Plugins, APEX und ich – eine Einführung

Sebastian Wittig, Daniel Horwedel
merlin.zwo InfoDesign GmbH & Co. KG
Karlsruhe / Bad Liebenzell

Schlüsselworte
APEX, Plugin, Dynamic Action, Page Item

Einleitung
Oftmals werden in APEX-Anwendungen Code-Bestandteile entwickelt, die schon einmal in einer Anwendung implementiert wurden oder bei zukünftigen Entwicklungen benötigt werden können. In den meisten Programmiersprachen bietet sich in diesem Falle die Erstellung von Libraries an – APEX allerdings bietet ein solches Feature nicht. Dennoch lässt sich auch bei der Entwicklung mit APEX Code effizient wiederverwenden, in dem dieser in Plugins gekapselt und für andere Anwendungen zur Verfügung gestellt wird.

Szenario
In diesem Vortrag werden zwei häufig benötigte Anforderungen behandelt, für die jeweils ein Plugin erstellt wird.

Shuttle-Item mit Filter-Funktion

Das standardmäßig durch APEX zur Verfügung gestellte Shuttle-Item eignet sich sehr gut zur Verwendung mit relativ geringen anzuzeigenden Datenmengen. Sollten allerdings große Datenmengen mittels eines Shuttle-Items zugeordnet werden, wird das Element schnell unübersichtlich. Diese Problematik lässt sich mit etwas JavaScript oder einer Dynamic Action, indem ein Eingabeelement zur automatischen Filterung der Daten implementiert wird, lösen.

Ein möglicher Ansatz ist auch in einschlägigen Blogs benannt. Er sieht vor dem Shuttle Item ein übergeordnetes, kaskadierendes Elternelement (Cascading LOV Parent Item) zuzuordnen. Durch diesen Ansatz kann dem Shuttle Item mit geringem Aufwand ein Textfeld zugordnet werden, welches die Wertemenge des Shuttles dann entsprechend filtert. Dieses Vorgehen hat jedoch leider einen entscheidenden Nachteil. Es wird nicht nur die linke Seite des Shuttles gefiltert, sondern auch bereits ausgewählte Werte wieder entfernt. Doch gerade bei großen Wertemengen möchte man selten gleichartige Werte selektieren, sondern gegenfalls. ganz unterschiedliche heraussuchen.

Autocomplete-Item mit separatem Return-Value

Die Verwendung von klassischen Dropdown-Select-Listen bietet sich bei der Auswahl von Werten an, die bereits in der Datenbank vorhanden sind. Sollen allerdings neue Werte hinzugefügt werden, so musste in früheren APEX-Versionen zusätzlich ein Eingabefeld bereitgestellt werden, um diese neuen Werte hinzufügen zu können. Seit Version 4.0 bietet APEX einen neuen Elementtyp an, mit dem dieses Problem sehr einfach im Web 2.0-Stil gelöst werden kann: das Autocomplete-Eingabefeld. Dieses Element bietet dem Benutzer ein Eingabefeld, das im Hintergrund automatisch anhand der bereits eingegebenen Zeichen nach in der Datenbank vorhandenen passenden Werten sucht und diese analog einer Select-Liste zur Auswahl vorschlägt. Eine Problematik hierbei: der angezeigte Wert entspricht dem an die Anwendung zurückgegebenen Wert. Zur Umgehung dieses Problemes lässt sich eine Dynamic Action definieren, die bei einer Änderung des Wertes des Autocomplete-Feldes den zur Eingabe zugehörigen Rückgabewert ermittelt und in einem separaten, ausgeblendeten Seitenelement speichert.

Lösungsstrategien
Die vorgestellten Szenarien lassen sich problemlos innerhalb einer APEX-Anwendung implementieren – allerdings muss die Umsetzung bei jeder Seite, die diese Funktionen nutzt, erneut durchgeführt werden. An dieser Stelle bietet sich die Kapselung der Funktionalität an, wofür die Kombination aus Oracle-Datenbank und APEX-Framework mehrere Ansätze bietet:

- **Implementierung als Datenbank-Procedure?**
  Eine Implementierung als wiederverwendbare Datenbank-Procedure ist nicht sinnvoll möglich, da es sich bei den benötigten Elementen um GUI-Elemente handelt, bei denen der Fokus auf den Daten und nicht auf der Business-Logik liegt.

