APEX verbessern mit Browser-Extensions

Till Albert
MT AG
Ratingen

Schlüsselworte
APEX, JavaScript, JQuery, HTML5

Einleitung

Wer kennt das nicht, beim Entwickeln mit APEX im Page Designer; irgendein kleineres Feature fehlt, oder etwas sollte anders funktionieren. Dabei kann es sehr einfach sein die Standard-Funktionalität zu erweitern, wie etwa, zusätzliche Informationen passend zum Kontext anzuzeigen oder ganze JavaScript-Tools einzubinden. In diesem Vortrag soll es darum gehen, wie man dies mithilfe von Browser-Extensions umsetzen kann.

Was sind Browser-Extensions?

Tatsächlich werden die Begrifflichkeiten häufig durcheinander gebracht. Plug-Ins, wie etwa das für die Java Runtime Environment (JRE) oder Adobe Flash, sind schon länger als Sicherheitslücke bekannt und werden kaum noch eingesetzt. Plug-Ins reagieren auf bestimmte Mime-Types oder werden direkt in die Webseite über entsprechende HTML-Tags eingebunden, etwa zum Anzeigen einer Flash-Animation. Dabei übergibt der Browser Inhalte der Website an das Plug-In und zeigt den Rückgabewert wieder an, wobei dieses nur teilweise unter der Kontrolle des Browsers steht. Der Vormarsch von JavaScript und insbesondere HTML5 haben viele Plug-Ins überflüssig gemacht.

Browser-Extensions dagegen laufen in einer eigenen Umgebung im Browser, Websites haben keinen Zugriff auf diese und können somit keine Sicherheitslücken ausnutzen. Diese Erweiterungen basieren auf JavaScript, HTML und CSS und können mit der geladenen Website sowie auch mit dem Browser selbst interagieren. Das bedeutet natürlich nicht, dass solche Erweiterungen per Se sicher sind, da der Entwickler/Herausgeber selbst natürlich trotzdem bösartigen Code in der Extension unterbringen kann, wie das Einfügen von Werbung oder sonstigen unerwünschten Inhalten. Daher sollte man auf die Bewertungen vor dem Download schauen. Im Zweifel kann man sich den Code der runtergeladenen Erweiterung auch einfach mit entsprechenden Tools im Browser anschauen.

Obwohl die meisten Browser dieselben Erweiterungen meinen, verwenden diese unterschiedliche Namensgebungen. So wird bei Mozillas Firefox der Begriff Addons benutzt, während solche Erweiterungen für Googles Chrome Browser-Extensions genannt werden.

Die typischen Einsatzzwecke der Browser-Erweiterungen sind das Verwalten von Lesezeichen und Tabs oder die Löschung von der Browser-History und dem Cache nach bestimmten Kriterien. Darüber hinaus natürlich auch jegliche Aufgaben die das manipulieren der Website (DOM) betreffen. Das können z.B. folgende sein:

- Das Entfernen von Werbung auf der Webseite
- Das Übersetzen der gesamten Webseite in eine andere Sprache
- Nützliche Informationen zum Kontext der Seite anzeigen
- Neue Features zum Page-Designer von APEX hinzufügen

Aufbau und Funktionsweise von Browser-Extensions
Es gibt verschiedene Möglichkeiten für den Aufbau einer Browser-Extension. In vielen Fällen wird eine sogenannte Background-Page benötigt, eine unsichtbare Seite die im Hintergrund die Haupt-Logik beinhaltet. Auch weitere HTML-Seiten für das Anzeigen von Informationen sind möglich. Diese können aufeinander zugreifen, allerdings nicht auf den DOM von anderen Webseiten.
Um den Inhalt von einer Website manipulieren zu können, und genau das muss getan werden, wenn der Page-Designer erweitert werden soll, müssen sog. Content-Scripte genutzt werden. Mithilfe eines Content-Scripts wird JavaScript geladen, das ausgeführt wird, sobald die dafür angegebene Seite im Browser geladen wird. Dieses Content-Skript kann dann wiederrum Informationen als Nachrichten zwischen der geladenen Webseite und der Background-Page, also der Haupt-Logik der Extension, austauschen. Aus Sicherheitsgründen befinden sich das JavaScript der Website und das JavaScript der Extension in verschiedenen isolierten Umgebungen. So wird vermieden dass unsichere Websites sensible Daten im Browser auslesen können.

