodtech/javatools/jscreator/
- [2] Christian Knothe: Corporate Creator. Konzeptwelt des Sun Java Studio Creator, in *Java Magazin* 5.2005
- [3] Rational Application Developer: www-130.ibm.com/developerworks/rational/
- [4] Michael Müller: Das Rad neu erfunden? IBMs J2EE-Entwicklungsumgebung Rational Application Developer in neuem Gewand, in *Java Magazin* 5.2005
- [5] JDeveloper 10g: www.oracle.com/technology/products/jdev/
- [6] Stefan Scheidt, Thorsten Winterberg: Java à la 4GL. Tools plus Frameworks – das Konzept des Oracle JDeveloper 10g, in *Java Magazin* 5.2005
- [7] JBuilder: www.borland.de/jbuilder/
- [8] Dirk Frischalowski: 2005 und mehr neue Features im neuen JBuilder, in *Java Magazin* 1.2005
- [9] Projekt Amateras: sourceforge.jp/projects/amateras/
- [10] Matthias Weßendorf: FacesIDE. JavaServer Faces-Anwendungen mit Eclipse erstellen, in *Eclipse Magazin* Vol. 4
- [11] NitroX IDE: www.m7.com
- [12] JSF Studio: www.exadel.com
- [13] Faces Console: www.jamesholmes.com/JavaServerFaces/
- [14] MyEclipse Enterprise Workbench: www.myeclipseide.com
- [15] Sven Hankemeier: Praktischer Einstieg in die Entwicklung mit der MyEclipse Enterprise Workbench, in *Eclipse Magazin* Vol. 5 (erscheint im November 2005)

Anzeige