Import einer APEX-Seite
Bei der Erstellung von komplexeren Benutzeroberflächen lässt sich die Implementierung durch den Export einer „Master-Seite" mit dem anschließenden Import der Kopie mehrfach verwenden. Aufgrund des hohen Anpassungsaufwandes an der importierten Seite ist dieser Weg aber ebenfalls nicht zu empfehlen, zumal eine Kombination der beiden benötigten Elemente damit nicht realisierbar ist.

- **JS-Library**
  Da es sich bei der Anforderung primär um ein GUI-Thema handelt, ist eine Realisierung mittels einer JavaScript-Library möglich – allerdings ist damit zumindest beim Autocomplete-Element kein direkter Zugriff auf die zugrundeliegende LOV möglich, weshalb alle in der LOV möglichen Werte vorab in den Browser geladen werden müssen, was zu Performance-Problemen führt.
- **Kombination aus diesen Möglichkeiten?**
  Jede der vorgestellten Lösungsmöglichkeiten ist für sich betrachtet für das beschriebene Szenario nicht sinnvoll anwendbar. Eine Kombination aus den Vorteilen der einzelnen Varianten allerdings ermöglicht eine effiziente Umsetzung. APEX bietet hierfür sogar eine definierte Schnittstelle: die APEX-Plugins eignen sich perfekt zur Umsetzung der Anforderungen.

Was sind Plugins überhaupt?
Abstrakt ausgedrückt ist ein Plugin eine gekapselte, möglichst standardisierte, Funktionalität. Die Vorteile sind hierbei in den Adjektiven versteckt.
Gekapselt bedeutet, dass alle zu dieser Funktionalität gehörenden Komponenten innerhalb des Plugins integriert werden können. Unabhängig davon ob zusätzliche Bilder, CSS-Dateien oder JavaScript Fragmente benötigt werden; diese müssen nicht in den Shared Components, sowie auf Seiten- oder Templateebene zusammengesucht und auf ihre Zusammengehörigkeit untersucht werden.

Vielmehr sind in Plugins alle diese Komponenten zusammen mit der darauf zugreifenden Funktionalität gekapselt.

Diese Kapselungsstrategie kann sogar soweit getrieben werden, dass sämtliche PL/SQL Funktionalität nicht innerhalb des anonymen PL/SQL Blocks stattfindet, sondern in ein Datenbankpackage ausgelagert wird. Erfahrungsgemäß kommt dieser Schritt dem Entwicklungsvorgang zu Gute, da er das Arbeiten in geläufigeren Code-Editoren (SQL-Developer, Notepad++, TOAD o.ä.) ermöglicht. Allerdings sollten die so entstandenen Funktionen und Prozeduren später in das Plugin eingefügt werden, da man diese sonst nur innerhalb der Anwendungen verwenden kann, deren Parsing Schema auch auf das zugrundeliegende Package zugreifen kann. In bestimmten Fällen könnte dies natürlich gewünscht sein, unter dem Paradigma dass ein Plugin Funktionalität und keine Logik enthalten sollte, würden wir es zu vermeiden suchen.

Der zweite große Vorteil ergibt sich aus dem Wort standardisiert: Diese Plugins können, einen entsprechend generischen Ansatz bei der eigentlichen Erstellung vorausgesetzt, fast

kontextunabhängig wiederverwendet werden. Durch das Herunterladen wird ein SQL-Skript erzeugt, welches die nötigen Einträge in den APEX-internen Metatabellen vornimmt. Anschließend kann das Plugin in fremden Anwendungen aufwandsarm wiederverwendet werden.

Trotz dieser Vorteile erweitern Plugins die Funktionalität von APEX an sich nicht. Alle Funktionen werden durch das Zusammenspiel der Datenbank- und Webtechnologien erreicht die überall im Framework genutzt werden können. Allerdings erhöht sich durch die Wiederverwendbarkeit und damit die Effizienz der Codegeneration.

Wo gibt es fertige Plugins?
Zum Auffinden von Plugins gibt es eine ganze Reihe Möglichkeiten. Bitte sehen Sie diese Auswahl daher als subjektiv an, sie dient nicht der Werbung, sondern dazu eine kleine Auswahl unseres Erachtens nützlicher Plugins vorzustellen.

