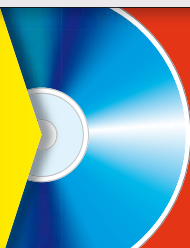


Java™magazin

Java • Architekturen • SOA • Agile

www.javamagazin.de 

Mit CD



Testversionen & more

- CAS Server 3.1.1
- Eclipse Modeling Framework Project (EMF)
- Glassfish V2
- JOSSO 1.6
- Modeling Workflow Engine (MWE)
- wingS 3.1

PLUS weitere Top-Tools

Infos auf S. 35

Specials

- JAX TV: Keynote von der W-JAX 2007



- Buchauszug: Hibernate – Praxisbuch für Entwickler

+ Bonus-CD für Abonnenten: Alle Ausgaben aus 2007!

Single Sign-on

SSO-Systeme auf dem Prüfstand

Test 5

Sonderdruck

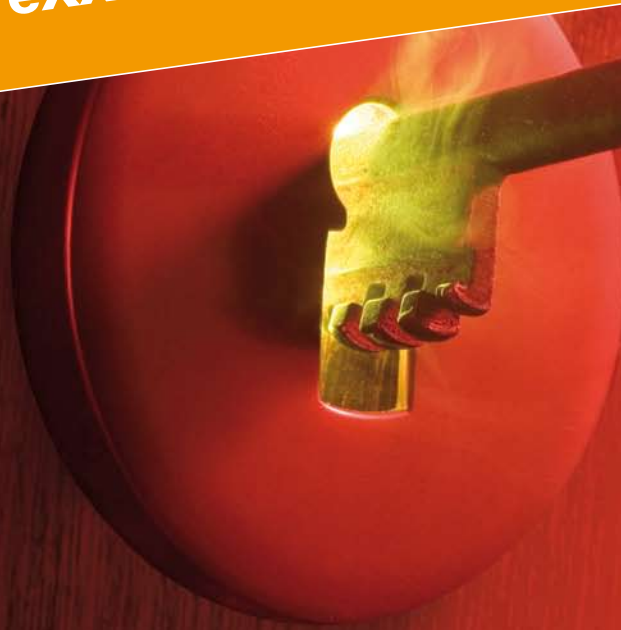
der Firma eXXcellent solutions

→ **Systemarchitektur mit der UML**
Kleine Anpassungen, großer Nutzen

→ **Get your wingS back!**
Framework für Ajax-Anwendungen

→ **Domänenspezifische Sprachen**
Erstellung einer textuellen DSL mit Xtext

→ **Software-Industrialisierung**
Java Banking Framework



Datenträger enthält nur Lehr- oder Infoprogramme

D 45867



4 194586 707500

0 1

Mit beflügelter Leichtigkeit Ajax-Applikationen in Java erstellen

Get your wings back!

VON BENJAMIN SCHMID



Mit der neuesten Version 3.0 rüstet das Open-Source Framework wingS Ajax-Funktionalitäten nach und legt noch ein paar neue Widgets mit dazu. Event- und Komponentenorientierung versprechen auf dem Markt immer mehr Frameworks. Wo liegen also die Vorteile und Stärken von wingS? Wir werfen einen Blick unter die Haube des an Swing anlehenden Web-Frameworks und geben in einer Gegenüberstellung mit Eclipse RAP und Echo2 Orientierungshilfe im Dschungel der RIA-Lösungen.

Es bedurfte eines etwas längeren Anlaufs, bis sich Swing zu dem Vorzeigebeispiel für plattformunabhängig saubere und modulare GUIs entwickelt hat. „Warum nicht eine gute API auch für die Erstellung von Web-Applikationen nutzen?“ dachten sich schon anno 2000 die Entwickler von wingS, als Sie die erste Version ihres Webframeworks mit weitestgehend identischer Schnittstelle ver-

öffentlichten. Inzwischen geht der Trend im Web unter dem Schlagwort RIA weiter Richtung Desktop, womit auch seitenorientierte Programmierparadigmen zunehmend überholt sind. Damit wird die komponentenorientierte Entwicklung von Webapplikationen aktueller denn je: Statt HTML, XML und Javascript-Programmierung sind Java, MVC und Events die bevorzugten Mittel und für wingS von Anfang an kennzeichnende Merkmale.

