mod_rexx

A rexx module for the apache webserver

Carsten Mjartan

Ski seminary WS 2000/2001

Wirtschaftsinformatik und Softwaretechnik

Uni/GH Essen

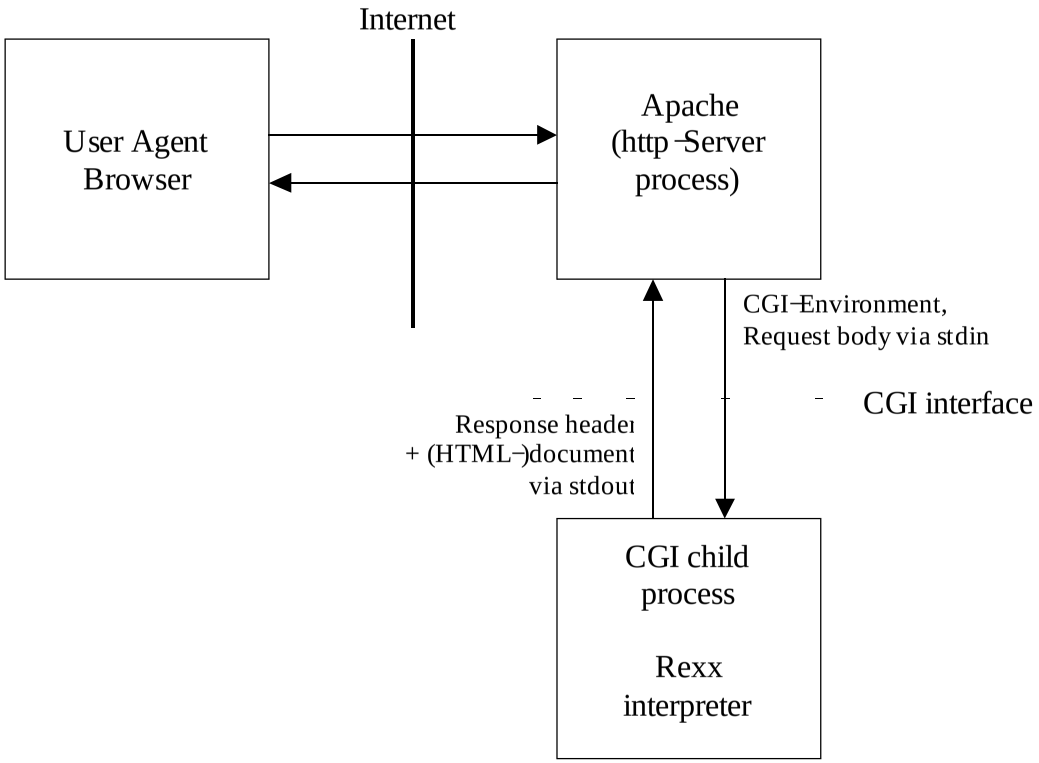
Overview

- CGI versus mod_rexx
- The Apache API
- The Rexx SAA API
- mod_rexx
- Rexx script examples

