
Coinmarketcap
via Jsoup und BSF4ooRexx

Agenda

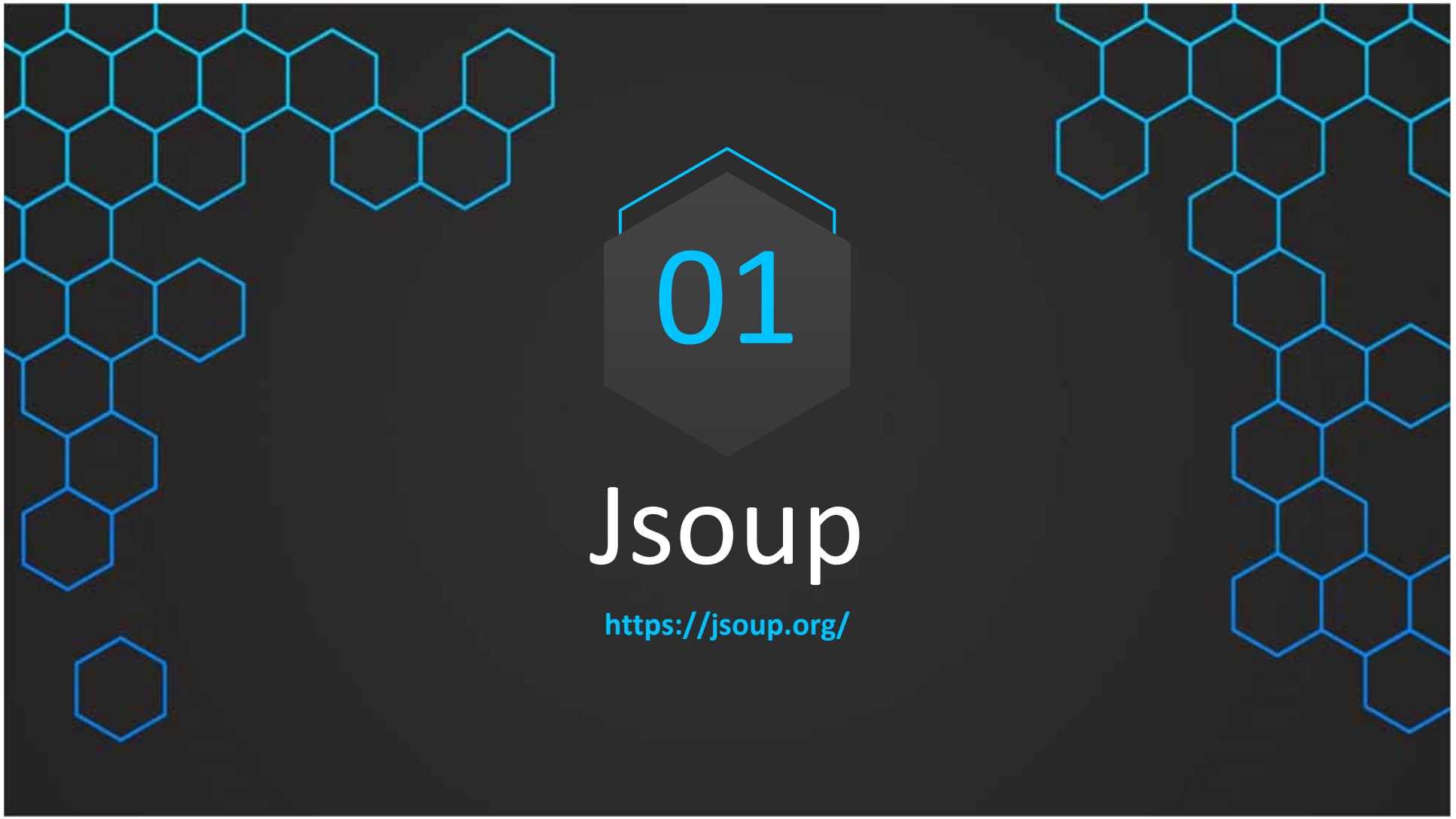
01 **Jsoup**
Kurzer Überblick über Jsoup Library

02 **SQLite**
Kurzer Überblick über SQLite-jdbc Library

03 **Funktionen**
Ausgewählte Codezeilen

04 **Challenges**
Daten aus Variablen in DB schreiben

05 **Mögliche Erweiterungen**
Was wäre noch möglich?