Eine kleine Auswahl Plugins gibt es bei auf der offiziellen Oracle Application Express Seite, beispielsweise den absolut unverzichtbaren, für jede Anwendung unerlässlichen, Facebook Like Button. Ein unseres Erachtens besonders hilfreiches Plugin sei an dieser Stelle erwähnt: Die Simple Checkbox, sie erleichtert den Umgang mit zweiwertigen Datenfeldern, da eine Checkbox dieses Typs einen Return-Value besitzt.

Eine möglichst umfassende Auswahl möchte Apex-Plugins bieten. Diese Webseite ermöglicht den Nutzern in Selbstverwaltung neue Plugins zu verlinken und zu kategorisieren. Die meisten frei zugänglichen Plugins sind hier aufzufinden.

So auch die Plugin von Carsten Czarski. Diese kann man natürlich im Rahmen der allgemeinen, von Oracle zur Verfügung gestellten Plugins verordnen. Aufgrund seines unermüdlichen Engagements für die deutsche APEX und PL/SQL Community im Allgemeinen, und die Plugin-Webinare im Besonderen, sei er hier erwähnt. Seine Plugins finden sie z.T. allerdings auch als Komponenten der Packaged Applications. Dort können diese untersucht werden.

Darüber hinaus gibt es natürlich verschiedenste z.T. kommerzielle Anbieter, die Plugins auf ihren Webseiten anbieten. Rein exemplarisch seien in diesem Zusammenhang Skillbuilders, deren Modal Page Plugin eine sehr ähnliche „Out-of-the-box“ Alternative in APEX 5.0 erhält, Enkitec, und FOEX erwähnt.

Wie sind APEX-Plugins aufgebaut?
Ein APEX-Plugin kann als eine Art Container betrachtet werden, der sowohl die Definition und PL/SQL-Logik des Plugins, wie auch zusätzlich benötigte Komponenten wie JavaScript-Code, CSS- und Grafikdateien enthält.

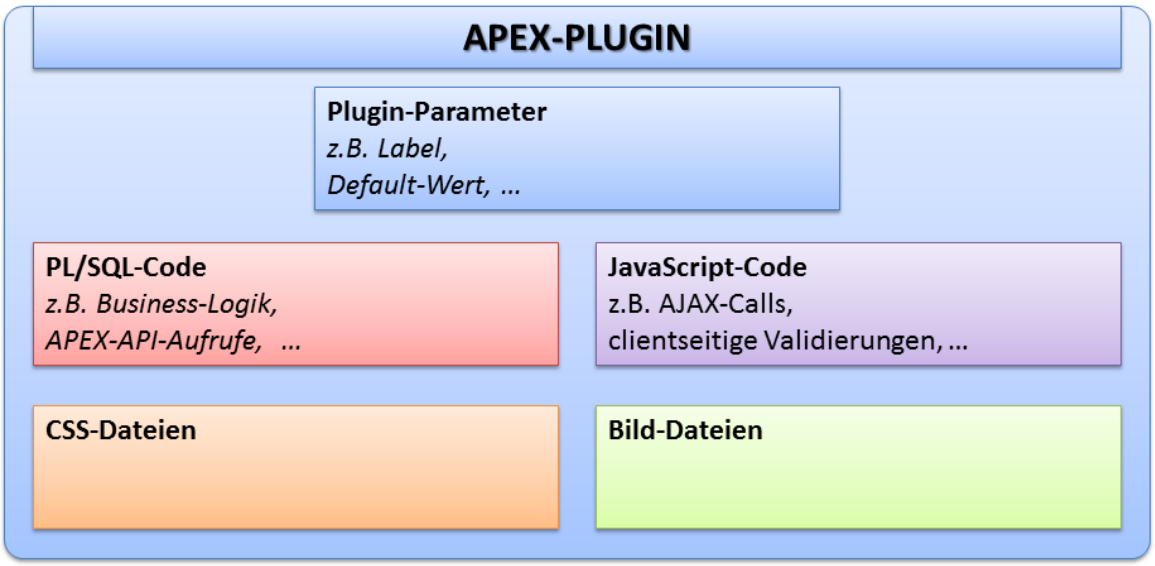

Abbildung 1: Aufbau eines APEX-Plugins

APEX ermöglicht die Implementierung von sechs verschiedenen Plugin-Typen:

- Authentifizierungs-Schema

  Mittels einem Authentifizierungs-Schema-Plugins können eigene, einheitlich wiederverwendbare Authentifizierungs-Schemas wie z.B. eine LDAP-Authentifizierung oder ein Single Sign On, implementiert werden.