Das einfache HelloWings-Beispiel in Listing 1 vermittelt einen ersten Ein-

druck in die Anwendungsentwicklung mit wingS und funktioniert in dieser Form unverändert auch mit allen bisherigen Vorgängerversionen. Ausführlichere Einstiegshilfen bieten die offizielle Projekt-Website [1] und Dokumentation sowie unser Artikel in [2]. Mit der Version 3 hat nun auch in wingS das Thema Ajax und RIA massiv Einzug gehalten, allerdings gänzlich ohne den Entwickler mit neuer Komplexität zu belasten. Die technischen Grundlagen hierzu wurden bereits im Vorfeld geschaffen: So wurde das Rendering auf



Quellcode auf CD

isoliert austauschbare Komponenten optimiert, verstärkt CSS eingesetzt und erste RIA-Features wie Drag & Drop oder Tastatur-steuerung in die API mit aufgenommen. Weitere Grundsteine waren durch das im Detail feiner mit Swing abgestimmte Verhalten der Komponenten und Layouts bereits gesetzt. Nun konnte bei unveränderter API eine neue Ajax Engine integriert werden, welche vollständige Seitenauslieferung überflüssig macht und damit einen völlig neuen Bedienkomfort ermöglicht. Zusätzliche Annehmlichkeiten bieten neu hinzugekommene Ajax-Widgets wie z.B. die Suggest-Box, der Date-Picker oder ein In-Place-Editor. Da es diese Widgets allesamt im Original Swing selbst nicht gibt, fanden sie ihren Platz im separaten Addon-Paket wingX.

Besonders bemerkenswert ist die transparente Verwendung der Ajax Technologie bei weiterhin guter Steuerbarkeit auf allen Ebenen. Der Anwendungsentwickler programmiert ausschließlich gegen eine reine Java API und nutzt dabei die altbekannten Funktionen und Design-Konzepte aus Swing. Trotz des hohen Abstraktionsgrades und der innovativen Technologie behält er jedoch die Möglichkeit, weiterhin auf Details Einfluss zu nehmen, wie wir im späteren Teil näher betrachten werden.

Transparentes Ajax

Werfen wir zuerst einen Blick auf die Technik der neu eingeführten inkrementellen Seitenaktualisierungen. Verantwortlich hierfür zeigt sich ein erweiterter *ReloadManager* in der wingS Architektur [5]. Dieser ist dafür zuständig, zu erkennen welche Komponenten sich während der Verarbeitung eines Requests durch die vom Event Dispatcher zugestellten Events und die UI-Logik geändert haben und daher im Browser aktualisiert werden müssen. Konnte bislang immer nur die HTML-Darstellung vollständig neu generiert und ausgeliefert werden, werden nun durch die Ajax-Engine lediglich die reinen Änderungen direkt in die Seite eingearbeitet. Dabei genügt es, wenn sich die Implementierungen der Renderer in der simpelsten Variante einfach weiterhin darauf beschränken, auf Anfrage

