Ausgabe 01/2023

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

D: 4,90 EUR | CH: 9,80 CHF | A: 5,60 EUR | Benelux: 5,80 EUR

4 191978 304903 01

Java aktuell

iJUG Verbund
www.ijug.eu

Enterprise
Jakarta Server Faces 4.0, Textblöcke

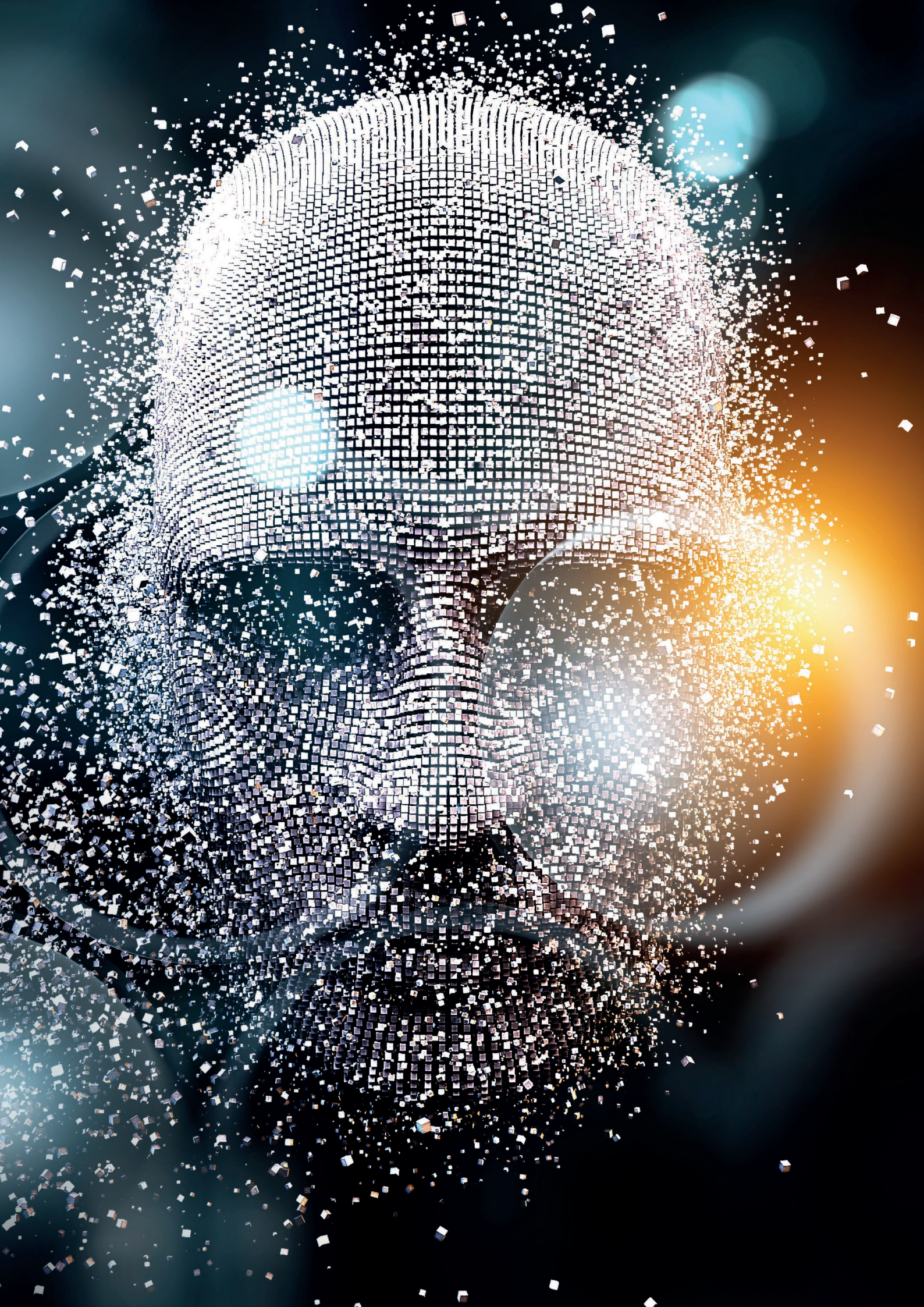
Moderne Frontends
mit Spring Boot, Thymeleaf und HTMX