Unterschiede zwischen den verschiedenen Browsern
Bei der Entwicklung einer Erweiterung für mehrere Browser gibt es mittlerweile keine allzu großen Unterschiede mehr. Das liegt daran, dass die Browser Chrome, Edge, Firefox und Opera auf der Extension-API von Google-Chrome basieren. Chrome-Extensions können sich meist direkt im Opera-Browser installieren lassen. Für die Konvertierung zum Edge-Browser, hat Microsoft erst kürzlich das Microsoft Edge Extension Toolkit veröffentlicht, welches dabei hilft die nötigen Änderungen vorzunehmen. Und auch für die Portierung nach Firefox sind nur ein paar weniger Änderungen notwendig. Das ist natürlich auch immer abhängig von den verwendeten APIs und dem Support zwischen den Browsern.

Die eigene Browser-Extension: Die Idee

Um den Einsatzzweck einer Browser-Extension in Verbindung mit APEX zu zeigen wird der Code-Editor für JavaScript im Page-Designer um einen Code-Validator und Code-Beautifier ergänzt. Der Code-Editor für PL/SQL kann dies bereits und zeigt beim Klick auf den Validate-Button die aufgetretenen Fehler an.

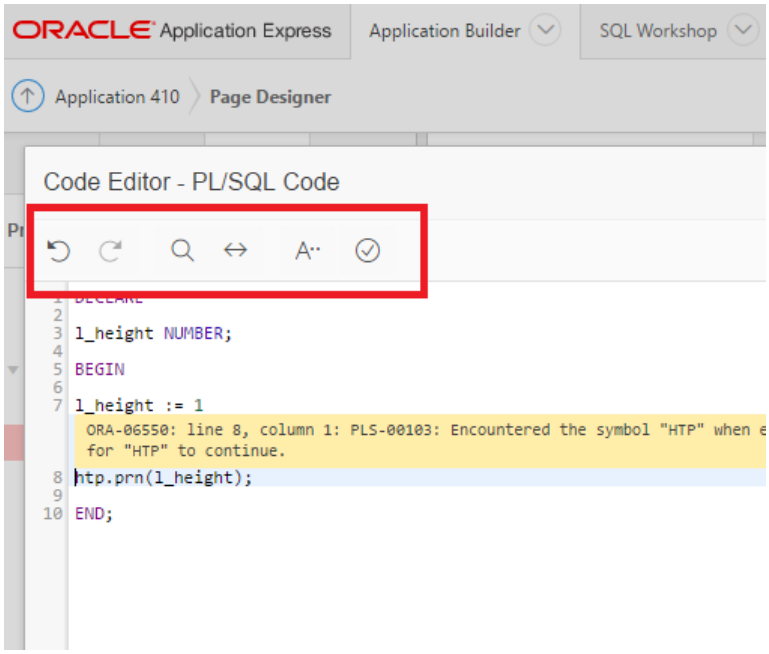

Abbildung 1: Code-Validation im APEX-Editor für PL/SQL-Code

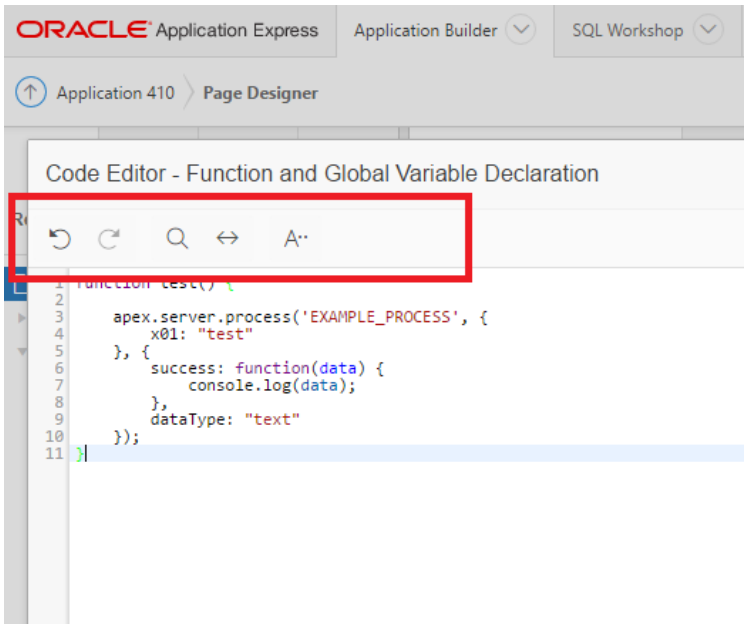

Abbildung 2: Keine Code-Validation für JavaScript-Code im Standard