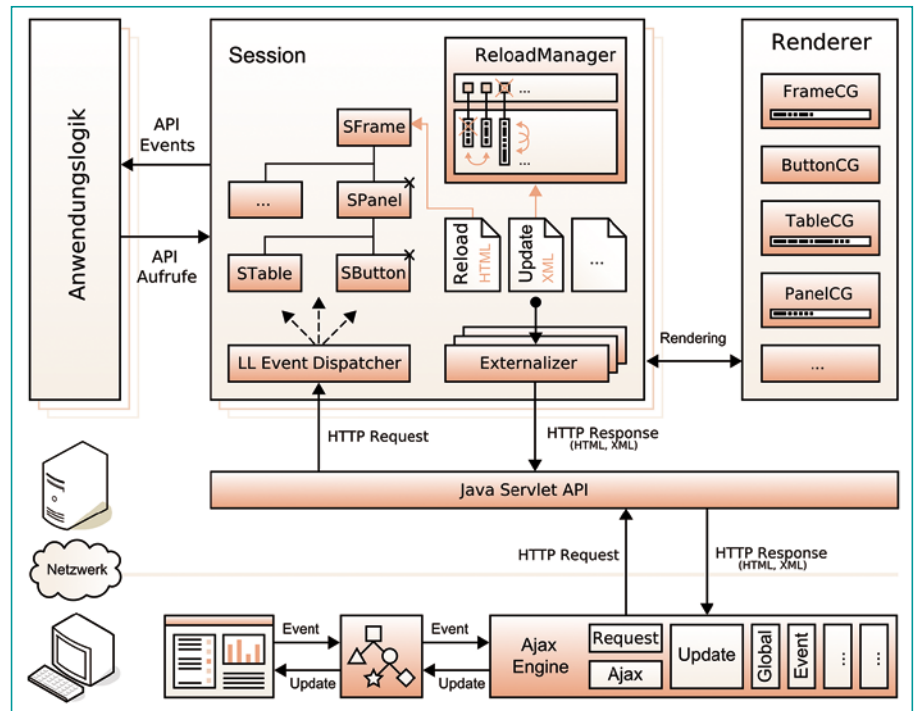


Abb. 1: Der ReloadManager in der wingS Architektur

die benötigte HTML-Darstellung einer Komponente zu generieren. Somit profitieren auch alle bisher realisierten Renderer und Komponenten sofort und ohne jegliche weitere Mitwirkung von dieser Neuerung.

Im nächsten Schritt filtert der *ReloadManager* redundante Aktualisierungsanfragen, wie sie z.B. bei gleichzeitiger Änderung einer Komponente und ihrer Vater-Komponenten entstehen können, selbständig aus und sammelt die verblei-

benden Aktualisierungen. Aus diesen werden dann *Update*-Nachrichten erzeugt und an die Ajax-Engine im Browser gesendet von wo aus sie dann in die Seite eingearbeitet werden.

Einige wingS Komponenten, allen voran die Tabelle, gehen sogar noch einen Schritt weiter und unterstützen zudem feingranularere Operationen. Diese kommen beispielsweise zum Zuge, wenn der Benutzer durch Klick auf eine Zelle die Bearbeitung und damit einen *STableCell*-

Listing 1

Ein kleines Ratespiel mit wingS

```
public class HelloWingS {
    public HelloWingS() {
        SFrame frame = new SFrame();
        SForm p=new SForm(new SGridLayout(5,1));
        final SLabel msg = new SLabel();
        final STextField tf = new STextField();
        SButton ok = new SButton(«Raten!»);

        // Anordnung mittels GridLayout
        p.add(new SLabel(«Hello wingS!\n\n» +
            «Rate meine Zahl zwischen 0 und 9!»));
        p.add(msg);
        p.add(tf);
        p.add(ok);

        frame.getContentPane().add(p);
        frame.setVisible(true);

        // Die Spiellogik als EventListener
        final int num = new Random().nextInt(10);
        ok.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                if (("" + num).equals(tf.getText()))
                    msg.setText(«Richtig!»);
                else
                    msg.setText(«Nicht die "+tf.getText()»);
            }
        });
    }
}
```

Editor startet oder sich Selektionen bzw. einzelne Werte im *TableModel* geändert haben. Das erhöht die Geschwindigkeit weiter, spart Ressourcen und führt zu stabileren Darstellungen im Browser. Ebenfalls erwähnenswert ist, dass die inkrementelle Aktualisierungen der Seite im Vergleich zu anderen Frameworks vollständig optional geblieben ist und daher jederzeit die aktuelle Sicht in gewohnt klassischer Manier auch als vollständige HTML-Seite ausgeliefert werden kann.

Wege zurück zur Basis

Einer der zentralen Vorteile Java-orientierter MVC-Web-Frameworks ist, dass sie den Entwickler in der alltäglichen Arbeit vom Umgang mit den technischen Spezifikas und Einschränkungen der Web-Plattform entbinden. Statt mit JavaScript, Ajax, XML, HTML und CSS zusätzliche Hürden in den Weg gelegt zu bekommen, kann er sich voll auf die

Qualität und das Design seiner Java-Lösung konzentrieren. Speziell jedoch während der Entwicklung komplexerer Webapplikationen kann es immer wieder zu Detailanforderungen oder Problemstellungen kommen, die eine individuelle Lösung und daher auch einen Durchgriff auf die verwendeten Basistechnologien verlangen. Spätestens in diesem Moment zeigt sich, ob bei der Wahl der Mittel eine gute Balance zwischen hohem Abstraktionsniveau und guter Beherrschbarkeit der Basistechnologien gelungen ist.