Tasks

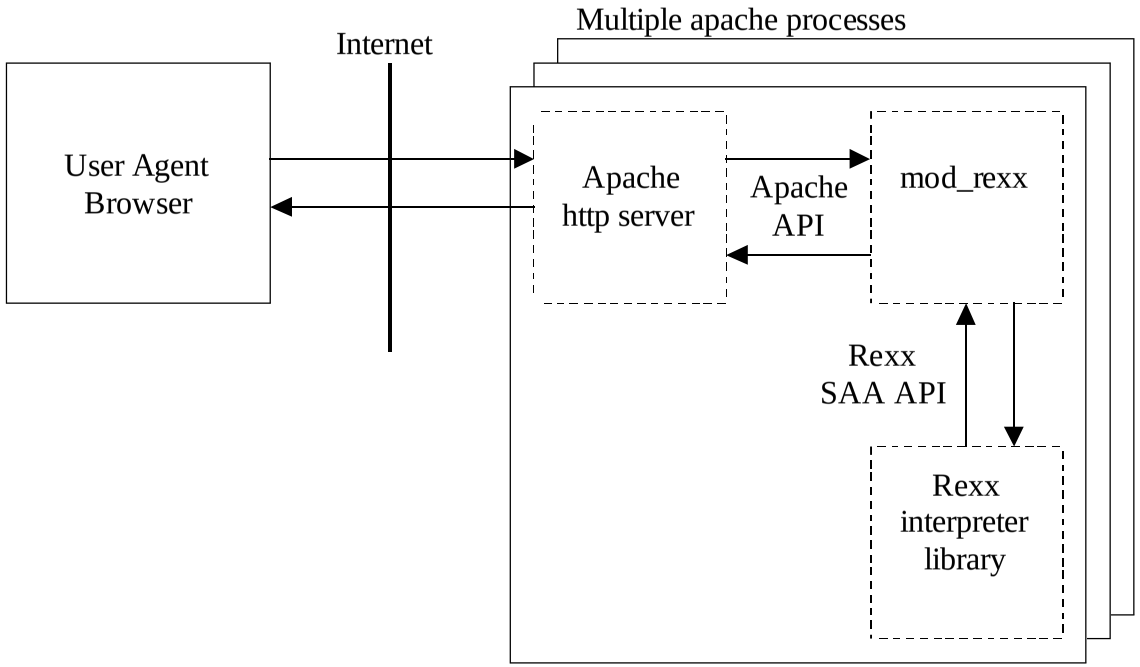
- Introduction to the mod−architecture of the open source server apache
- Study of existing implementations for different scripting languages
- Development and implementation of a platform independent mod−rexx

The CGI interface



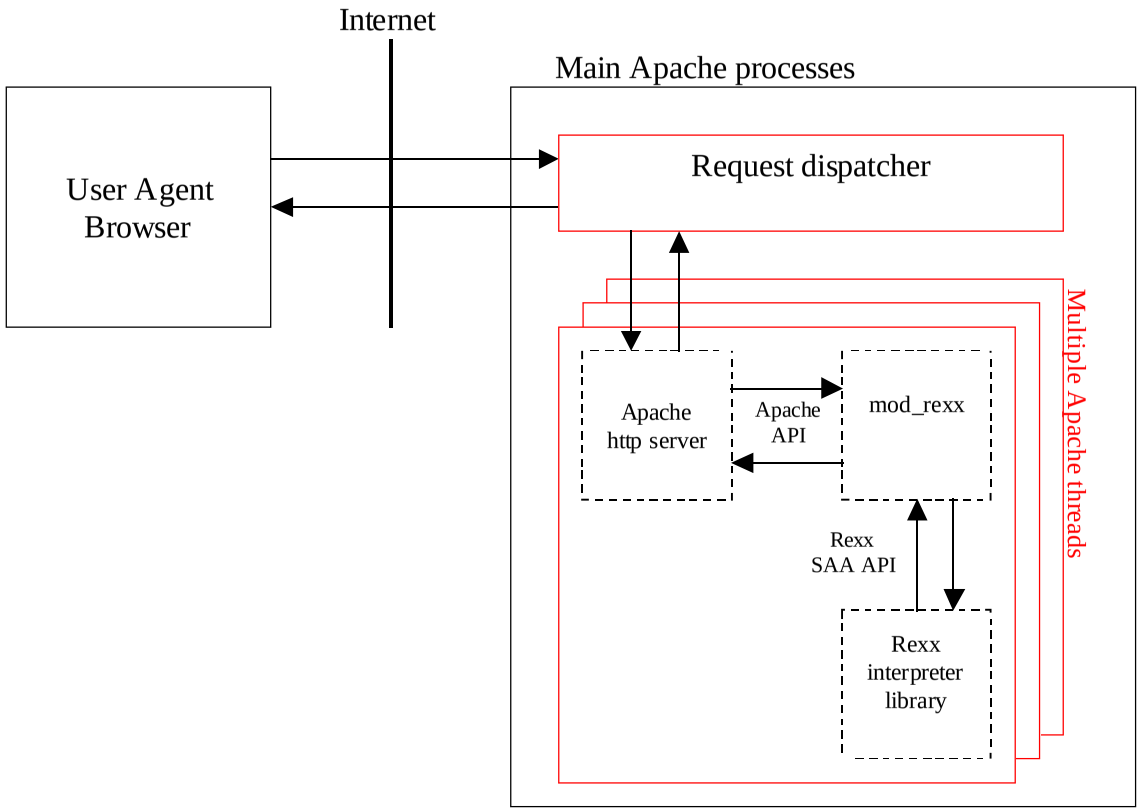
The Apache API

(without multithreading)