01
Jsoup
https://jsoup.org/

Jsoup 1/3

Jsoup ist ein Open Source Java Library zum

- Auslesen
- Manipulieren

von HTML Code von Webseiten.

Jsoup 2/3

Code:

```
doc1=bsf.import("org.jsoup.Jsoup")~connect("https://www.wu.ac.at/")~get
title = .bsf~new("org.jsoup.select.Elements")
title = doc1~title
say "Titel der Website:" title
```

Ausgabe:

```
Titel der Website: Wirtschaftsuniversitõt Wien: WU (Wirtschaftsuniversitõt Wien)
```

Jsoup 3/3

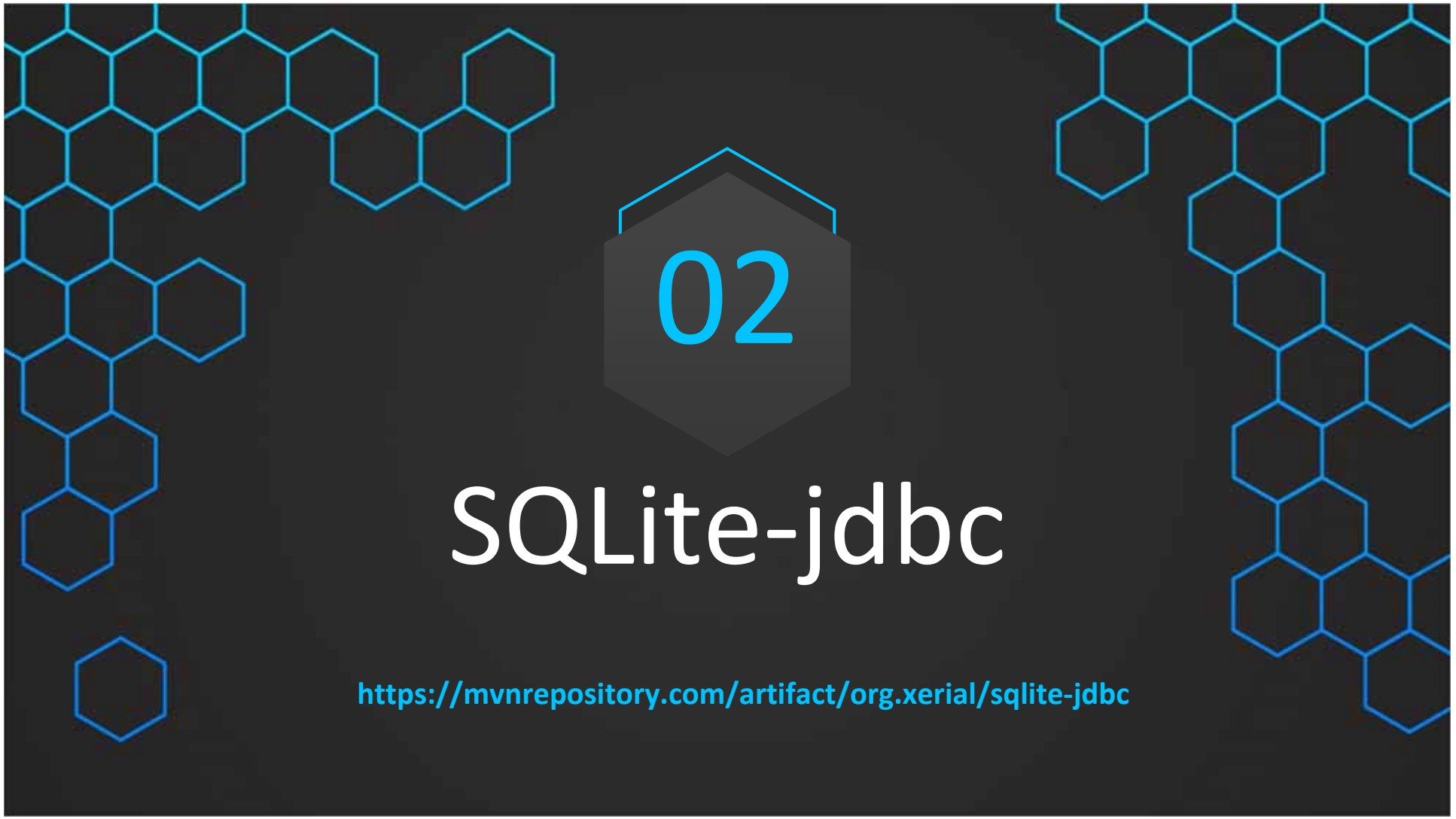
Code:

```
doc1=bsf.import("org.jsoup.Jsoup")~connect("https://www.wu.ac.at/")~get~html
say doc1
```