Hier sticht *wingS* im Feld der Ajax-bewehrten Frameworks besonders hervor. Die geschilderte Architektur des *ReloadManagers* ermöglicht es jederzeit durch einfaches Neuladen auf die vollständige, generierte HTML-Sicht mitsamt aller eingebundener Skript- und CSS-Dateien zurückzugreifen, was die Entwicklung neuer Komponenten und die Fehlersuche

ungemein erleichtert. Auch beim Styling und Layout verleugnet *wingS* nicht seine Wurzeln. Alle Komponenten tragen bereits von Werk aus eindeutige CSS-Klassen über die sich ihr Look & Feel mittels Stylesheets einfach, mit der Web Developer Toolbar [4] sogar interaktiv an die eigenen Bedürfnisse anpassen lässt. Neben den Swing-Methoden beinhaltet die *wingS* API auch noch handverlesene Zusatz-Funktionen mit Web-Bezug. Diese erlauben es programmatisch Komponenten jederzeit eigene CSS-Klassen, Attribute oder JavaScript-Listener zuzuweisen. Obwohl *wingS* alle Swing-Layouts fast vollständig nachbildet, wird die Ausrichtung und das Sizing clientseitig weitestgehend über reine Standard-Mittel wie HTML und CSS realisiert. Dies ermöglicht auch einzelnen Container-Komponenten statische HTML Templates z.B. für ein fixes Basis-Layout zuzuweisen. All diese Mittel eröffnen somit einen mächtigen und flexiblen Zugang zu den zugrunde liegenden Web-Technologien, wohl gemerkt: nur für den Fall der Fälle.

Flugsicherung

Ein wichtiger Punkt, der gerade mit dem Schlagwort RIA immer mehr an Bedeutung gewinnt, ist das Thema Sicherheit. Generell impliziert Ajax, dass vermehrt Logik auf der Seite des Clients und damit in einer potenziell nicht vertrauenswürdigen Umgebung zum Ablauf kommt. Problematisch ist dabei besonders, wenn dies nicht nur Präsentations- sondern auch sicherheitsrelevante Teile der Geschäftslogik betrifft. Das kann besonders bei „echten RIA“ Ansätzen, wie sie in Google's Web Toolkit oder anderer JavaScript-basierten Ajax-Frameworks vertreten sind, schnell der Fall sein. Hier gilt es an der Schnittstelle zum Server die nötigen Sicherungsmaßnahmen zu ergreifen um spätestens dort wieder auf eine gesicherte Ausgangslage vertrauen zu können.

In *wingS* wird Ajax, und damit JavaScript, von Haus aus nur zur Umsetzung komfortabler Widgets und Seitenaktualisierungen genutzt, nicht jedoch für jegliche Geschäftslogik. Eine weitere Hürde stellt zudem noch der Low-Level Event-

wingS Komponenten im Selbstbau

Das einfache Beispiel einer eigenen Minimal-Komponente soll im Folgenden die einfache Erweiterbarkeit aber auch die innere Funktionsweise von *wingS* etwas veranschaulichen. Wie bereits erläutert nimmt selbst diese Minimal-Komponente bereits automatisch an der inkrementellen Aktualisierung teil. Die Mittel hierzu sowie ihre Eigenschaft als *wingS*-Komponente erbt sie über die gemeinsame Basis-Klasse *SComponent*. Wichtig ist nur, dass sie den *ReloadManager* über alle Aktualisierungen in ihrer Darstellung informiert. Im folgenden Fall erledigt das der Aufruf der Methode *reloadIfChange*.

```
public class MyComponent extends SComponent {
    private String payload;
    public MyComponent() {
        setCG(new MyComponentCG());
    }
    public void setValue(String v) {
        reloadIfChange(v, payload);
        this.payload = v;
    }
    public String getValue() {
        return payload;
    }
}
```