Ethik
Ethisches Handeln in der Softwareentwicklung

ENTERPRISE

Totgesagte leben länger: Jakarta Faces 4.0 ist erschienen

Bernd Müller, Ostfalia



JavaServer Faces sind ganz bestimmt nicht mehr en vogue, wenn es um die Oberflächenerstellung geht, viele gehypte JavaScript-Frameworks dagegen schon. Die Halbwertszeit der JavaScript-Frameworks ist jedoch zum Teil recht niedrig, die von JavaServer Faces sehr hoch. Sie entwickeln sich immer noch weiter, sodass ein neues Release erschienen ist. Mit dem Umzug von Java EE zur Eclipse Foundation wurden die JavaServer Faces umbenannt und heißen nun Jakarta Faces. Im Rahmen von Jakarta EE 10 wurden Jakarta Faces 4.0 veröffentlicht, die wir in diesem Artikel kurz vorstellen.

Ein kleiner Rückblick

JavaServer Faces 1.0 erblickte im März 2004 das Licht der Welt. Mit der im Mai 2004 nachgeschobenen Version 1.1 waren JavaServer Faces dann auch tatsächlich in Produktivsystemen verwendbar. Im Bereich der Software-Entwicklung sind JavaServer Faces damit bereits in einem recht hohen Alter. Die letzte unter dem Dach von Java EE erschienene Version war 2.3, veröffentlicht im März 2017. Die Überführung der umfangreichen Code-Basis von Java EE zur Eclipse Foundation und vor allem die Klärung rechtlicher Fragen zog sich recht lange hin. Im Rahmen von Jakarta EE 8 wurde Jakarta Server Faces 3.0 veröffentlicht. Wie bei allen anderen EE-Spezifikationen auch, war das API identisch zur Vorgängerversion. Jakarta EE 9 vollzog die Änderung der Package-Namen, wobei der Präfix javax zu jakarta wurde. Jakarta EE 10 war das erste Release, das tatsächlich die Gelegenheit hatte, Neuerungen zu realisieren. JavaServer Faces wurden zu Jakarta Faces und die Version 4.0 deutet an, dass es wirklich Neuerungen gab. Im Folgenden werden wir meist von Faces sprechen, unabhängig von der Version.

Die Faces-Spezifikationen und viele andere Spezifikationen der EE-Familie beziehungsweise die Personen, die maßgeblich an der Erstellung der Spezifikation beteiligt waren, haben aus den Fehlern der frühen EJB-Spezifikationen gelernt und spezifizierten Framework-Eigenschaften nur, wenn die zugrunde liegenden Konzepte und Ideen bereits implementiert und auch erfolgreich eingesetzt wurden. Bei Faces sind hier vor allem MyFaces Codi, DeltaSpike *[1]* und OmniFaces *[2]* zu nennen, die die Entwicklung maßgeblich, auch personell, begleiteten. Durch das recht hohe Alter ist Faces ein sehr ausgegorenes und hochentwickeltes Framework, das wenig Wünsche offenlässt. Da ein grundlegendes Paradigma von Faces die Unterstützung von HTML ist, ist dieser Aspekt tatsächlich mehr oder weniger abgeschlossen. Für Features, die über Standard-HTML hinausgehen, sieht Faces die Verwendung von Komponentenbibliotheken vor, von denen PrimeFaces *[3]* aktuell sicher die bekannteste ist.

Die mit Faces 4.0 bezüglich HTML realisierten Neuerungen sind daher eher kosmetischer Natur. Auch die anderen neuen Features können hauptsächlich unter der Überschrift vereinfachte Entwicklung subsumiert werden. Faces 4.0 bricht aber mit einer sehr alten Tradition von Enterprise-Java, die der Rückwärtskompatibilität. Einige Framework-Eigenschaften, die schon länger nicht mehr wirklich genutzt werden, wurden tatsächlich entfernt.

