Aktuell stellen sich viele Unternehmen die Frage, wie mit bestehenden Forms-Anwendungen in Zukunft umgegangen wird. Dabei werden verschiedene Migrations- und Modernisierungsansätze sowie mögliche Zielplattformen diskutiert. Hinsichtlich der Benutzeroberfläche strebt man oftmals eine Technologie an, die AJAX unterstützt. Ein Vertreter ist hier das Google Web Toolkit. Der Artikel zeigt, wie bestehende Forms-Module in eine GWT-Anwendung integriert werden können und wie sich durch diese clientseitige Integration neue Möglichkeiten bei der inkrementellen Modernisierung von Forms-Applikationen eröffnen.

Oracle Forms und GWT: Integration in eine AJAX-Web-Anwendung

Philip Stroh, METRO SYSTEMS GmbH

Das Google Web Toolkit (GWT) wurde 2006 unter Open-Source-Lizenz veröffentlicht und hat sich seitdem zu einem populären Framework zur Erstellung von Web-Anwendungen mit Java entwickelt. GWT ist speziell für die schnelle und effiziente Realisierung von AJAX-Anwendungen (AJAX = Asynchronous JavaScript and XML) ausgelegt. Diese Art von Web-Anwendungen gehört zu den sogenannten „Rich-Internet-Applications“, die dem Benutzer im Browser ein ähnliches Aussehen und Verhalten wie eine Desktop-Anwendung bieten. Sie sind somit moderne Vertreter des Konzepts, das auch Oracle Forms zugrunde liegt, und sie können unter anderem Interaktionstechniken wie Drag-and-Drop umsetzen. Dabei kommen die gängigen Technologien HTML, JavaScript und XML zum Einsatz. Somit ist im Unterschied zu Forms kein zusätzliches Plug-in zur Ausführung im Browser erforderlich.

Wie Oracle Forms können die Anwendungen mit kurzen Reaktionszeiten auf Benutzereingaben punkten. Die Eingaben werden clientseitig verarbeitet und die Oberfläche ohne Neuladen der Webseite aktualisiert. Hierzu werden die Inhalte asynchron vom Server nachgeladen und anschließend das Document Object Model (DOM) der Webseite angepasst.

Besonderheiten von GWT

Kern des Google Frameworks ist der Java-to-JavaScript-Compiler. Dieser wandelt Programmcode, der mit der Java-API von GWT entwickelt wurde, in JavaScript um. Der JavaScript-Code wird dabei optimiert für unterschiedliche Browser generiert. Dieses Vorgehen ermöglicht die Programmierung von dynamischen Web-Anwendungen, ohne dass der Entwickler Expertenwissen im Bereich HTML und JavaScript sowie der jeweiligen Browser-Spezifika benötigt.

Da die Entwicklung komplett in Java erfolgt, ermöglicht GWT im Gegensatz zu anderen Javascript-Frameworks den Einsatz einer mächtigen Java-IDE, etwa Eclipse, inklusive Debugging-Funktionalität. Das Toolkit kommt in der Präsentationsschicht zum Einsatz, bezüglich der Realisierung des Backends wird der Entwickler nicht eingeschränkt. Es können daher beliebige Java-APIs zum Zuge kommen, beispielsweise die in vielen Projekten etablierte Kombination von Spring und Hibernate.

Bei der Erstellung komplexerer Anwendungen bietet es sich an, auf eine der teilweise kostenpflichtigen Komponenten-Bibliotheken aufzusetzen. Diese erweitern den Umfang der von GWT bereitgestellten Oberflächenelemente um zusätzliche Typen und Funktionen. Beispielsweise beinhaltet Ext GWT (GXT) eine Tabellenkomponente, die die Sortierung der Einträge, das Ein- und Ausblenden von Spalten und einen Mechanismus zum Blättern durch eine Ergebnisliste ermöglicht.

Inkrementelle Modernisierung

Bei der Modernisierung von Forms-Anwendungen sind verschiedene Herangehensweisen möglich. Ein Ansatz ist zum Beispiel die toolgestützte, automatisierte Migration der Forms-Module in eine gewünschte Zielsprache.