Unsere Komponente benötigt noch einen passenden Renderer, der sich um die Generierung

der HTML-Darstellung kümmert. Hierbei sollten auch alle bereits in der Basisklasse *SComponent* deklarierten Eigenschaften wie Größe, Ausrichtung und Farbe mit berücksichtigt werden, was wir uns aber aufgrund unseres einfachen Beispiels ersparen.

```
public class MyComponentCG
    extends AbstractComponentCG {
    public void writeInternal(Device d,
        SComponent c) throws IOException {
        MyComponent mc = (MyComponent) c;
        d.print("<span");
        Utils.optAttribute(d, "id", mc.getName());
        d.print(">").print(mc.getValue());
        d.print("</span>");
    }
}
```

In der gezeigten Variante ist unsere Komponente ein reines Textlabel. Soll die Komponente aktiv werden, d.h. Eingaben vom Client verarbeiten können, muss sie noch die Schnittstelle *LowLevelEventListener* implementieren, um vom Dispatcher die an ihre ID gerichteten Request-Parameter erhalten zu können. Noch mehr Möglichkeiten bei der Komponentenentwicklung kann man sich auf einfache Weise, z.B. durch die Verwendung der bereits mit *wingS* enthaltenen Yahoo UI Library [3] oder weiterer Ajax-Bibliotheken, erschließen.

jax[®] 08

**Europas Nr. 1 Konferenz
für Enterprise-Technologien & Strategien**

21.–25. April 2008, Wiesbaden

3-in-1 Konferenzpaket!

JAX buchen und die SOACON
sowie das Eclipse Forum Europe
gleichzeitig mitbesuchen!



Goldsponsoren:



Bronzesponsor:



Agile Day Sponsor:



Loungesponsor:



Präsentiert von:



Media-Sponsoren:



Veranstalter:



Anmeldung und Information unter www.jax.de

Dispatcher dar. Dieser ist die zentrale Instanz auf dem Server, welcher sich für die Annahme und Verteilung sämtlicher Client-Eingaben an die adressierten Komponenten verantwortlich zeichnet. Dabei prüft er auch, ob diese einer gültigen und nicht etwa einer veralteten Sicht entstammen und ob die adressierten Komponenten überhaupt aktiv und damit auch auf dem Client bedienbar sein sollten. Somit stehen zusätzlich zu der schon der Architektur entspringenden Eigenschaft, dass sämtliche Geschäftslogik in reinem Java auf einer gesicherten Umgebung im Server abläuft, auch noch grundlegende Sicherungsmechanismen der Client-Schicht bereit.

Alternativen

Zum Abschluss wollen wir noch eine kurze Gegenüberstellung mit den ebenfalls freien MVC-Frameworks Echo2 [6] und Eclipse RAP wagen. Beide können ebenfalls sehr beeindruckende Ergebnisse vorweisen und verfolgen mit ihren reinen Java-APIs durchaus ähnliche Ansätze. Das Echo2-Framework orientiert sich hierbei nicht ganz so stark an den bekannten Brüdern aus dem Desktop-Bereich und vernachlässigt einige Querschnitt-Details zugunsten einer einfacheren Implementierung. So bietet es nur stark reduzierte Layoutmöglichkeiten (horizontal, vertikal, matrix) und schwächelt durch den fehlenden *CellEditor*-Ansatz oder einer Aktualisierung auf Zellebene im Bereich der Tabellen. Dafür bietet es eine schicke Optik und eine hervorragende Cross-Browser-Kompatibilität.

Deutlich anspruchsvoller dagegen ist dort der Blick hinter die Kulisse. Aufgrund des Ansatzes, die Sicht exklusiv über DOM aufzubauen, kann das

aktuelle HTML-Dokument nur über zusätzliche Werkzeuge wie FireBug [7] eingesehen werden. Etwas problematisch dagegen ist, dass selbst mit diesen Hilfsmitteln der Zugriff auf große Teile des JavaScript- und Ajax-Teils verwehrt bleibt, was die eigene Komponentenentwicklung und Fehlersuche schwierig gestaltet. Viele Detailfunktionen wie Abstände, Ränder, Tastatursteuerung und Drag & Drop welche in wingS standardmäßig enthalten sind, rüstet erst das unabhängige Projekt EchoPointNG mit erweiterten Komponenten nach. Leider vermisst man dann auch wieder öfters den Blick unter die Haube des sonst sehr stabilen Frameworks.