Obige Screenshots zeigen den Unterschied zwischen dem PL/SQL und JavaScript-Editor. Die fehlende Funktionalität für JavaScript soll mit der in diesem Vortrag entwickelten Extension ergänzt werden. Um dies zu erreichen, braucht es neben einer Extension auch eine passende JavaScript-Bibliothek die diese Aufgabe für uns erledigt. Für das aufbessern der Syntax sowie das Validieren des JavaScript-Codes gibt es jeweils eine Open-Source Bibliothek, die über die Browser-Extension eingebunden und genutzt werden soll. Beide Ressourcen sind sowohl auf ihrer Website direkt nutzbar als auch auf GitHub für jeden verfügbar:

JSBeautifier

- http://jsbeautifier.org/
- https://github.com/beautify-web/js-beautify

JSHint

- http://jshint.com/
- https://github.com/jshint/jshint

JSHint verfügt dabei über verschiedene Parameter, die auf bestimmte Eigenschaften im Code prüfen. Einige Beispiele die sich an und abschalten lassen:

- Wurden Variablen oder Funktionen verwendet, aber nie benutzt?

- Wurden Blöcke zu tief verschachtelt? Die maximale tiefe lässt sich angeben

- Wurde ein Semikolon vergessen?

- Wurden Variablen benutzt die nicht initialisiert wurden?

Das folgende Bild zeigt einen einfachen Tippfehler, mit obigen Anwendungsfall, einer nicht initialisierten Variablen. Der Tippfehler in der Variable „height“ wird zu einem ungewollten Verhalten führen und lässt sich mit JSHint leicht aufdecken.

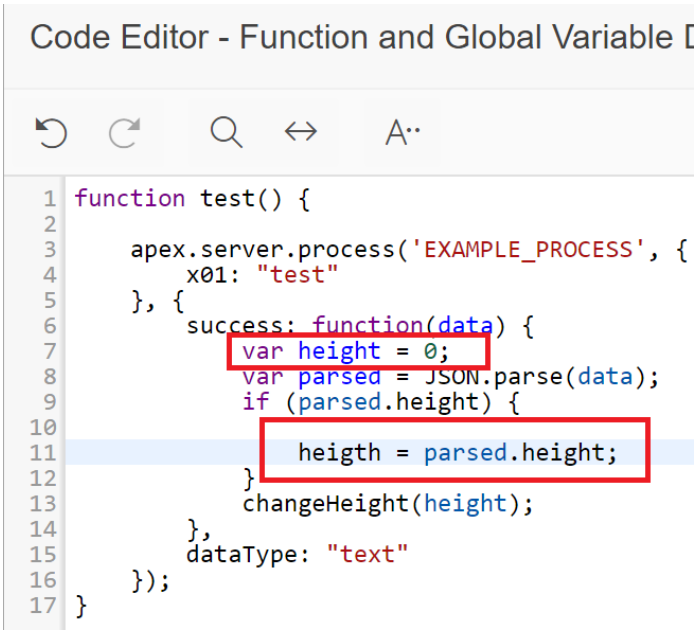

Abbildung 3: Ein kleiner Fehler mit großer Wirkung. Lässt sich mit JSHINT leicht aufdecken.

