Making Web Sites Dynamic

Part of Communication and Visualisation

Th. Leize

Summer 2025

Table of Contents – what comes next?

Motivation / Outline

What is the current state of our knowledge in this semester?
What do we already know?

- HTTP Protocol to get web page contents.
- Static web pages, including
  - HTML
  - CSS
  - Forms

Motivation / Outline 2

What will come next?

- Upload of form data and processing this data in the backend.
- Dynamic web pages / On the fly generated web pages. Programs run on the server and create web pages for delivery to the client.
- First programming language: Perl, easy and fast
- Storing data on the server
  - in text files
  - in XML files
  - in Databases (Introduction to databases necessary)
- reading the data and displaying on a dynamic web page

Motivation / Outline 3

Later:

- we will do a bigger project with a backend of your choice.
- data privacy in general and on web pages
- (Maybe also some front-end programming (Javascript)

Prerequisites:
Please download and copy and paste the "tiny web server" on Ilias to your local drive. This contains everything we need and also contains these examples.

Table of Contents – what comes next?

Perl 1

Perl is just one of the possibilities for backend programming.
Perl is:

- It is pre-installed with the tiny web browser we use
- It is easy to use, not much theory needed
- It's an interpreted language

That's why we want to start with that.

Perl 2

Special rules in the programming language Perl:

- Function call parenthesis may be there or be omitted

```
print("Hello, world");
# same:
print "Hello, world";
```

- # comment
- String literals are embedded in single or double quotes.

Perl - Names and Variables

No declaration needed, just name the variables:

$	variable or array element ("scalar")
@	array, ordered list of scalars, numbered starting with 0
%	"hash", unordered list of key/value pairs. Keys as subscripts.

Perl - Arrays

Just an example:

```
@array = (1,2,3,4,5);
@array2 = qw/1 2 3 4 5/;
$size = array;
print $array[0];
```

Arrays may be dynamically extended by just writing to them.
Negative indexes count from the end.

Perl - More see docs

For more details see the online docs.
But how to produce HTML files?
See:
`https://www.irt.org/articles/js171/index.html#cgi_object`

Perl - Example / Try yourself!

Our tiny web browser comes with a perl example.
Please have a look and try and try to understand.
Modify and extend!

Perl - Generating HTML 1: Plain

You simply can produce the html output "by hand":

```
#!/usr/bin/perl
print "Content-type: text/html\r\n\r\n";
print "<HTML>\n";
print "<HEAD><TITLE>Hello World!</TITLE></HEAD>\n";
print "<BODY>\n";
print "<H2>Hello World!</H2>\n";
print "<a href='../index.html'>Home</a>";
print "</BODY>\n";
print "</HTML>\n";
exit (0);
```

Example of our tiny web server. Have a look and try!

Perl - Plain output / Explanation

On the previous slide please note that there is a complete http response.

- Please remember the structure of the response.
- Identify a parameter that is transferred
- There is the empty line at the end of the parameters
- everything after the empty line is the html file

Perl - Plain output / Easy File I/O

Please note that the hello world example also contains an example of writing to a local file on the server:

```
#!/usr/bin/perl
open (MYFILE, '>>data.txt');
print MYFILE "test\n";
close (MYFILE);
```

We will need it later to store some data on the server.

Perl - HTML Output 2: Module CGI

There is a ready done module in perl for generation of html files.
You can extend the functionality of perl with modules. Just load them to the correct directory (we might later need for e.g. XML and/or graphics.
The module CGI already is included in the tiny web server.

Perl - HTML Output 2: Module CGI: Example

```
#!/usr/bin/perl
# load modules:
    use strict;
    use warnings;
    use CGI;
 # create CGI object
    my $q = new CGI;
 # and use it. It has methods for
 # many element types:
    print $q->header;
    print $q->start_html;
    print $q->h1("Hello␣World!");
    print $q->end_html;
```

Perl Form Data Processing

In Perl it also is easy to access and process the data the user gave to the form. See form on next page and perl script on next but one page.