Relativ neu auf dem Markt ist das Projekt Eclipse RAP [8], welches sich ehrgeizige Ziele gesetzt hat. Es ist analog zu den anderen Teilen in Eclipse als OSGi-Plug-in realisiert und möchte die API der Eclipse Rich Client Plattform (RCP) vollständig für Web-Anwendungen zugänglich machen. Das ist ein anspruchsvolles Ziel, beinhaltet RCP doch letztendlich alle Elemente der bekannten Eclipse IDE Workbench. Zum Einsatz kommt daher das Ajax-Framework quooxdoo, welches clientseitig selbst nochmals eine vollständige, recht gut gelungene Bibliothek an Widgets für JavaScript bereitstellt. RAP generiert daher selbst auch keinerlei HTML sondern ausschließlich JavaScript-Code der dann im Browser zur Ausführung kommt und die entsprechenden quooxdoo Widgets erzeugt, konfiguriert und nutzt. Jedes Widget hat somit immer zwei Komponenten-Repräsentationen: Eine „SWT“-Instanz auf dem Server und eine quooxdoo-Instanz auf dem Client (vgl. Half-Object Pattern). Auch beim Layout

greift RAP etwas tiefer in die Trickkiste, da es originale SWT Implementierungen verwendet. Bei Komponenten deren Pixelgröße nicht a priori bekannt sind, werden diese im Einzelfall erst unsichtbar im Browser gerendert, deren Maße an den Server zurückgesendet um dann erst im zweiten Schritt im Browser gelayoutet zu werden. Ein Styling dieser Widgets erlaubt quooxdoo, und damit auch RAP, über das Setzen von Properties der quooxdoo Widgets welche in RAP dann als Property-Datei abgelegt werden können.

Fazit

Dass die Entwicklung von Web-Anwendungen längst nicht mehr bedeuten muss, in Seiten, HTML und HTTP Requests zu denken, sondern dass auch eine komponenten- und eventorientierte Entwicklung möglich ist, ist inzwischen ein alter Hut. Nicht ganz so verbreitet dagegen ist wohl die Erfahrung, dass man hierbei im Entwicklungsalltag auf fachfremde Technologien wie XML, HTML oder JavaScript sogar gänzlich verzichten kann und sich gerade umfangreiche und komplexe Web-Applikationen wunderbar über reine Java-APIs realisieren lassen. Durch die immer stärker verbreitete Technologie „Ajax“ hat sich die Komplexität im Bereich der Web-Entwicklung zudem stark erhöht, so dass es immer wichtiger wird mit sinnvollen Abstraktionsstufen arbeiten zu können. Wer diese Erfahrung bislang vermisst hat, dem kann nur geraten werden, z.B. mit einem der genannten Frameworks, erste Gehversuche zu wagen. Hervorragende Lösungen stellen sie fraglos alle drei dar.

RAP unternimmt mit seinem Ansatz den ehrgeizigsten Versuch das Verhalten

Feature	wingS	Echo2	RAP
Programmiermodell	Swing-nahe	Properitär	SWT/JFace-nahe, Eclipse RCP
Container	Servlet	Servlet	OSGi
Styleability	CSS, Layout-Templates	XML Stylesheets	Quooxdoo Properties
Layouter	Grid, GridBag, Box, Flow, Template, Border	Row, Column, Grid	Row, Grid, Form, Fill
Layout-Berechnung	Client-Seitig	Client-Seitig	Client & Server-Seitig
Render-Technik	HTML, CSS, JavaScript	HTML via DOM, indirekt CSS	JavaScript-Code
Verwendete UI Kits	Yahoo UI Library [3]	Keine	quooxdoo [9]

Tabelle 1: Gegenüberstellung verschiedener MVC Webframeworks mit reiner Java API

und Programmiermodell seiner Eclipse Desktop-Umgebung auf das Web zu übertragen. Ob dabei seine Gesamt-Komplexität in der Anwendung beherrschbar bleibt, muss sich noch zeigen. Auf der anderen Seite gibt die verwendete OSGi- und RCP-Architektur bereits einen umfangreichen Rahmen vor, den es aber auch anzunehmen gilt.