Ausgabe:

```
<!doctype html>
               <html dir="ltr" lang="de">
                                          <head>
                                                  <meta http-equiv="X-UA-Compatible" content="IE=edge">
                                                                                                        <meta ch
 website is powered by TYPO3 - inspiring people to share! TYPO3 is a free open source Content Management Framework
ht 1998-2021 of Kasper Skaarhoj. Extensions are copyright of their respective owners. Information and contribution

iversitõt Wien)</title>
```


02
SQLite-jdbc
https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc

SQLite-jdbc

SQLite-jdbc ist eine Java Library mit der es sehr einfach möglich ist, seine Daten in eine Datenbank auszulagern.

Es ist eigentlich keine Konfiguration notwendig, im Vergleich zu z.B. MySQL

SQLite-jdbc

„Konfiguration:“

```
url ="jdbc:sqlite:cryotohist.db"     ---> Speichert die db im selben ordner
mydriver=.bsf~new('org.sqlite.JDBC')
man=bsf.loadClass("java.sql.DriverManager")
man~registerDriver(mydriver)
conn=man~getConnection(url)
```

Anlegen von Tabelle:

```
query=statement~execute('CREATE TABLE IF NOT EXISTS history (id integer primary key , name text, price text);')
```


03
Funktionen

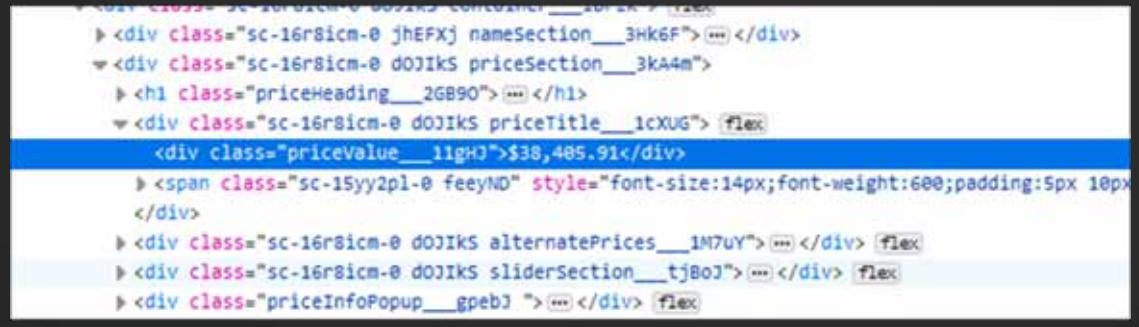
Funktionen

Auslesen der Daten

Darstellen der Daten

Abfrage-History

Auslesen der Daten:



Auslesen der Daten:

Code:

```
doc=bsf.import("org.jsoup.Jsoup")~connect("https://coinmarketcap.com/currencies/"||name)~get
-- Abrufen der Werte von der Website:
-- Abrufen des Preises
preis = .bsf~new("org.jsoup.select.Elements")
preis =doc~select("div.priceValue___11gHJ")~first~text
```

Ausgabe:

Darstellen der Daten:

1) Zugriff zu GUI-Elementen holen

```
::routine buttonClicked1 public    --- Butt

slotDir=arg(arg())  -- note: last argument
  /* @get(label_name)*/
  /* @get(label_wert)*/
  /* @get(label_marktwert)*/
  /* @get(label_marktwert_besch)*/
```

2) Werte auf die Labels schreiben

```
label_wert~text(wert[1])
label_name~text(wert[4])
label_marktwert~text(wert[2])
label_supply~text(wert[3])
label_supply1~text(wert[5])
label_rank~text(wert[6])
```