Eine alternative Methode ist eine Erneuerung der Applikation, bei der die Anwendung nicht nur technologisch auf eine zukunftssichere Basis umgestellt wird, sondern gleichzeitig auch ein fachliches Redesign stattfindet. Da diese Umstellung meist ein größeres Projektvorhaben ist, empfiehlt sich ein iterativ inkrementelles Vorgehen. Hierzu wird die Anwendung in fachliche Komponenten eingeteilt, die stufenweise in die neue Technologie überführt werden. Dabei ist die Aufteilung und Größe der einzelnen Stufe sicherlich in großem Maße von den Projektgegebenheiten abhängig. Was die Projekte, bei denen dieses Vorgehen gewählt wird, gemeinsam haben, ist die Tatsache, dass für eine gewisse Zeit Bestandssystem und neue Anwendung parallel betrieben werden. Dies bedeutet unter anderem, dass Anwender in vielen Fällen in beiden Applikationen arbeiten und zwischen den Systemen wechseln müssen.

User-Interface-Integration

An dieser Stelle bietet die Integration der Anwendungsoberflächen von bestehender und neuer Technologie Vorteile. Sie macht Dialoge aus späteren Migrationsstufen über die neue Oberfläche zugänglich. Dies kann über einen Link erfolgen, über den ein bestimmter Dialog angesprochen wird. Es ist aber auch möglich, Forms-Dialoge direkt in eine Web-Anwendung ein-

zubetten. Im Folgenden wird diese Integration am Beispiel von Forms und GWT vorgestellt, dabei wird sowohl die Einbindung in die GWT-Benutzeroberfläche als auch die Kommunikation zwischen den beiden Technologien gezeigt.

Einbettung per iFrame

Um einen Forms-Dialog innerhalb einer GWT-Anwendung zur Verfügung zu stellen, muss man diesen in die Web-Oberfläche integrieren. Zuerst ist die Forms-Anwendung entsprechend zu konfigurieren. Über einen Applet-Parameter kann festgelegt werden, ob das Applet im Browser eingebettet oder in einem eigenständigen Fenster angezeigt wird. Damit das Applet direkt im Browser erscheint, ist die Forms-Anwendung mit der Konfiguration „separateFrame=false“ zu betreiben (formsweb.cfg).

Ist diese Konfiguration eingerichtet, kann die Einbindung in die Web-Oberfläche als iFrame erfolgen. Als Quelle wird der Link zur Forms-Anwendung angegeben. GWT stellt für die Erstellung von iFrames eine Klasse (Frame) zur Verfügung, die einfach genutzt werden kann (siehe Abbildung 1).

Kommunikation per JavaScript

Mit der Einbindung von Forms per i-Frame ist der erste Schritt zur User-Interface-Integration geschafft, beide Technologien können dem User in einem gemeinsamen Kontext präsentiert werden. Allerdings kann es in den meisten Fällen auch notwendig sein, dass zum Beispiel eine Aktion des Users per Event zwischen den Applikationen übertragen wird.

Damit Forms im Browser mit einer Web-Anwendung kommunizieren kann, ist eine gemeinsame Sprache notwendig, die es ermöglicht, Events oder Daten von einer Anwendung zur anderen zu senden. Hier kann JavaScript zum Einsatz kommen, da die Applet-Technologie, auf der Oracle Forms aufgebaut ist, dafür eine Schnittstelle bereitstellt. Durch die LiveConnect-Schnittstelle des Java-Plug-ins ist es möglich, öffentliche Methoden des Applets per JavaScript aufzurufen sowie JavaScript-Code aus dem Applet heraus auszuführen (siehe Listings 1 und 2).

```
document.AppletName.raiseMes-
sage();
```

Listing 1: Aufruf einer Applet-Methode per JavaScript

```
JSObject win = JSObject.
getWindow((Applet) this);
win.eval(“window.open
(‘www.oracle.com’)”);
```