Die eigene Browser-Extension: Die Implementierung

Der Grundaufbau der Extension zum erweitern des Page-Designers ist sehr einfach. Es werden keine Background-Pages oder viele Funktionen der Extensions-API benötigt, was auch die Implementierung für andere Browser einfacher macht. Es wird im Beispiel eine Erweiterung für den Chrome-Browser entwickelt. Dazu wird zunächst ein Ordner für die Grundstruktur des Projektes erstellt. Dieser enthält folgende Unterordner und JavaScript-Bibliotheken, sowie eine Manifest-Datei, welche die genutzten Ressourcen, Berechtigungen und Eigenschaften auflistet.

- **Projektordner\js\jsbeautifier**

  Enthält die JavaScript-Bibliothek JSBeautifier

- **Projektordner\js\jshint**

  Enthält die JavaScript-Bibliothek JSHint

- **Projektordner\js\apex_validator.js**

  Enthält die eigene JavaScript-Logik. Fügt zusätzliche Buttons zum Aufrufen der Validierung in den Code-Editor von APEX ein. Steuert das Verhalten von JSBeautifier und JSHint, liest den Code aus dem JavaScript-Editor aus und übergibt ihn an beide Bibliotheken. Darüber hinaus wird eine eigene UI erstellt um die Parameter für JSHint steuern zu können. Mehr Details dazu im Vortrag.

- **Projektordner\src\inject\inject.js**

  Als Content-Script dient diese Datei zum Einfügen der eigenen JavaScript-Logik und der externen Bibliotheken JSHint und JSBeautifier in den DOM des Page-Designers. Folgender Codeabschnitt zeigt die Vorgehensweise:

```
 //Erstelle ein neues Script-Tag für die eigene JavaScript-Logik
var l_script   = document.createElement("script");
l_script.type  = "text/javascript";
//Erzeuge den Pfad zur Datei im Projektordner
l_script.src   = chrome.runtime.getURL('/js/apex_validator.js');
// Füge den Script-Tag in den DOM ein
document.body.appendChild(l_script);
```

- **Projektordner\manifest.json**

  In der Manifest-Datei sind die Eigenschaften der Browser-Extension definiert. Die wichtigsten Punkte sind dabei web_accessible_ressources, welches die JavaScript-Ressourcen angibt, die im Kontext einer Website benutzt werden dürfen, sowie content_scripts, welches angibt, für welche Website welches Content-Script geladen wird.

```
{
  "name"             : "JavaScript-Validation-Extension for APEX5.0",
  "version"          : "1.0",
  "author"           : "Till Albert",
  "manifest_version" : 2,
  "description"      : "Enhance the APEX 5.0 Page-Designer…",
  "web_accessible_resources" :
["js/apex_validator.js","js/jsbeautifier/lib/beautify.js","js/jshint.
js","js/jquery.ui.tabs.min.js"],
  "content_scripts": [
    {
      "matches": [ "*://*/*f?p=4000:*"    ],
      "js"     : [ "src/inject/jquery-
2.2.3.min.js","src/inject/inject.js"]
    } ]
```

Die Source-Files von APEX

Um die UI vom Page-Designer auch für die eigenen Elemente der Erweiterung nutzen zu können, lohnt sich ein Blick in die Source-Files von APEX. Diese bieten einige Widgets und Ideen die benutzt werden können. So zum Beispiel das Widget PropertyEditor, welches für die Auswahl der Optionen für JSHint benutzt wurde. Im Page-Designer ist dieses Widget fast immer präsent.

