search box 2.0

Handbuch

jQuery Plugin - Erweiterte Autovervollständigung auf der Basis von SPARQL Endpunkten

Inhaltsverzeichnis

1 Einleitung

Das Ziel des Projektes ist es beliebige SPARQL-Endpunkte zu durchsuchen und anhand von wenigen vorhandenen Informationen durch die RDF-Graphen, anhand der verlinkten Daten, letztlich zum gesuchten Ziel zu navigieren. So soll eine erweiterte Autovervollständigung für eine Suche nach Ressourcen ermöglicht werden. Primär soll das Plugin im OntoWiki zum Einsatz kommen, aber auch das Einbinden in andere Umgebungen ist problemlos möglich.

2 Nutzung des Plugins

1. Herunterladen des Plugins

Das Plugin ist von der Projektseite herunterzuladen und mittels einem geeigneten Dekomprimierer wie WinZip, WinRAR oder 7Zip zu entpacken.

2. Einbinden des Plugins

Soll das Plugin in das OntoWiki integriert werden, so wird eine korrekte Installation von OntoWiki vorausgesetzt.

Das Einbinden der Javascriptdatei jquery.searchbox.js geschieht durch Einfügen in den Header der verwendeten html-Datei.

Hierzu muss folgende Zeile hinzugefügt werden:

```
<script type="text/javascript" src="(your source)/jquery.searchbox.js" />
```

Desweiteren muss das Stylesheet des Plugins eingebunden werden:

```
<link rel="stylesheet" type="text/css" href="(yoursource)/jquery.searchbox.css">
```

Um Searchbox 2.0 korrekt nutzen zu können ist dies unumgänglich.

Voraussetzung für das Plugin ist eine aktuelle Version des jQuery Frameworks und jQueryUI.

3. Anwenden des Plugins

Benötigt wird ein Inputfeld mit beliebiger ID , auf das die searchbox-Funktion angewendet wird. Dies geschieht folgendermaßen:

```
<input type="text" id="input" />
```

„$('#input').searchbox({<options>});“

3 Funktionalitäten

Folgende Funktionen stehen zur Verfügung:

- einen oder mehrere SPARQL Endpunkte definieren

  Dem Widget können mehrere SPARQL - Endpunkte als Parameter übergeben werden, an die die Suchanfrage parallel gestellt wird und deren Rückgabewerte asynchron in der Ergebnisliste dargestellt werden

- Definition von Facetten

  Der Nutzer kann die Darstellung der Ergebnismenge durch eine GUI basierte Definition von Facetten filtern und reduzieren;dabei können schrittweise auf die Ergebnismenge angepasste Facetten gewählt werden.

- Anpassung der Parameter

  Mit der Verwendung des Widgets erhält der Administrator die Möglichkeit durch Überschreiben von vordefinierten Default- Werten das Plugin genauer anzupassen

- Anzahl der gelisteten Elemente aus der Ergebnismenge festlegen

  Es lässt sich eine Obergrenze für die Zahl der angezeigten Ergebnisse festlegen.

- CSS-Styling des Widgets

  Die DOM-Elemente des Widgets sind als CSS-Klassen realisiert und können mit eigenen Style-Definitionen versehen werden.

- Lokalisierung

  Mit Hilfe des Parameters "strings" lassen sich die einzelnen Textstellen in eine beliebige Sprache übersetzen.

- Query exportieren

  Benutzer mit installiertem Firebug oder einem anderen Javascript-Debugger haben die Möglichkeit sich alle gesendeten Queries in Echtzeit anzuzeigen.

Hinweis :

Eine Einschränkung ist derzeit, dass das Plugin auf Grund der Same Origin Policy nur Endpunkte abfragen kann, die aus der selben Quelle wie der Einsatzort des Plugins stammen.

4 Parameter

4.1 Standardparameter

Name	Beschreibung	Beispiel
`endpoints`	Array von Endpunkten	["http://catalogus-professorum.org/sparql", "http://od.fmi.uni-leipzig.de/sparql"]

4.2 optionale Parameter

Name	Beschreibung	Standardwert
`limit`	Limit für SPARQL-Queries / maximale Anzahl der ResultElemente	`limit: 10`
`shownFacets`	maximale Anzahl der gezeigten Facetten	`shownFacets: 6`
`inputChars`	Anzahl der eingegebenen Zeichen, ab denen die Suche beginnt	`inputChars: 3`

`delay`	Verzögerung nach dem Eintippen bis zum Starten der Suche in ms	`delay : 750`
`strings`	Anzeige der Beschriftungen	`strings: { facetString : 'Facets', endpointString: 'Endpoints', resultString: 'Results', endpointErrorString: 'Error: You have to select at least one endpoint.', facetCountString: 'result elements have this property.', noElementsString: 'no elements to display' }`

4.3 Callbacks

Name	Beschreibung	Beispiel
`startCallback`	Diese Funktion wird vor dem Absenden des Queries zum Empfangen der Ergebnisse, ausgeführt.	`StartCallback : 'function() { alert("start"); }'`
`stopCallback`	Die Funktion wird nach dem Empfangen der Ergebnisse ausgeführt.	`StopCallback : 'function() { alert("stop"); }'`
`queryCallback`	Diese Funktion ersetzt komplett den Query zum Empfangen der Ergebnisse. Sie muss den Query-String zurückliefern und erhält ein Array von Zusatzklauseln (Strings) , entstehend aus den Facetten.	`queryCallback : 'function(optQueries) { return "SELECT * WHERE {?s ?p ?o}"; }'`
`resultClickCallback`	Mit dieser Funktion kann das Standardverhalten beim Anklicken eines Ergebnisses überschrieben werden.	`resultClickCallback : 'function() { alert("clicked"); }'`

`popupGuiCallback`	Diese Funktion überschreibt die Standard HTML-Struktur des Plugins. Die Funktion muss einen String mit der entsprechenden Struktur zurückliefern.	`popupGuiCallback : function(){ return " <div id="endpoints"></div> <div id="facets"></div> <div id="results"></div>"; }'`
`resultsSortCallback`	Diese Funktion überschreibt die standardmäßige, alphabetische Sortierung der Ergbenisse. Bekommt jeweils 2 ResultElement-Instanzen geliefert.	`resultsSortCallback : 'function(a,b){ if (a.object.length < b.object.length) return -1; if (a.object.length > b.object.length) return 1; return 0; }'`
`resultsOutputCallback`	Diese Funktion überschreibt die standardmäßige Anzeige der Ergebnisse.	`resultsOutputCallback: 'function(){ for (var i = 0; i < this.elements.length; i++) $('#results').append( "Ergebnis"); }'`

5 sonstige Hinweise

Als SPARQL-Client wird folgendes externes Skript automatisch eingebunden: http://www.thefigtrees.net/lee/sw/sparql.js

Beim Absenden des Eingabefeldes, auf das das Plugin angewendet wurde, wird automatisch die URI der gewählten Ressource in dem hidden Input-Feld mit name="URI" mitgeschickt.

6 Mitwirkende

Das jQuery Plugin „Searchbox 2.0“ wurde im Rahmen des Moduls Softwaretechnik an der Universität Leipzig entwickelt.
Mitgearbeitet haben:

Projektleitung	Marcus Nitzschke
Technischer Assistent	Clemens Hoffmann
Recherche	Thomas Schöne
Modellierung	Marcus Nitzschke, Thomas Schöne
Testing	Konrad Baumheier
Implementierung	Clemens Hoffmann
Dokumentation & Qualitätssicherung	Marina Mitjagin

Bildnachweise

Logo: Projektgruppe swp10-7