Listing 2: Aufruf einer JavaScript-Methode aus dem Applet

Eigene JavaScript-Schnittstelle in Forms

Ab Version 11*g* bietet Oracle Forms basierend auf LiveConnect eine neue JavaScript-Programmier-Schnittstelle an. Aber schon mit vorherigen Versionen können aus Forms heraus mithilfe einer JavaBean JavaScript-Befehle abgesetzt und verarbeitet werden. Der folgende Abschnitt zeigt anhand von Code-Snippets, wie man per Java Script ein Event in einer Forms-Anwendung auslöst. Zuerst wird hierzu das Standard-Applet von Forms um eine öffentliche Methode erweitert. Diese lässt sich per JavaScript aufrufen und nimmt im Beispiel einen Meldungstext entgegen. Das neue Applet wird auf dem Forms-Server anstelle des Standard-Applets konfiguriert (siehe Listing 3).

Als Nächstes wird eine JavaBean erstellt und in einem Forms-Modul als Bean-Area eingebunden. Während der Initialisierung meldet sich die Bean beim Applet an. Somit kann das Applet die Meldung an die JavaBean weiterleiten, die daraufhin ein CustomEvent auslöst (siehe Listing 4).

Das durch die JavaBean ausgelöste CustomEvent wird im „WHEN CUSTOM ITEM EVENT“-Trigger der Bean-Area abgefangen und kann beispielsweise per Alert zur Anzeige gebracht werden (siehe Listing 5).

Die bereitgestellte Methode im Applet kann nun per JavaScript aufgeru-

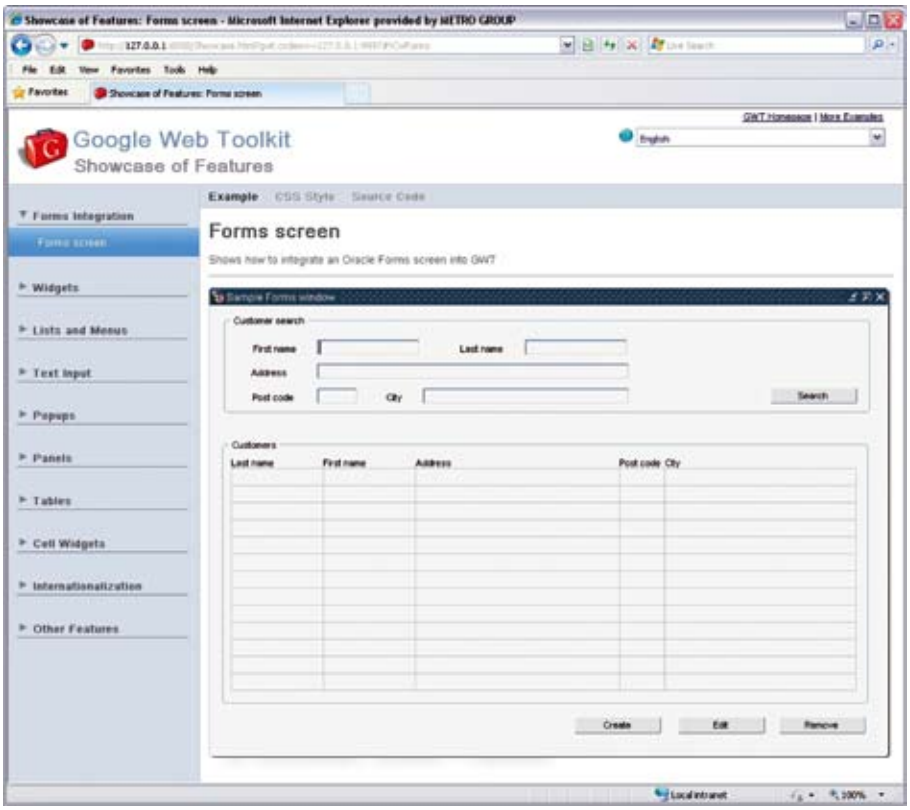

Abbildung 1: Beispiel einer Integration in die GWT-Showcase-Anwendung

```
public class MyApplet extends oracle.forms.engine.Main {
      private JavaScriptBean jsBean;

      public void setJsBean(JavaScriptBean jsBean) {
            this.jsBean = jsBean;
      }

      public void raiseMessage(String msg) {
            jsBean.raiseMessage(msg);
      }
}
```

