Burning-Seri.ES API – Dokumentation

Grundinformationen über die API

Die API von Burning-Seri.es ist in erster Linie deswegen vorhanden, um die Android-App anbieten zu können. Das heißt, dass die API normalerweise nicht dafür gedacht ist, um beliebige drittanbieter abzudecken. Anfragen bezüglich der Erweiterung der API werden ggf. nicht beachtet.

Die API ist direkt via GET zu erreichen. Dabei ist aktuell die Verbindung zum Server mit GZip komprimiert. Dies muss ggf. in diversen Programmiersprachen beachtet werden.

Als Rückgabewert erhält man stets einen JSON-Wert.

Login und Sessions

Die API unterstützt den Login mit Benutzernamen und Kennwort. Dies ist jedoch nur möglich, wenn entsprechender Benutzer auch einen Account auf bs.to besitzt.

Die Session wird nicht über ein Cookie weitergereicht, sondern über einen GET-Parameter, welcher die Session-ID beinhaltet.

Nutzung der API über PHP und dessen Nachteile

Es ist möglich, die API via PHP abzurufen. Dies bietet jedoch den gewaltigen Nachteil, dass, wenn das PHP-Script von einem Server ausgeführt wird, dessen IP an die Server von bs.to weitergeleitet und nicht die IP des eigentlichen nutzers. Dies kann dazu führen, dass eventuell die Session ungültig wird.

Auch ist es über PHP nur sehr schwer mögich, die entsprechenden Hoster auszulesen bzw. zu Parsen.

Abfragen

Serienliste

Die Serienliste ist das Erste, was bisher in jeder Anwendung abgefragt wird. Diese Liste beinhaltet jede Serie und jede ID zur entsprechenden Serie, welche man auf bs.to finden kann. Der Rückgabewert ist selbst mit einer Gzip-Kompression noch relativ groß und sollte daher idealerweise gecached werden, um dem Server unnötiges Neuladen der Serienliste zu ersparen.

Der Abfrage-Link lautet:

> http://bs.to/api/series/

Ein Ausschnitt aus dem Rückgabewert, welcher bei dieser Abfrage zurückgegeben wird:

```
[{"series":".hack\/\/Legend of the Twilight","id":"757"},
{"series":".hack\/\/Roots","id":"807"},
{"series":".hack\/\/Sign","id":"1"}, ... ,{"series":"Zwei Singles
im Doppelbett","id":"2828"}]
```

Der Aufbau beruht auf einem Array mit beliebig vielen Elementen. Jedes Element ist ein Objekt mit „series“ und „id“. Der Wert „series“ beinhaltet den Namen der Serie, wobei „id“ die entsprechende ID der Serie beinhaltet. Diese wird benötigt, um weitere Informationen sowie Staffeln und Episoden der Serie auszulesen. Im weiteren Verlauf wird diese mit <ID> angezeigt.

Eine Strukturierte Darstellung eines Eintrages:

```
Object {
      series => ".hack\/\/Legend of the Twilight",
      id => "757"
}
```

Auf Basis der ID kann nun eine Abfrage zur entsprechenden Serie durchgeführt werden. Im Beispiel nutzen wir diesbezüglich die Serie mit der ID 60. Dabei handelt es sich um „Die Simpsons“.

Nutzt man die ID und sortiert diese absteigend (50, 49, 48 ...), kann man so sehen, welche Serie zuletzt hinzugefügt wurde (höchste ID bedeutet, dass diese Serie zuletzt hinzugefügt wurde). Einen Zeitstempel diesbezüglich gibt es jedoch nicht.

Abfragen

Serie

Um eine Serie abzufragen, müssen Sie lediglich die <ID> an den Serienabfrage-Link anhängen und dazu den Wert 1, um die erste Staffel (welche ausnahmslos jede Serie besitzt) anhängen.

Der Abfrage-Link lautet:

> http://bs.to/api/series/<id>/1/

Ein Ausschnitt aus dem Rückgabewert, welcher bei dieser Abfrage zurückgegeben wird:

```
{"series":{"id":"60","series":"Die Simpsons","url":"Simpsons", ... ,
"season":1,"epi":[{"german":"Es Weihnachtet Schwer","english":"Simpsons
Roasting on an Open Fire","epi":"1","watched":"0"}, ... ]}
```

Wie man hier sieht, erhält man kein Array zurück, sondern direkt ein Objekt. Hierbei ist die Struktur schon etwas umfangreicher und ohne bekannte Struktur auch schwerer zu erkennen. Daher hier direkt die komplette mögliche Auflistung des Aufbaus:

```
Object {
      series => Object {
            id => "60",
            series => "Die Simpsons“,
            url => "Simpsons“,
            description => “Die von Matt Groening geschaffenen ...“,
            start => “1989“,
            end => null,
            movies => 1,
            seasons => 26,
            data => Object {
                  producer = Array ( ... ),
                  actor = Array ( ... ),
                  director = Array ( ... ),
                  genre = Array ( ... ),
                  genre_main = Array ( ... )
            }
      }
      season => 1,
      epi => Array (
            Object {
                  german => “Es Weihnachtet Schwer“
                  english => “Simpsons Roasting on an Open Fire“
                  epi => 1,
                  watched => 0
            }
      )
}
```

Auslesen der Serieninformationen

Fangen wir bei dem ersten Wert an. Dieser befindet sich im Objekt „series“ unter dem namen „id“ wieder (Object->series->id) und beinhaltet erneut die ID der Serie.