Leichtgewichtiger sind dagegen Echo2 und wingS unterwegs. Beide besitzen sehr ähnliche Konzepte und grenzen sich mehr unter der Haube voneinander ab. Echo2 steckt vor allem bei den Möglichkeiten für Layout und Tabellen aber auch seinen Komponenten im Allgemeinen zurück, kann dafür aber mit schickeren MDI-Fenstern, optisch ansprechenden Zusatzkomponenten und einem kommerziellen GUI-Builder aufwarten.

wingS imitiert wortgetreu sein Vorbild Swing und bietet besonders Einsteigern mit entsprechendem Vorwissen einfachen Zugang und viele Möglichkeiten.

Im direkten Vergleich hat es durch volle CSS-Unterstützung, seinem einfachen Renderer-Ansatz und der Möglichkeit, für HTML-Layouts beim Thema Styling und Layout umfassende Möglichkeiten, um individuelle Designs umzusetzen. Die feingranularen Aktualisierungen und der original Swing-Funktionsumfang lassen es auch im Bereich der Standard-Komponenten etwas im Vorteil gegenüber Echo2 erscheinen.

Trotz des hohen Maßes an Kontrollmöglichkeiten über die Web-Schicht bleiben RIA-Features wie die Tastatursteuerung, Drag & Drop oder die inkrementellen Seitenaktualisierungen transparent für den Entwickler und belegen die gelungene Mischung zwischen Abstraktion und Detail. Eine Brücke in die Welt der agilen Entwicklung schlägt der neu erschienene WingSBuilder [10] und ermöglicht die dynamische Einbindung von wingS in Groovy-Skripte. Zukünftig steht u.a. das Thema Reverse-Ajax bzw. Comet

auf dem Plan der geplanten Neuerung. Es bleibt also in jedem Fall spannend.



Benjamin Schmid ist Software-Architekt und Technischer Consultant bei der eXcellent solutions GmbH. Neben seiner Mitarbeit bei zahlreichen Kundenprojekten liegen seine Schwerpunkte in Methodiken und Frameworks für graphische Oberflächen. Seit 2003 beteiligt er sich aktiv an der Weiterentwicklung des wingS Frameworks. Kontakt: B.Schmid@excellent.de

■ Links & Literatur

- [1] wingS Projektseite: www.wingsframework.org
- [2] Benjamin Schmid: Beflügelte Weiten, in *JavaMagazin* 7/2005, S.98
- [3] Yahoo! UI Library (YUI): developer.yahoo.com/yui/
- [4] Web Developer Toolbar: chrispederick.com/work/web-developer/
- [5] wingsframework.org/doc/papers/Diplomarbeit_Stephan_Schuster.pdf
- [6] Echo2: nextapp.com/platform/echo2/
- [7] FireBug: www.getfirebug.com
- [8] Eclipse RAP: eclipse.org/rap/
- [9] qooxdoo: qooxdoo.org
- [10] Groovy WingsBuilder: groovy.codehaus.org/WingSBuilder

ex|Xcellent solutions

eXcellent solutions bietet seinen Kunden umfassende Unterstützung bei der Konzeption und Umsetzung von IT-Projekten. Eine besondere Stärke ist die ganzheitliche Betrachtung, von der Analyse und Optimierung der Prozesse bis hin zur Implementierung der sich daraus ergebenden Ziel-Systemlandschaft.

Aus der Erkenntnis, dass es immer schwieriger wird, die Komplexität von Softwaresystemen zu beherrschen, entstand unsere aspektorientierte MDSD-Suite "orchideo". In ihr verbinden sich die Vorteile von aspektorientierter und modellgetriebener Softwareentwicklung zu einer verblüffenden Gesamtlösung mit herausragendem Komfort und maximaler Flexibilität. Mit Tools basierend auf Eclipse, EMF, GMF und openArchitectureWare (oAW) verschwinden die Grenzen zwischen Modellierung und Codierung. Aspektorientierung - verankert im Design der Software - macht Komplexität beherrschbar und ermöglicht Effizienz in der Entwicklung und Pflege der Systeme.

Anschrift

Beim Alten Fritz 2
D-89075 Ulm

Tele.: +49 (0)731 55026 10
Fax: +49 (0)731 55026 99

www.excellent.de

Hinter allem steht unser Anspruch, die ideale Softwarelösung für unsere Kunden zu erstellen.

Gerhard Gruber
Dipl.-EU-Betriebswirt (ebsi)
Geschäftsführer