Perl Form Data Processing – HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Your Comment</title>
</head>
<body bgcolor="#E0E0E0">
<h1>Your Comment</h1>
<form action="/cgi-bin/comments.pl" method="post">
<p>Name:<br>
<input type="text" name="UserName" size="40" maxlength="40">
</p><p>Text:<br>
<textarea rows="5" cols="50" name="CommentText">
</textarea></p>
<p><input type="submit" value="Submit"></p>
</form>
</body>
</html>
```

Perl Form Data Processing – Perl

```
#!/usr/bin/perl -w
use CGI;
my $q = new CGI;
print $q->header;
print $q->start_html(-title => "An␣example␣web␣page
my $name = $q->param('UserName');
my $text = $q->param('CommentText');
print $q->h1("CGI-Feedback␣from␣<i>comments.pl</i>"
print $q->ul(
$q->li(["<b>Name:</b>␣$name</p>",
            "<b>Comment:</b>␣$text"])
);
print $q->end_html;
```

Perl Exercises

Please test forms and reading from forms on the server. Sending the form data back in a dynamic web page.
Content is up to you.

Perl Exercises

Please see Ilias for our next assignment.

Table of Contents – what comes next?

GNUPlot

GNUPlot is open source (but not GPL!), see `www.gnuplot.info`
GNUPlot is a quite old command line tool for generating high quality graphics.
You can download it from that site or install inside cygwin (recommended).
Available for nearly all platforms.

GNUPlot – how to use

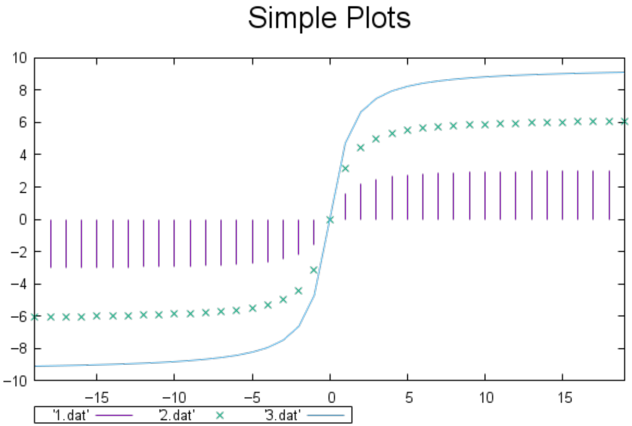
GNUPlot can be used to periodically generate a graphics file and show this graphically with your web site.
Exercise:

- First install GNUPlot
- Run GNUPlot on the command line by hand
- Include in your program

Note: You have to store the values in a suitable format.

GNUPlot – example

This is an example of the GNUPlot examples given on the web site:



Using Perl Modules

GNUPlot also can be used from within a Perl program.
From inside Perl you can create plots using the module GD.
Exercise:

- Search for this module and add it to your web server, if necessary
- Search for some documentation of that module. (e.g. see here: `http://www.mathematik.uni-ulm.de/help/perl5/doc/GD/Graph.html`
- Extend your server perl program with such a module

Table of Contents – what comes next?

File Formats

Next step: Let's think about alternative, maybe more convenient file formats.

Possibilities

- XML
- JSON
- Databases (extra, longer chapter)

XML – Extendable Markup Language

Same idea as HTML, list of tags.
Frequently used standard for storing data. Gives easy definition of structures of the data.
Good to store objects.
Even simply and tiny databases sometime just are based on XML.

XML – Start

Recommended: Give information about XML.

```
<?xml version="1.0" encoding="utf-8"
      standalone="yes" ?>
<!-- This is a comment, same as in HTML -->
```

XML – Tags

Tags are same as we already used with HTML.
But:
tags always have to have values or be empty. So: End tag necessary.
tag names must not start with numbers or xml and may not contain spaces or =

```
<dataset>
<!-- dataset may contain several items-->
<dataitem>
<value>23</value>
<unit>m</unit>
<comment />    <!-- empty -->
</dataitem>
</dataset>
```

XML – Example from real life

For an example, too big for these slides,
see Ilias, inside "Exercises etc."

It's an example data file of our buoy that swam inside the Rhine river, full of sensors to measure different properties of the water in the river. It very nicely shows the usage of XML.

XML – Exercise

Please add an output in XML to your program

- Do it by hand, by direct output
- **or** use an additional Perl module

If you add schemes and css to the xml, it can automatically be displayed in a nice way.

JSON

An even younger alternative is JSON.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

(from `https://www.json.org/`)

JSON – Example

```
{
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
}
```

JSON example

JSON – Exercise

Please add a JSON output file.
Please add an output in XML to your program

- Do it by hand, by direct output
- **or** use an additional Perl module

Table of Contents – what comes next?

To be continued

More to follow. . .