Alternative Programmierkonzepte (T3INF4271)

Logische Programmierung

09
Prolog
Web Programming

DHBW Stuttgart Campus Horb
Fakultät Technik
Studiengang Informatik
Dozent: Antonius van Hoof

AVH 2021

Benutzung dieser Folien ist nur im Rahmen Ihres DHBW-Studiums erlaubt

Prolog und Web

- **Prolog als Web-Client**
  - Aus Prolog hinaus können leicht URLs angesprochen werden
  - Leichte Verarbeitung von HTML in effizienten Prolog-Strukturen
  - Und umgekehrt
- **Prolog als Web-Server (multi-threaded, scalable durch load balancing)**
  - Kann direkt als Webserver gestartet werden
    (Selbständig oder hinter z.B. Apache WS geschalten)
  - Dynamische Seitengenerierung
  - Direkt vorhandenen Bibliotheken für Routing usw.
    damit REST-, AJAX- usw. fähig
  - Funktioniert mit Handlerkonzept (wie in anderen Sprachen auch)
  - Auch Unterstützung von WebSockets
- **Prolog unterstützt auch Web Assembly**
  - WASM: https://webassembly.org/
  - https://www.swi-prolog.org/build/WebAssembly.txt
- **Web-Serving Prolog selbst!**
  - Pengines
  - Prolog in a Sandbox

Web Scraping (Fremde Inhalte holen und parsen)

SWI-Prolog: https://www.youtube.com/watch?v=zKrw-pRI9ac

Gleiche Libraries wie in Scryer. Nur andere Behandlung von Strings

http_open('https://news.ycombinator.com',S,[]), load_html(stream(S),DOM,[]),xpath(DOM,//a(@class='titlelink', text), E).

Prolog als Web-Server

```
:- use_module(library(http/thread_httpd)).
:- use_module(library(http/http_dispatch)).
:- use_module(library(http/html_write)).                % new

:- http_handler(root(hello_world), say_hi, []).         % (1)
:- http_handler('/',say_root,[]).

:- use_module(library(http/http_error)).

server(Port) :-                                         % (2)
        http_server(http_dispatch, [port(Port)]).

say_root(_Request) :-
        format('Content-type: text/plain~n~n'),
        format('You are at the root of this prolog server~n').

say_hi(_Request) :-
        reply_html_page(title('Hello World'),
                        [ h1('Hello World'),
                          p(['This example demonstrates generating HTML ',
                             'messages from Prolog'
                            ])
                        ]).
```

The HTTP server infra structure consists of a number of small modular libraries that are combined into library(http/http_server).

U.a. die hier aufgeführten module

root(): ein Mechanismus um Routes/Directories im Server einfach zu ändern

Man muss Handler melden und definieren. Handler können Argumente haben und bekommen als erstes Argument ein Request-Objekt mit Info

Der Server wird gestartet über http_server/2

Der Handler kann beliebiges zurückgeben (complete control over http response)

reply_html_page/2,3 generiert aus Prolog-DOM eine HTML seite

Prolog DOM und DCG

```
server(Port) :-
        http_server(http_dispatch, [port(Port)]).

list_modules(_Request) :-
        findall(M, current_module(M), List),
        sort(List, Modules),
        reply_html_page(title('Loaded Prolog modules'),
                        [ h1('Loaded Prolog modules'),
                          table([ \header
                                | \modules(Modules)
                                ])
                        ]).

header -->
        html(tr([th('Module'), th('File')])).

modules([]) --> [].
modules([H|T]) --> module(H), modules(T).

module(Module) -->
        { module_property(Module, file(Path)) }, !,
        html(tr([td(Module), td(Path)])).
module(Module) -->
        html(tr([td(Module), td(-)])).
```

- **Komplexe Seiten über Grammatik aufbaubar**
- **Kompletter Durchgriff auf Prolog Environment sowie Betriebssystemumgebung**
  - Natürlich über die entsprechende Libraries dafür (z.B. current_module/1 und module_property/2)
  - Es gibt auch Interfaces zu Datenbanken (ODBC) https://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/odbc.html%27)