Bei der Vorstellung der Änderungen von Faces 4.0 beginnen wir mit den Features, die in Zusammenhang mit HTML stehen. Danach gehen wir auf Neuerungen ein, die die Entwicklung mit Faces vereinfachen, um am Ende entfernte Features zu erwähnen.

Neuerungen mit HTML-Bezug

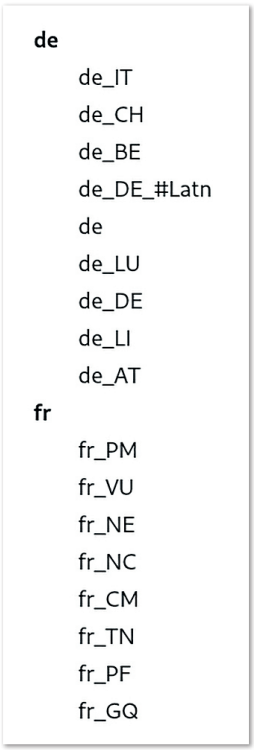
Da Faces ein Server-seitig HTML-generierendes Framework ist, sind die Neuerungen mit Bezug auf HTML von besonderem Interesse. Als Zielsprache wird schon seit mehreren Versionen HTML5 erzeugt. Zunächst konkurrierend arbeiten das W3C und die WHATWG (Web Hypertext Application Technology Working Group) mittlerweile gemeinsam an der Weiterentwicklung von HTML. Während jedoch das W3C in der Vergangenheit jeweils stabile Momentaufnahmen des Standards als HTML5, 5.1, und 5.2 *[4]* spezifizierte, arbeitet die WHATWG an einem sogenannten Living Standard *[5]* mit Namen HTML, das heißt ohne Versionsnummer. Mittlerweile verweisen die W3C-Seiten auf den Standard der WHATWG. Unabhängig von diesen beiden Varianten kann man HTML durchaus bescheinigen, dass die Weiterentwicklung im Fluss war und ist. Ein Framework wie Faces hat es damit recht schwer, up to date zu sein. Sehr vorausschauend wurden daher mit JSF 2.2 die sogenannten Pass-Through-Attribute und Pass-Through-Elemente definiert, die es ermöglichen, neue HTML-Tag-Attribute an die entsprechenden Faces-Tags unverändert durchzureichen (pass through). Die ersten der nun vorzustellenden Änderungen sind unter diesem Blickwinkel einzuordnen. Sie erleichtern dem geübten HTML-Nutzer die Arbeit, sind aber nicht zwingend notwendig. Um etwa im generierten HTML eine einfache Eingabe für eine E-Mail-Adresse zu erzeugen (`<input type="email">`), wurde bisher `<h:inputText pt:type="email" />` verwendet, wobei pt der entsprechende XML-Namensraum für Pass-Through-Attribute ist. Mit Faces 4.0 kann man nun `<h:inputText type="email" />` schreiben. Keine ganz grundlegende Verbesserung, aber einfacher zu lesen und zu schreiben. Werden Eingabetypen verwendet, für die dedizierte JSF-Tags existieren, wird im Development-Modus eine entsprechende Warnung mit Verweis auf das zu verwendende JSF-Tag ausgeben. Wird etwa der Eingabetyp password verwendet, so verweist die Warnung auf das JSF-Tag `<h:inputSecret>`, das an seiner Stelle zu nutzen ist.

```
<h:selectOneMenu>
  <f:selectItemGroups value="#{locales.localesFor('de', 'es', 'fr')}"
        var="languageLocales" itemLabel="#{languageLocales.key}">
     <f:selectItems value="#{languageLocales.value}"
        var="locale" itemLabel="#{locale}" />
  </f:selectItemGroups>
</h:selectOneMenu>
```

Listing 1

Ähnlich verhält es sich mit dem Faces-Tag `<h:inputFile>`, das gleich zwei neue Attribute bekommen hat: `multiple` und `accept`. Wird das boolesche Attribut `multiple` auf `true` gesetzt, können im Browser mehrere Dateien selektiert werden. Das Attribut `accept` erwartet eine durch Kommas getrennte Liste von Mime-Typen, also etwa `image/png` oder `application/pdf`. Vom Browser werden dann nur Dateien dieser Typen zur Auswahl angeboten.