Als nächstes folgt der Wert „series“ (Object->series->series), welches den Namen der Serie beinhaltet.

„url“ (Object->series->url) beinhaltet den Wert, welcher nötig ist, um die URL zur Seite (bs.to) zur entsprechenden Serie zu erzeugen. In diesem Falle würde der Link folgendermaßen lauten: http://bs.to/serie/Simpsons

Der Wert „description“ (Object->series->description) beinhaltet die Beschreibung der Serie, wie sie auch auf bs.to zu finden ist. Nicht zu verwechseln mit den einzelnen Episoden-Beschreibungen.

Die nächsten 2 Werte sind „start“ und „end“ (Object->series->start / Object->series->end), welche den Serienstart (start) und das Serienende (end) zurückgeben. Sollte die Serie noch nicht abgesetzt oder beendet worden sein, steht bei „end“ der wert „Null“ drin.

Der Wert „movies“ (Object->series->movies) beinhaltet entweder den Wert „1“, wenn Filme zu der Serie existieren, ansonsten „0“.

Der Wert „seasons“ gibt an, wieviele Staffeln eine Serie besitzt. Das heißt in diesem Falle, würde die letzte Staffel, welche man auslesen kann, die 26te Staffel sein. Der Abfrage-Link würde folgendermaßen aussehen: http://bs.to/api/series/60/26/
Dabei würde der Aufbau des Rückgabewertes identisch mit diesem gerade beschriebenem Aufbau sein.

Unter dem Wert „data“ (Object->series->data->...) findet man Produktions-Informationen wie z. B. die Namen der Produzenten. Auch kann man darüber die Genre-Liste und das Haupt-Genre auslesen.

Der Wert „season“ (Object->season) gibt zurück, welche Staffel das folgene Array beinhaltet.

Der Wert „epi“ (Object->epi) beinhaltet die einzelnen Episoden der gewählten Staffel. Dabei ist der deutsche sowie der englische Name angegeben. Sollte der Wert „german“ leer sein, gibt es diese Episode nicht mit deutscher Synchronisation.

Der Wert „epi“ (Object->epi[???]->epi) beinhaltet die Episodennummer. Da diese fortlaufend (von 1) ist, ist dieser Wert nur zur Überprüfung vorhanden. Um die Episode 5 aus dem Array auszulesen, greift man auf den Index 4 des Arrays zurück. Jedoch ist die Episodennummer 5.

Zu guter Letzt ist der Wert „watched“ (Object->epi[???]->watched) da, um zu zeigen, ob der bereits eingeloggte Benutzer diese Episode bereits gesehen hat. Diese wird immer „0“ ergeben, solange kein Benutzer angemeldet ist oder die Serie noch nicht mit der Watch-Abfrage oder über die Webseite gesehen wurde.

Staffel und Filmlisten

Es gibt einen Unterschied zwischen einer Staffel und der Liste der Filme. Denn jede Staffel wird mit der entsprechenden Nummer abgerufen. bs.to/api/series/<ID>/<Staffel>/, wobei die 1 als Wert auch die 1te Staffel zurückgibt. Nun stellt sich die Frage, wie man an die Filme herankommt, wenn der wert „movies“ gleich 1 ist.

Dies ist besonders einfach, da man hierfür anstelle der Staffelnummer nur eine 0 angeben muss. Schon erhält man auf Basis des Aufbaus einer Staffel die vorhandenen Filme.

Ein Beispiel sähe so aus:

http://bs.to/api/series/60/0/

Einzelne Episoden

Um eine einzelne Episode (oder Film) auszulesen, bedarf dies wiederum wenig Umdenken bezüglich der URL. Diese wird einfach um den „epi“-Wert aus vorheriger Staffel-Information erweitert.

http://bs.to/api/series/60/1/1/

Als Rückgabewert erhält man in diesem Falle etwa Folgendes:

```
{"series":"Die Simpsons","epi":{"german":"Es Weihnachtet
Schwer","english":"Simpsons Roasting on an Open
Fire","description":" ... ","id":"17296"},"links":
[{"hoster":"CloudTime","part":"1","id":"441544"},
{"hoster":"FileNuke","part":"1","id":"1430229"}, ...]}
```