- Authorisierungs-Schema

  Analog zum Authentifizierungs-Schema-Plugin lassen sich auch für die Authorisierungs-Schemas Plugins erstellen, mit deren Hilfe ein Authorisierungs-Schema gekapselt und wiederverwendet werden kann.

- Element

  APEX bietet standardmäßig sehr viele verschiedene Element-Typen für die verschiedensten Einsatzzwecke. Sollten jedoch Elemente benötigt werden, die von APEX nicht zur Verfügung gestellt werden, so lassen sich diese durch die Verwendung von Element-Plugins „nachrüsten“ - so sind zum Beispiel filterbare Shuttle-Elemente oder HTML5-Eingabetypen wie Kamera oder GPS damit realisierbar.

- Prozess

  Auch die Geschäftslogik lässt sich mittels Plugins kapseln – hierbei muss aber präzise evaluiert werden, ob sich hier nicht eher die Verwendung eines PL/SQL-Packages in der Datenbank anbietet. Dennoch lassen sich APEX-Seitenprozesse problemlos auch als Plugin zur Verfügung stellen.

- Region

  Mittels Region-Plugins lassen sich komplette Regionstypen wie z.B. eigene Chart-Darstellungen, selbst implementierte Berichte oder Geodatendarstellungen basierend auf externen Anbietern wie Google Maps oder OpenStreetMaps als Plugin zur Verfügung stellen.

- Dynamic Action

  Auch die in APEX verfügbaren Dynamic Actions lassen sich um eigene Aktionstypen erweitern: so sind hiermit per AJAX nachgeladene Benachrichtigungen oder auch zeitgesteuerte Aktionen wie z.B. das automatische Speichern der Nutzereingaben nach Ablauf eines bestimmten Zeitintervalles mittels einer Dynamic Action realisierbar.

Abhängig von der Auswahl des Plugin-Typs stehen unterschiedliche Optionen und Standard-Attribute zur Verfügung, mit denen das Verhalten des Plugins detailliert festgelegt werden kann.

Was passiert im Hintergrund?
Hauptsächlich wird im Hintergrund auf zwei APEX-Packages zurückgegriffen. Namentlich APEX_PLUGIN und APEX_PLUGIN_UTIL. Diese definieren die Standards und Typen, welche von einem Plugintyp zurückgegeben werden und stellen Hilfsroutinen für das Debuggen zur Verfügung.

APEX_PLUGIN erledigt dabei den Hauptteil der Arbeit. Das Package definiert eine umfangreiche Sammlung an Records, die hauptsächlich zum einen die Rückgabewerte der entsprechenden Pluginfunktionen, bspw. der Renderfunktionen eines Regions/Page Items Plugins, zum anderen die eigentlichen Inputparameter geschrieben werden.

Der PL/SQL Block eines Plugins natürlich mehrere unterschiedliche Funktionen und Prozeduren beinhalten. Darüber hinaus können auch andere Packages genutzt werden. Es bietet sich zum Beispiel das APEX_ITEM Package zur Generierung von Eingabefeldern an.

Am Wichtigsten ist die Callback Function, die in dem entsprechenden Attribut unterhalb des Blocks eingetragen wird. Für ein Page Item sieht diese so aus:

```
function <name of function> (
    p_item   in apex_plugin.t_page_item,
    p_plugin in apex_plugin.t_plugin,
    p_value  in varchar2 )
    return apex_plugin.t_page_item_validation_result
```

Als Inputparameter dient ein Parameter vom Typ T_PAGE_ITEM.

```
type t_page_item is record (
    id                          number,
    name                        varchar2(255),
    label                       varchar2(4000),
    plain_label                 varchar2(4000),
    placeholder                 varchar2(255),
    format_mask                 varchar2(255),
    is_required                 boolean,
    lov_definition              varchar2(4000),
    lov_display_extra           boolean,
    lov_display_null            boolean,
    lov_null_text               varchar2(255),
    lov_null_value              varchar2(255),
    […]
    attribute_10                varchar2(32767) );
```

Diese gekürzte Übersicht soll vor allem eines verdeutlichen: Innerhalb des Records werden die