Faces enthält eine Reihe von Tags, die zu HTML-Auswahlkomponenten gerendert werden. Die entsprechenden Auswahldaten werden mit `<f:selectItem>` und `<f:selectItems>` und der Klasse `SelectItem` definiert. Das daraus gerenderte HTML besteht aus `<option>`-Elementen. Um diese gruppieren zu können (`<optgroup>`), musste bisher die Klasse `SelectItemGroup` Java-seitig verwendet werden. Eine Gruppierung direkt in der View war nicht möglich. Mit Faces 4.0 wurden die Tags `<f:selectGroup>` und `<f:selectGroups>` eingeführt, die dies ermöglichen. Das *Listing 1* zeigt ein kleines Beispiel zur Verwendung, das die Lokalisierungen der JVM als Auswahldaten anzeigen soll. Die in `<f:selectItemGroups>` als Wert verwendete Methode `localesFor()` erwartet einen Varargs-Parameter von Strings, den Sprach-Codes von Lokalisierungen, und liefert eine Map zurück, die als Schlüssel die übergebenen Sprach-Codes und als Werte eine Liste der Lokalisierungen dieser Sprache hat. Die Iterationsvariable `languageLocales` wird nun verwendet, um im Beispiel über die drei Map-Einträge zu iterieren. Das geschachtelte `<f:selectItems>`-Tag iteriert über die Liste der Lokalisierungen der jeweiligen Sprache. Die *Abbildung 1* zeigt das resultierende Pop-up-Fenster nach Drücken des Auswahlmenüs. Das ebenfalls neue Tag `<f:selectItemGroup>` wird verwendet, wenn nur ein einzelnes `<optgroup>` gewünscht ist.



Die letzte Neuerung in Bezug auf HTML betrifft die Faces-Tags `<h:selectManyCheckbox>` und `<h:selectOneRadio>`. Das gerenderte HTML verwendete eine Tabelle für das Layout. Dies entspricht nicht mehr modernen Anforderungen für das Layout. Mit dem neuen Wert `list` für das Attribut `layout` wird eine `<ul><li>`-Struktur für die Checkboxen beziehungsweise Radio-Buttons verwendet. Die entsprechenden CSS-Definitionen sind zu ergänzen.

Nicht ganz korrekt unter der HTML-Überschrift einsortiert ist das neue Attribut `onerror` des `<f:websocket>`-Tags. Als Wert des Attributs ist eine JavaScript-Funktion anzugeben, die als Event-Handler für Verbindungsfehler des WebSocket dient.

Programmatische Erstellung von Views

Faces-Views werden mit einer Seitenbeschreibungssprache, Facelets genannt, deklarativ erstellt. Die mit JSF 1.0 ursprüngliche Definition mithilfe von JavaServer Pages wurden mit der Version 2.0 als deprecated erklärt. Mit der Version Faces 4.0 wurden nun alle mit JSP in Zusammenhang stehenden Artefakte entfernt. Dies ist eine nicht rückwärtskompatible Änderung, die allerdings zu verschmerzen sein wird, da JSPs nie richtig mit zentralen Faces-Konzepten harmoniert haben. Die programmatische Erstellung von Views war schon immer möglich, da die Elemente der Seitenbeschreibungssprache in Java-Objekte gewandelt werden, die alternativ auch direkt erzeugt werden können. Man benötigte aber eine minimale Faces-View, in der Regel mit `<f:view>` als Wurzelelement. Mit Faces 4.0 ist es nun möglich, eine View komplett mit Java, ganz ohne XHTML-Datei, zu erstellen. Da eine XHTML-Datei aber namensgebend für die View-Id ist, wird dies nun durch die neue `@View`-Annotation realisiert. *Listing 2* zeigt ein Beispiel für diese programmatische View-Definition. Wir verzichten auf eine Erläuterung, da *Listing 3* das Äquivalent in der gängigen View-Syntax zeigt und eine Gegenüberstellung der beiden Alternativen durch den Leser den Java-Code selbsterklärend macht. Abschließend lässt sich sagen, dass *Listing 3* deutlich kürzer, leichter verständlich und – nach Meinung des Autors – dem *Listing 2* vorzuziehen ist. Die programmatische Erstellung einer View ist aber nur ein Angebot. Der Entwickler hat wie immer die letzte Entscheidung.

