Preis-Crawler

Aus IT Wiki

Inhaltsverzeichnis

Überblick

Dient zur Wettbewerbsanalyse und sammelt Preisinformationen von verschiedenen Konkurrenzseiten, Preisvergleichsseiten und unseren eigenen Online-Shops. Statt Preis-Crawler wird auch mal von Preis-Roboter gesprochen.

Ziel

- es sollen Produkte und Preise der wichtigsten Wettbewerber für das Segment Kontaktlinsen und Pflegemittel ermittelt werden
- mit Hilfe dieser Informationen wird die Sortimentsabdeckung und Positionierung bzgl. Verkaufspreise geprüft, chronologisch abgebildet und entsprechende Maßnahmen eingeleitet
- alle Wettbewerbsdaten sollen mittels Cronjob 1xtäglich bzw. 1x wöchentlich ermittelt werden und zur Auswertung als CSV-Export zur Verfügung gestellt werden
- das Crawlen der eigenen Seiten, dient zusätzlich zur Kontrolle unserer Preise

Hinweise

- Von dieser Datenerhebung ist keine weiteren Prozesse abhängig
- Treten Fehler durch Seitenstrukturänderung des Wettbewerbers auf, werden diese als "normal" eingestuft (nicht kritisch oder blocker) und entsprechend dieser Priorität behoben, dazu werden diese Fehler in einer gesonderten CSV gesammelt
- Ähnliche Funktionalität bieteten die Crawler für Amazon und MeinPaket. (amazonExport und meinPaketExport)

Daten

Folgende Daten (Spaltenangaben) sollen für alle Artikel der Kategorien Kontaktlinsen und Pflegemittel ermittelt werden:

- Zeitstempel (Datum, Uhrzeit)
- Seite (Seitenname)
- URL (vollständige Produkt-URL)
- Wettbewerber
- Status
- Gruppe
- Produkt-Nummer
- Produkt-Name
- Hersteller
- Artikel-Anzahl
- Preis
- Rabatt
- Versandkosten
- Packmenge (Inhaltsmenge)
- Anzahl verkauft
- Position (Ranking bei verschiedenen Anbietern auf einer Platform)

Zu crawlende Seiten

Seite	Crawlername	Aktualität	Priorität	Status
Lensspirit	lensspirit	Täglich	normal	fertig
LinsenQuelle	linsenquelle	Täglich	normal	fertig
ebay	ebay	Täglich	normal	fertig
Kontaktlinsenpreisvergleich	kontaktlinsenpreisvergleich	Täglich	hoch	fertig
Idealo	idealo	Täglich	hoch	fertig
Google Shopping	googleshopping	Täglich	hoch	fertig
Lensbest	lensbest	Wöchentlich	normal	fertig
Mr.Spex	mrspex	Wöchentlich	normal	fertig
Linsenplatz	linsenplatz	Wöchentlich	normal	fertig
Linsensuppe	linsensuppe	Wöchentlich	normal	fertig
Linsenpate	linsenpate	Wöchentlich	normal	fertig
321 Linsen		Wöchentlich	gering	
Brille24		Wöchentlich	gering	
MeineLinse		Wöchentlich	gering	
Bestpricelens		Wöchentlich	gering	
discountlens.de		Wöchentlich	gering	
Amazon Seller Central	amazon	nach Bedarf	normal	fertig

Nutzung

Die Crawler werden über Jenkins aus dem LS Netzwerk gesteuert:

```
http://dev.ls.local:8080/view/Preis-Crawler/
```

Dazu wurden mehrere Jenkins-Jobs angelegt:

- Preis-Crawler Lokal Täglich
- Preis-Crawler Lokal Wöchentlich
- Preis-Crawler Lokal Amazon
- Preis-Crawler Lokal (nur für Tests)
- Preis-Crawler Lokal (nur für Tests)

Welcher Job welche Crawler ausführt wird über die jeweilige Job-Konfiguration gesteuert und orientiert sich an der oben stehenden Tabelle.

Crawler starten

Um einen Crawler bzw. Jenkins-Job manuel zu starten, muss der jeweilige Job aufgerufen werden. Z.B.:

```
http://dev.ls.local:8080/view/Preis-Crawler/job/PC_local_amazon/
```

Im linken Menü dient die Option **Jetzt bauen** zum Starten des Jobs.

In der darunter stehenden Tabelle (Build Verlauf) ist ersichtlich ob ein Job bereits läuft oder nicht. Durch anklicken eines solchen Builds kann man mehr Informationen zu diesem Durchlauf erhalten. Die meisten Informationen bringt der Menü-Punkt **Console Output** über den auch im Fehlerfall ersichtlich ist, welche Probleme es gab.