Wie man sehr gut erkennen kann, beginnt das ganze wieder mit einem Objekt. Hier eine aufgelistete Anordnung:

```
Object {
     series => "Die Simpsons",
     epi => Object {
          german => "Es Weihnachtet Schwer",
          english => "Simpsons Roasting on an Open Fire",
          description => " ... ",
          id => "17296"
     },
     links => Array (
          Object {
               hoster => "CloudTime",
               part => "1",
               id:"441544"
          },
          ...
     }
}
```

Wie man sieht, ist es wieder recht einfach gehalten. Der Wert „series“ (Object->series) beinhaltet den Namen der Serie.
Direkt gefolgt kommt der Wert „epi“ (Object->epi), welcher gefüllt ist mit dem englischen und deutschen Namen sowie der Episodenbeschreibung (Object->epi->description). Die „id“ (Object->epi->id) ist die explizite ID in der Datenbank von bs.to, unter welcher diese Episode gespeichert ist. Anhand dieser ID kann man auch die vorhandenen Episoden sortieren und so erkennen, welche Episode zuletzt hinzugefügt wurde.

Der nächste Wert wäre „links“ (Object->links), welcher die verfügbaren Hoster in einem Array aufzählt. Als Beispiel steht dort „CloudTime“ (Object->links[???]->hoster). Dieser beinhaltet noch den Wert „part“ (Object->links[???]->part), welches den Part, also das Teilstück der Episode bei dem Hoster bestimmt. Der letzte Wert „id“ (Object->links[???]->id) ist die eigentliche ID zur Abfrage des eigentlichen Hoster-Links. Zu diesem kommen wir im nächsten Teil der Dokumentation.

Hoster einer Episode erhalten

Um den korrekten Link eines Hosters einer Episode zu erhalten, ist nun nur noch der Wert „id“ aus vorherigem Abschnitt der Hoster-Liste nötig. Als Beispiel dient uns der CloudTime-Hoster der Episode 1 aus Die Simpsons Staffel 1. Der Wert lautet “441544“.

Die Abfrage-URL für einen Link einer Episode lautet:

> http://bs.to/api/watch/<ID>/

Der Rückgabewert würde folgendermaßen aussehen bei der ID 441544:

```
{"hoster":"CloudTime","url":"51cefb8b8a979","part":"1","epi":
"17296","fullurl":"http:\/\/www.cloudtime.to\/video\/51cefb8b
8a979"}
```

Man sieht direkt, dass es neben „url“, welche nur den wichtigsten Teil der URL beinhaltet auch den Wert „fullurl“ gibt, welcher den gesamten Link beinhaltet. Dazu enthält es noch den Wert „hoster“, welcher nochmal den Namen des Hosters zurückgibt, sowie „part“ und „epi“. Wobei „epi“ die Episoden-ID zurückgibt.

Somit wäre man bereits am Ziel der Abfragen angelangt.

Weitere Abfragen

Login und Logout

Um einen Login durchzuführen, benötigt man erst einmal die Zugangsdaten des Benutzers. Als nächses sendet man diese via POST-Abfrage an http://bs.to/api/login/.

Dabei werden die Daten mit folgenden Namen via POST übermittelt:

```
login[user] = USERNAME
login[pass] = PASSWORT
```

Es ist hierbei egal, in welcher Reihenfolge die Daten übermittelt werden.

Als Rückgabewert erhält man entweder einer Fehlermeldung:

```
{"error":"Username oder Passwort falsch"}
```

Oder eine Session-ID:

```
{"user":"Droggelbecher","session":"da159710fd72aa871b6bddc1ff
63da31"}
```

Wie man sieht, erhält man hierdurch auch den exakten Usernamen zurück. Der Wert „session“ sollte gespeichert/gemerkt werden, da mit diesem der User sich ausweisen wird. Dies geschieht durch einen einfachen Get-Parameter an jeder API-Abfrage.

```
?s=da159710fd72aa871b6bddc1ff63da31
```

Gibt man diesen Wert bei /api/series/<ID>/<Staffel>/ mit an, wird entsprechend auch jeweils der „watched“-status 1 oder 0 ergeben, insofern der User die Episoden bereits gesehen hat.

Ein Logout ist einfach über die URL http://bs.to/api/logout/?s=<SESSIONID> möglich. Es wird empfohlen, diese Methode zu nutzen, da sonst die Session-ID weiterhin gültig ist und eventuell von Hackern mißbraucht werden könnte.

Zurücksetzen des „Gesehen“-Status einer Episode

Sollte man eine Episode mit /watch/ abfragen, aber die entsprechende episode bzw. der hoster ist ungültig, kann man den Gesehen-Status zurücksetzen mittels der selben Abfrage, jedoch mit /unwatch/.

Einschränkungen

Die API kann nicht alle Funktionen abdecken und wird dies auch nie. Denn sie ist eigentlich wie bereits erwähnt dazu gedacht, die Android-App zu bedienen und mit Daten zu versorgen.