Neuer CDI-Scope für Browser-Tabs

Mit der neuen CDI-Scope-Annotation @ClientWindowScoped kann die Lebensdauer einer Backing-Bean an ein Browser-Tab gebunden werden. Damit ist es sehr einfach möglich, dieselbe Faces-Seite, beispielsweise die der Kundenstammdaten, in mehreren Tabs für verschiedene Daten zu öffnen.

Neue Methode `getLifecycle()` der Klasse `FacesContext`

Die abstrakte Klasse FacesContext enthält alle für einen Faces-Request vorhandenen Informationen. Ein Zugriff auf den Lebenszyklus eines Faces-Request war bisher nur mit erheblichem Aufwand möglich. Die neue Methode `getLifecycle()` ändert dies und erlaubt damit zum Beispiel das programmatische Hinzufügen oder Entfernen eines Phase-Listener als Einzeiler.

Für alle bisher vorgestellten Neuerungen in Faces 4.0 stellen wir auf GitHub ein Projekt bereit *[6]*, das an kleinen Beispielen das bereits erläuterte noch einmal praktisch vorstellt.

Weitere Neuerungen, Änderungen und Löschungen

Es gibt noch weitere Neuerungen, die aber für die meisten Faces-Entwickler eher weniger relevant sein werden, sodass wir sie nicht explizit aufführen. Für eine komplette Übersicht empfehlen wir What's new in Faces 4.0? *[7]*, erstellt von Bauke Scholtz, besser bekannt unter dem Pseudonym BalusC, und unter diesem Pseudonym auf Stack Overflow der Nutzer mit den meisten Antworten zu Faces-Fragen.

Bei den Änderungen sind vor allem die XML-Namensräume wichtig. Diese waren bisher als URLs mit dem Präfix `http://xmlns.jcp.org/jsf/*` bekannt und wurden nun zu URNs mit dem Präfix `jakarta.faces.*`. Wir haben dies bereits in *Listing 3* verwendet. Weitere Änderungen sind eher kosmetischer Natur und ebenfalls unter *[7]* nachzulesen.

Als nicht rückwärtskompatible Löschungen ist neben der JSP-Löschung vor allem auf das Entfernen der Faces-eigenen Anno-

```
@View("/programmatic-facelet.xhtml")
@ApplicationScoped
public class ProgrammaticFacelet extends Facelet {

     @Override
     public void apply(FacesContext facesContext, UIComponent root) throws IOException {
            if (!facesContext.getAttributes().containsKey(IS_BUILDING_INITIAL_STATE)) {
                  return;
            }
            ComponentBuilder components = new ComponentBuilder(facesContext);
            List<UIComponent> rootChildren = root.getChildren();
            UIOutput output = new UIOutput();
            output.setValue("<html xmlns=\"http://www.w3.org/1999/xhtml\">");
            rootChildren.add(output);
            HtmlBody body = components.create(HtmlBody.COMPONENT_TYPE);
            rootChildren.add(body);
            HtmlForm form = components.create(HtmlForm.COMPONENT_TYPE);
            form.setId("form");
            body.getChildren().add(form);

            HtmlOutputLabel label = components.create(HtmlOutputLabel.COMPONENT_TYPE);
            label.setValue("Enter some text: ");
            form.getChildren().add(label);
            HtmlInputText input = components.create(HtmlInputText.COMPONENT_TYPE);
            input.setId("input");
            form.getChildren().add(input);
            String textEl = "#{backingBean.text}";
            FacesContext fc = FacesContext.getCurrentInstance();
            ValueExpression valueExpression = fc.getApplication().getExpressionFactory()
        .createValueExpression(fc.getELContext(), textEl, String.class);
            input.setValueExpression("value", valueExpression);

            HtmlCommandButton actionButton = components.create(HtmlCommandButton.COMPONENT_TYPE);
            String actionEl = "#{backingBean.action}";
            actionButton.setId("button");
            Class<?>[] parameterTypes = new Class[0];
            MethodExpression methodExpression = fc.getApplication().getExpressionFactory()
        .createMethodExpression(fc.getELContext(), actionEl, String.class, parameterTypes);
            actionButton.setActionExpression(methodExpression);
            actionButton.setValue("Do action");
            form.getChildren().add(actionButton);

            output = new UIOutput();
            output.setValue("</html>");
            rootChildren.add(output);
     }

     private static class ComponentBuilder {

            FacesContext facesContext;

            ComponentBuilder(FacesContext facesContext) {
                  this.facesContext = facesContext;
            }

            <T> T create(String componentType) {
                  return (T) facesContext.getApplication().createComponent(facesContext, componentType, null);
            }
     }

}
```