Listing 3: Erweiterung des Forms-Applets

```
public class JavaScriptBean extends VBean {
      static final ID VALUE = ID.registerProperty(„value“);
      IHandler handler;

      public void init(IHandler handler) {
          super.init(handler);
          this.handler = handler;

          // get applet and set bean
          MyApplet myApplet = (MyApplet) handler.getAp-
plet();           myApplet.setJsBean(this);
      }

      public void raiseMessage(String msg) {
            // simplified: try-catch removed
            handler.setProperty(VALUE, msg);
            CustomEvent event = new CustomEvent(handler,
“MSG”);
            dispatchCustomEvent(event);
      }
}
```

Listing 4: JavaBean

```
DECLARE
  event_paramsPARAMLIST;
  param_type  NUMBER;
  event_name  VARCHAR2(30) := :SYSTEM.CUSTOM_ITEM_EVENT;
  event_value VARCHAR2(2000);
  dummy_var   NUMBER;
BEGIN
  -- access event data
  event_params := get_parameter_list
                (:SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS);
  IF (event_name = ‚MSG‘) THEN
    GET_PARAMETER_ATTR(event_params, ‚value‘,
                       param_type, event_value);
    -- method to raise alert
    handle_message(event_value);
  END IF;
END;
```

Listing 5: „WHEN CUSTOM ITEM EVENT”-Trigger

fen werden und im Forms-Modul erscheint der als Parameter übergebene Meldungstext.

Der Aufruf der Applet-Methode ist auch aus GWT über das JavaScript-Native-Interface (JSNI) möglich, das die Erstellung von nativen JavaScript-Methoden erlaubt. Angenommen die Forms-Applikation ist in GWT per iFrame eingebunden, dann könnte eine Implementierung wie im nachstehenden Listing aussehen. Dabei ist eine Einschränkung der iframe-Einbettung zu beachten, GWT- und Forms-Anwendung müssen aufgrund der Same-Origin-Policy von JavaScript innerhalb einer Domain bereitgestellt werden (siehe Listing 6).

```
public native void sendMessage(Element frame,
String msg) /*-{
      frame.contentWindow.MyApplet.
raiseMessage(msg);
}-*/;
```

Listing 6: Native JavaScript-Methode in GWT

JavaScript API in Forms 11*g*

Mit Forms 11*g* lässt sich diese Lösung wesentlich einfacher umsetzen, denn hier steht das neue Feature „JavaScript-API“ zur Verfügung. Mit dieser neuen Schnittstelle erhält das Forms-Applet eine öffentliche Methode „raiseEvent“, die per JavaScript aufgerufen werden kann. Beim Aufruf dieser Funktion wird der Trigger „WHEN CUSTOM JAVASCRIPT EVENT“ ausgelöst. In diesem kann, ähnlich wie zuvor gezeigt, mittels zweier Systemvariablen auf das Event und dessen Daten zugegriffen werden. Die im Beispiel beschriebene Erweiterung des Applets und die Erstellung der JavaBean ist somit nicht mehr notwendig, um die Interaktion zwischen einer Forms- und eine Web-Anwendung zu realisieren. Des Weiteren werden durch die neue JavaScript-API Funktionen zur Verfügung gestellt, die die Ausführung von JavaScript-Befehlen direkt aus dem Forms-Code ermöglichen.

Fazit

Der Artikel zeigt, wie Forms-Dialoge in die Oberfläche einer GWT-Anwendung eingebunden werden und mit dieser per JavaScript kommunizieren können. Die JavaScript API mit Forms 11*g* stellt in diesem Zusammenhang eine wirkliche Vereinfachung dar, doch auch mit vorherigen Versionen ist eine Kommunikation per JavaScript möglich. Neben der interessanten technologischen Komponente bietet der vorgestellte Ansatz der Client-seitigen Integration neue Möglichkeiten bei der inkrementellen Modernisierung einer Forms-Applikation, die aus fachlicher und betriebswirtschaftlicher Sicht einen Mehrwert schaffen.

Kontakt:
Philip Stroh
philip.stroh@metrosystems.net