Attribute übergeben die Sie innerhalb der entsprechenden Page Item Komponente festgelegt haben. Ob beispielsweise beim Submit implizit eine NOT NULL-Validierung stattfindet wird über das IS_REQUIRED Attribut entschieden. Eine andere Möglichkeit ist das Format welches die Anzeige im Falle eines numerischen oder Date-Datentyps haben soll. Es kann also, innerhalb der Darstellung des Page-Items anschließend durchaus auf entsprechend gesetzte Attribute reagiert werden. Ob dem Entwickler seinerseits entsprechende Attributsets zur Verfügung stehen wird über die Standard Attributes über Checkboxen gewohnt einfach definiert. Je nach Typ des Plugins stehen hier natürlich unterschiedliche Subsets zur Verfügung.

Der zweite Parameter p_plugin ist bei jedem Plugin enthalten und enthält Informationen über das aktuelle Plugin, darunter den Namen, das Präfix für hinzugefügte Dateien sowie möglicherweise definierte Attribute.

```
type t_process is record (
    id                  number,
    name                varchar2(255),
    success_message     varchar2(32767),
    attribute_01        varchar2(32767),
 […]};
```

Zuletzt gibt es für jeden Plugintyp einen Rückgabewert, dieser bestimmt weitere Attribute für die Darstellung – bzw. weitere Verarbeitung – innerhalb von APEX. Für ein Page Item kann man bestimmen ob zu dem Item navigiert werden kann und ggfs. wie die entsprechende ID innerhalb des DOM sein soll.

```
type t_page_item_render_result is record (
    is_navigable    boolean default false,
    navigable_dom_id varchar2(255)        );
```

Jeder Plugin-Callback folgt also, trotz der Vielzahl möglicher Typen, einem vergleichsweise festen Muster. Der Aufruf des Plugins enthält immer auch einen entsprechenden Parameter, welche die Attribute der Komponente innerhalb des PL/SQL Blocks adressierbar macht. Ein weiterer Parameter enthält die Attribute die innerhalb des Plugins hinzugefügt werden und im Rückgabewert kann man darüber hinaus weitere Attribute manipulieren.

Die APEX_PLUGIN_UTIL stellt darüber hinaus verschiedene Hilfsfunktionen und –prozeduren zur Verfügung. So können mit diesem Package Dynamic Actions debugged werden oder in Abhängigkeit eines BOOLEAN-Wertes Sonderzeichens eines Strings escaped werden. Wie so häufig im Umgang mit Packages ist es dabei nicht entscheidend jede Funktion und Prozedur inklusive Parametern fehlerfrei wiedergeben zu können. Vielmehr sollte man über die Existenz der entsprechenden API aufgeklärt sein und die Dokumentation im Zweifelsfall konsultieren, weshalb wir an dieser Stelle auf ausufernde Beschreibungen von Funktionen verzichten.

Entwicklungsprozess

Um nicht die ganze Vorfreude auf den eigentlichen Vortrag zu nehmen, sei an dieser Stelle auf den am 20.11.2014 von 10:00-10:45 im Raum Istanbul stattfindenden Vortrag verwiesen indem auf die Details des Entwicklungsprozess und Strukturierung der Komponenten eingegangen wird.

Im Anschluss an die DOAG werden Sie die Möglichkeiten haben die Präsentationsfolien ebenfalls herunterzuladen.

Fazit
Plugins stellen eine effiziente Möglichkeit dar, Bestandteile von APEX-Anwendungen wiederverwendbar zu kapseln und für andere Anwendungen zur Verfügung zu stellen. Insbesondere bei komplexeren Implementierungen, die nicht nur ausschließlich aus Business-Logik oder GUI-Elementen, sondern aus einer Kombination aus Datenbanklogik, Benutzeroberflächen-Definitionen mit HTML und JavaScript sowie Grafiken und CSS bestehen, empfiehlt sich die Verwendung der Plugins als „Container“ zur einfachen Bereitstellung und Nutzung von Anwendungsbestandteilen.

Kontaktadresse:
Sebastian Wittig, Daniel Horwedel
merlin.zwo InfoDesign GmbH & Co. KG
Karmelstraße 9
D-75378 Bad Liebenzell

Telefon: +49 (0) 7052 508 98 0
Fax: +49 (0) 7052 508 98 30
E-Mail sebastian.wittig@merlin-zwo.de
daniel.horwedel@merlin-zwo.de
Internet: http://www.merlin-zwo.de