Listing 2

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="jakarta.faces.html">

     <h:body>
            <h:form id="form">
                  <h:outputLabel value="Enter some text: " />
                  <h:inputText id="input" value="#{backingBean.text}" />
                  <h:commandButton id="button" value="Do action" action="#{backingBean.action}" />
            </h:form>
     </h:body>
</html>
```

Listing 3

tationen für Dependency Injection hinzuweisen. Diese wurden mit JSF 2.0 zu einer Zeit eingeführt, als sich Dependency Injection insgesamt etablierte, aber CDI noch nicht verfügbar beziehungsweise innerhalb von Java EE noch nicht etabliert war. Die seit JSF 2.3 als deprecated gekennzeichneten Annotationen @ManagedBean sowie die Scope-Annotationen, die namensgleich zu den entsprechenden CDI-Annotationen sind, wurden entfernt. Es wurden noch einige wenige weitere Klassen entfernt, die schon seit geraumer Zeit deprecated waren. Aufgrund der geringen Relevanz zählen wir diese aber nicht auf. Faces 4.0 macht durch den Abwurf dieses alten Ballasts einen sehr aufgeräumten Eindruck. Es ist zu hoffen, eher zu erwarten, dass viele Altanwendungen die nun entfernten Klassen schon seit geraumer Zeit nicht mehr verwenden und mit Faces 4.0 problemlos lauffähig sind.

Zusammenfassung

JavaServer Faces, nun Jakarta Faces oder kurz Faces, existieren bereits seit vielen Jahren. Durch das für IT-Systeme hohe Lebensalter kann man Faces mit Fug und Recht als sehr ausgereift bezeichnen. Die neueste Version, Faces 4.0, ist die erste Version unter dem Jakarta-Dach, die mit wirklichen Änderungen aufwartet. Dazu zählen Neuerungen, tatsächliche Änderungen an Bestehendem, aber auch das Entfernen alter Zöpfe. Durch den hohen Reifegrad von Faces sind die Änderungen mehrheitlich als Features zu bewerten, die die Entwicklung vereinfachen. Es gibt ganz einfach keine grundlegenden Feature-Requests für wirklich Neues. Es bleibt abzuwarten, ob in einem der nächsten Releases derartig Neues Einzug in Faces hält und Faces damit konkurrenzfähig bleiben.

Referenzen

[1] DeltaSpike, *https://deltaspike.apache.org/*.
[2] OmniFaces, *https://omnifaces.org/*.
[3] PrimeFaces, *https://www.primefaces.org/*.
[4] W3C HTML 5.2, *https://www.w3.org/TR/2017/REC-html52-20171214/*.
[5] WHATWG HTML, *https://html.spec.whatwg.org/*.
[6] Faces4.0, *https://github.com/BerndMuller/faces4*.
[7] What's new in Faces 4.0?, *https://balusc.omnifaces.org/2021/11/whats-new-in-faces-40.html*.

Bernd Müller
Ostfalia
bernd.mueller@ostfalia.de

Nach seinem Studium der Informatik und der Promotion arbeitete Bernd Müller für die IBM und die HDI Informationssysteme. Er ist Professor, Geschäftsführer, Autor mehrerer Bücher zu den Themen JSF und JPA, sowie Speaker auf nationalen und internationalen Konferenzen. Er engagiert sich im iJUG und speziell in der JUG Ostfalen.

Mitglieder des iJUG

01 BED-Con e.V.