Die Source-Files befinden sich unter

- **Installations_Verzeichnis_APEX\images\libraries\apex**
  Diese Komponenten kennt man vom Entwickeln mit APEX5 bereits.
- **Installations_Verzeichnis_APEX\images\apex_ui\js**
  Die meisten dieser Komponenten können nicht genutzt werden, bieten aber eine Interessante Quelle um sich Anregungen und Ideen zu holen. Enthält z.B. den PropertyEditor, wie folgt im Einsatz in der Extension zu sehen.

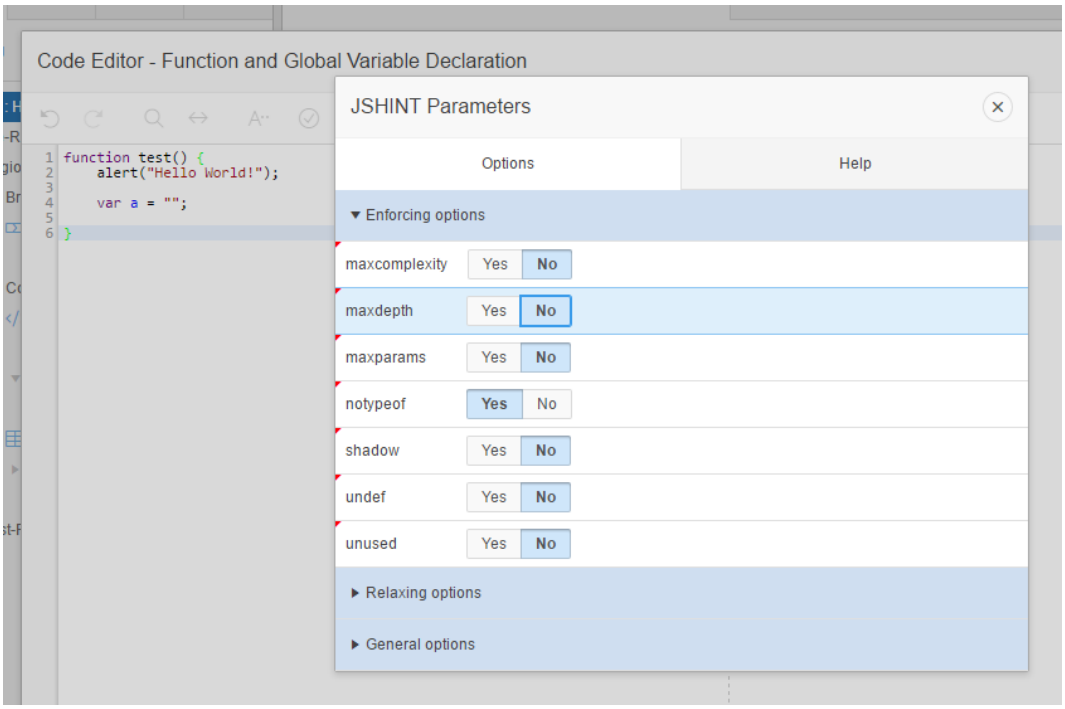

Abbildung 4: Einige Parameter für JSHINT, auswählbar in einem Property-Editor von APEX5.

Ausblick

Eigene Ideen lassen sich mithilfe von Browser-Extensions sehr einfach in das bestehende User-Interface von APEX integrieren. Die im Vortrag demonstrierte Demo soll zu eigenen Ideen anregen und veranschaulicht wie mit den Board-Mitteln von APEX ein praktischer Anwendungsfall integriert wurde.

Kontaktadresse:
Till Albert
MT AG
Balcke-Dürr-Allee 9
D-40882 Ratingen

Telefon:	+49 2102 30961-0
Fax:	+49 2102 30961-101
E-Mail	till.albert@mt-ag.com
Internet:	http://www.mt-ag.com/