3) GUI-Elemente sichtbar machen

```
---- Anzeigen sichtbar machen

background~setVisible(.TRUE)
rect1~setVisible(.TRUE)
rect2~setVisible(.TRUE)
rect3~setVisible(.TRUE)
l1~setVisible(.TRUE)
```

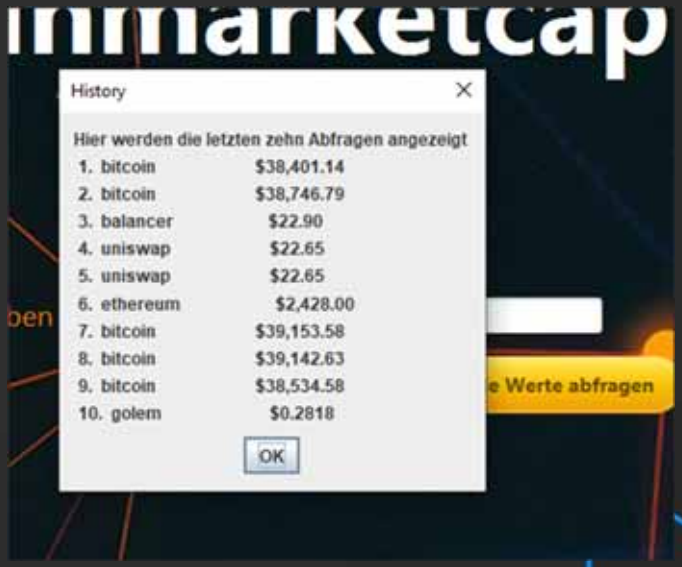
Darstellen der Daten:

Abfrage-History

Darstellung per bsf.dialog Messagebox

Code:

```
.bsf.dialog~messageBox(string,"History")
```



Abfrage-History

Daten in SQLite DB speichern:

1) Alle Abfragen werden in DB gespeichert:

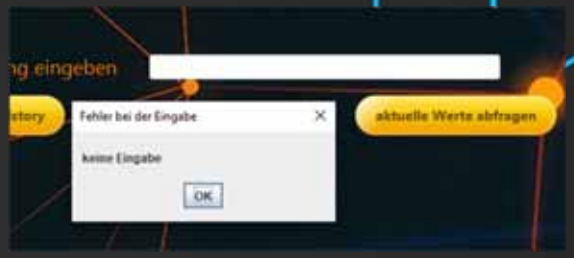
```
query2 = statement~execute("INSERT INTO history (name,price) VALUES(''input~toString'','''input2~toString''');")
```

2) Letzten zehn Datensätze aus DB Laden:

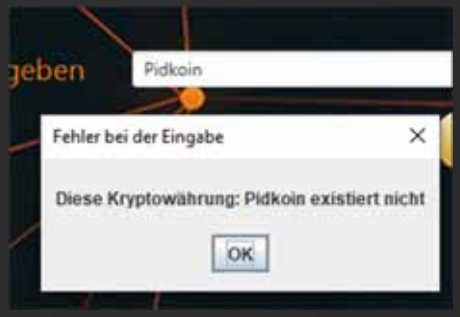
```
query3= statement~executeQuery('SELECT id,name,price FROM history ORDER BY id DESC LIMIT 10')
```

Weitere Funktionen:

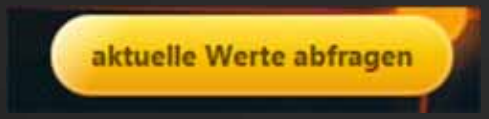
Leerabfragen abfangen:

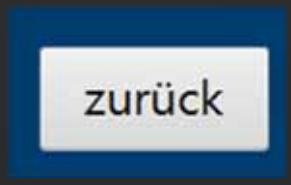


Falsche Eingaben abfangen:



CSS zum Formatieren:






4
Challenges

Challenges

Problem: Werte aus Variablen in DB schreiben

Lösung:

```
input = .bsf~new("java.lang.String",wert[4])
input2 = .bsf~new("java.lang.String",wert[1])

query2 = statement~execute("Insert INTO history (name,price) VALUES('"input~toString"','"input2~toString"');")
```


5
Mögliche
Erweiterungen

Mögliche Erweiterungen

- **Diagramme mit Preisverlauf**
- **Eigenes Portfolio anlegen**
- **Automatisierte Preisabfragen mit Alarmfunktionen**

Thanks


slidesgo