02 Clojure User Group Düsseldorf
03 DOAG e.V.
04 EuregJUG Maas-Rhine
05 JUG Augsburg
06 JUG Berlin-Brandenburg
07 JUG Bremen
08 JUG Bielefeld
09 JUG Bonn
10 JUG Darmstadt
11 JUG Deutschland e.V.
12 JUG Dortmund
13 JUG Düsseldorf rheinjug
14 JUG Erlangen-Nürnberg
15 JUG Freiburg
16 JUG Goldstadt
17 JUG Görlitz
18 JUG Hannover
19 JUG Hessen
20 JUG HH
21 JUG Ingolstadt e.V.
22 JUG Kaiserslautern
23 JUG Karlsruhe
24 JUG Köln
25 Kotlin User Group Düsseldorf
26 JUG Mainz
27 JUG Mannheim
28 JUG München
29 JUG Münster
30 JUG Oberland
31 JUG Ostfalen
32 JUG Paderborn
33 JUG Saxony
34 JUG Stuttgart e.V.
35 JUG Switzerland
36 JSUG
37 Lightweight JUG München
38 SOUG e.V.
39 SUG Deutschland e.V.
40 JUG Thüringen
41 JUG Saarland

iJUG Verbund
www.ijug.eu

Impressum

Java aktuell wird vom Interessenverbund der Java User Groups e.V. (iJUG) (Tempelhofer Weg 64, 12347 Berlin, *www.ijug.eu*) herausgegeben. Es ist das User-Magazin rund um die Programmiersprache Java im Raum Deutschland, Österreich und Schweiz. Es ist unabhängig von Oracle und vertritt weder direkt noch indirekt deren wirtschaftliche Interessen. Vielmehr vertritt es die Interessen der Anwender an den Themen rund um die Java-Produkte, fördert den Wissensaustausch zwischen den Lesern und informiert über neue Produkte und Technologien.

Java aktuell wird verlegt von der DOAG Dienstleistungen GmbH, Tempelhofer Weg 64, 12347 Berlin, Deutschland, gesetzlich vertreten durch den Geschäftsführer Fried Saacke, deren Unternehmensgegenstand Vereinsmanagement, Veranstaltungsorganisation und Publishing ist.

Die DOAG Deutsche ORACLE-Anwendergruppe e.V. hält 100 Prozent der Stammeinlage der DOAG Dienstleistungen GmbH. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. wird gesetzlich durch den Vorstand vertreten; Vorsitzender: Björn Bröhl. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. informiert kompetent über alle Oracle-Themen, setzt sich für die Interessen der Mitglieder ein und führen einen konstruktiv-kritischen Dialog mit Oracle.

Redaktion:
Sitz: DOAG Dienstleistungen GmbH
ViSdP: Fried Saacke
Redaktionsleitung: Lisa Damerow
Kontakt: redaktion@ijug.eu

Redaktionsbeirat:
Andreas Badelt, Melanie Andrisek, Marcus Fihlon, Markus Karg, Manuel Mauky, Bernd Müller, Benjamin Nothdurft, Daniel van Ross, André Sept

Titel, Gestaltung und Satz:
Alexander Kermas,
DOAG Dienstleistungen GmbH

Bildnachweis:
Titel: Bild © vectorpouch
https://freepik.com
S. 10 + 11: Bild © nexusplexus
https://123rf.com
S. 16 + 17: Bild © vectorjuice
https://freepik.com
S. 24: Bild © Harryarts
https://freepik.com
S. 27: Bild © lemonos
https://123rf.com
S. 32 + 33: Bild © StockVector
https://stock.adobe.com
S. 38: Bild © Bro Vector
https://stock.adobe.com

Anzeigen:
DOAG Dienstleistungen GmbH
Kontakt: sponsoring@doag.org

Mediadaten und Preise:
www.doag.org/go/mediadaten

Druck:
WIRmachenDRUCK GmbH
www.wir-machen-druck.de

Inserentenverzeichnis