Crawler beobachten

Um einen Crawler bei der Arbeit zu beobachten, muss eine Remotedektopverknüfung aufgebaut werden. Da die Selenium-Server je nach Auslastung entscheiden, wer welchen Job übernimmt, muss ausprobiert werden, auf welchem Server der Crawler gerade läuft.

Dazu einfach per Remotedektopverknüfung mit folgenden Servern verbinden und das **Login to xrdp** mit **OK** bestätigen:

- selenium01
- selenium02

Umsetzung

- Die Umsetzung erfolgt mittels Selenium auf unseren Selenium-Servern
  - für den Live-Betrieb gibt es zwei Jenkins-Jobs, in die nach obiger Tabelle die Crawler nach Aktualität eingeordnet sind
    - "Price-Crawler Local Täglich" (PC_local_daily), läuft jede Nacht zwischen 3 und 4Uhr
    - "Price-Crawler Local Wöchentlich" (PC_local_weekly), läuft jeden Montag zwischen 3 und 4Uhr
  - es existiert ein separater Jenkins-Job "Price-Crawler Local" (PC_local) für Tests
  - für die Programmierung kann sich jeder Entwickler einen eigenen Job anlegen z.B. "Price-Crawler Paul" (PC_paul)
  - für jeden Job kann über Jenkins konfiguriert werden, welcher Crawler genau laufen soll
- Der Preis-Crawler ist ein eigenständiges Projekt und wird im Repository "price-crawler"

versioniert
    - das Repository inkludiert genau wie die Online-Shops das "core"-Verzeichnis
- wenn es möglich ist wird das Artikelsortiment automatisch gecrawlt, alternativ kann auch eine Konfigurations-Datei hinterlegt werden die eine Liste von Links zu den entsprechenden Artikeldetailseiten oder z.B. Suchbegriffe enthält

Dateipfade:

Alle Daten werden in dem jeweiligen Jenkins-Job-Verzeichnis hinterlegt:

```
\\DEV\www\deploy.dev.ls.local\jenkins-jobs\PC_local
\\DEV\www\deploy.dev.ls.local\jenkins-jobs\PC_local_daily
\\DEV\www\deploy.dev.ls.local\jenkins-jobs\PC_local_weekly
```

Die Konfigurationsdateien werden hier abgelegt und nach dem entsprechenden Crawler bezeichnet:

```
crawler\config
```

Die Ergebnis- und Fehlerdateien werden nachfolgendem Schema benannt:

```
<Datum>_<fortlaufende Nummer>_result.csv
<Datum>_<fortlaufende Nummer>_error.csv
```

Die Ergebnisse werden direkt hier her geschrieben:

```
crawler\result
```

Nach jedem Crawl wird die Ergebnisdatei zusätzlich hier her kopiert:

```
crawler\export
```

Wenn Parsing-Fehler auftreten, dann werden diese hier her geschrieben:

```
crawler\error
```

Programmierung

Der Einstiegspunkt ist wie auch bei den Selenium-Tests eine "bootstrap.php" die im Root-Verzeichnis des Projekts liegt. Diese läd alle Konfigurationen und unser Framework, damit auch unsere Klassen aus dem Core zur Verfügung stehen.

In dem Hauptverzeichnis sind zwei abstrakte Klassen "crawler" und "crawlerPage" hinterlegt. Diese enthalten die meiste Logik und grundlegende Funktionenm welche von den Crawler spezifischen Klassen geerbt werden.

- *crawler.php*: erbt Funktionalitäten von class_seleniumTestCase und stellt Methoden zum crawlen also einsammeln und aufrufen von Links zur Verfügung
- *cralwerPage.php*: enthält Funktionen zum Parsen einer aufgerufen Seite, um entsprechende Inhalte zu extrahieren

Im Unterverzeichnis *website* erhält jede zu crawlende Seite ihr eigenens Verzeichnis mit entsprechenden Dateien:

- *crawler.php*:
  - erbt vom übergeordneter *crawler.php*
  - enthält Grundvariablen-Definitionen
  - enthält die Testfunktion, welche den Einstieg für Selenium darstellt
  - kann individuelle Funktion zum Beispiel zum Crawlen von Kategorieseiten enthalten
- *page/xxx.php*:
  - erbt vom übergeordneter *crawlerPage.php*
  - enthält Definitionen für Elemente auf der entsprechenden Seite: z.B. auf Kategorie-Seiten Identifier in Form eines XPaths zu den jeweiligen Artikel-Links
  - kann individuelle Funktionen enthalten um zusätzliche dynamisch Daten auf dieser Seite zu finden, wie Angaben zu verschieden Packmengen oder Preise unterschiedlicher Anbieter

---