Web Sockets

```
:- use_module(library(http/websocket)).
:- use_module(library(http/thread_httpd)).
:- use_module(library(http/http_dispatch)).

:- http_handler(root(ws),
        http_upgrade_to_websocket(echo, []),
        [spawn([])]).

echo(WebSocket) :-
    ws_receive(WebSocket, Message),
    (   Message.opcode == close
    ->  true
    ;   string_concat('Hey, you said: ', Message.data, MessageRes),
        writeln(WebSocket),writeln(MessageRes), %avh: my line
        ws_send(WebSocket, text(MessageRes)),
        echo(WebSocket)
    ).

:- initialization
        http_server(http_dispatch, [port(8083)]).
```

Aus Browser aufrufbar unter

wsUri = 'ws://localhost:8083/ws'

Javascript:

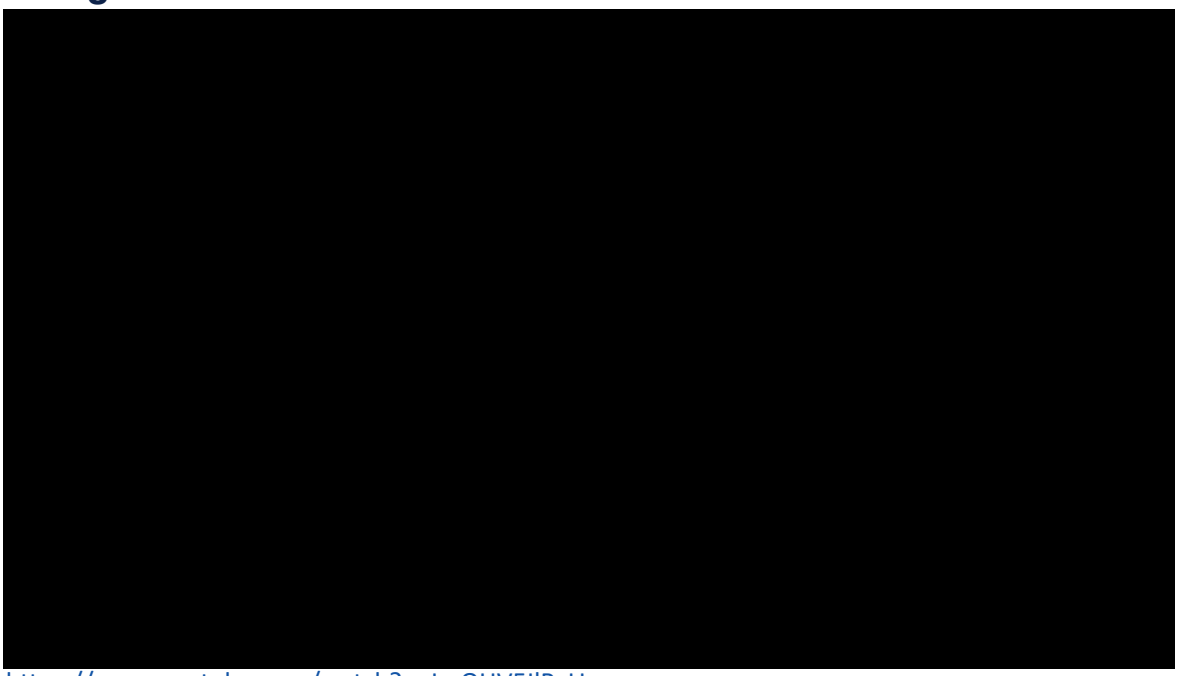
let ws = this.ws = new WebSocket(wsUri);

... usw.

Weiteres Demo:

Web socket chat

Pengines



https://www.youtube.com/watch?v=JmOHV5IlPyU

Weitere Infos

- **SWI Prolog Dokuseiten zu Web Programming**
  - https://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/http.html%27)
  - Aber auch:
    SSL support: https://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/ssl.html%27)
    SGML/HTML/XML support:
    https://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/sgml.html%27)
  - Zu Pengines:
    https://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/pengines.html%27)
    https://github.com/SWI-Prolog/pengines
- **YouTube**
  - https://www.youtube.com/playlist?list=PLo2Yjnxu38Q8VI6WVFLITQOZ9LH9M4wyk
  - Begleitmaterial:
    https://github.com/Anniepoo/strangeloop
  - Noch ausführlicher Material:
    https://github.com/Anniepoo/swiplwebtut/blob/master/web.adoc