The Apache API

(with multithreading)



Apache API basic concepts

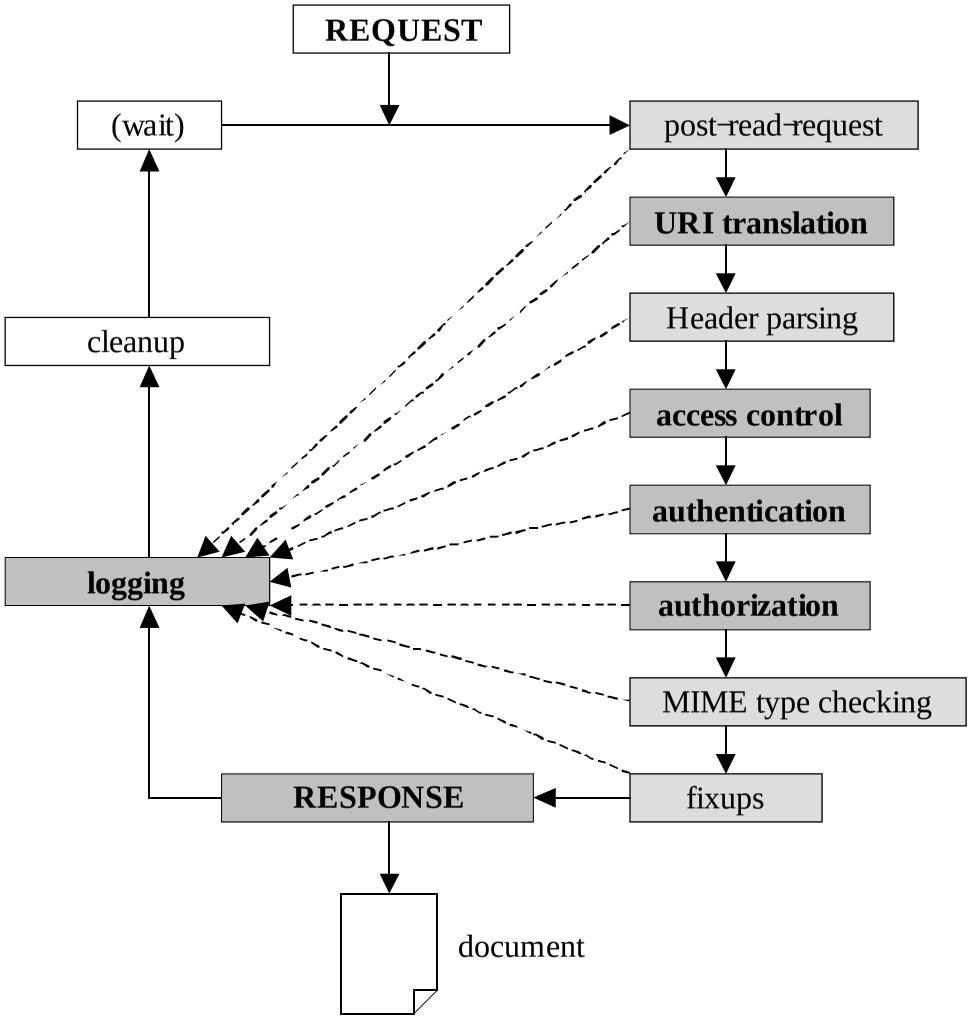
- Requests, handlers, ’module’−structure
- Resource management

  request pool, server pool, ...
- Apache configuration

  per server / per directory
- API functions

  ap_*

Apache request handling cycle



mod_rexx.c structure

```
/* Apache specific includes */
#include "httpd.h"
#include "http_config.h"
#include "http_request.h"
#include "http_core.h"
#include "http_protocol.h"
#include "http_main.h"
#include "http_log.h"
#include "util_script.h"
#include "http_conf_globals.h"
#include "multithread.h"
#include "util_script.h"

/* Declaration of Apache Handlers */
static void rexx_initialize(server_rec *s, pool *p);
static char *rexx_create_dir_config(pool *p, char *path);
static const char *rexx_cmd_createEnvironment(cmd_parms
        *parms,
void *mconfig, char *yesno);
static int rexx_handler(request_rec *r);

/* Rexx-specific configuration commands */
static const command_rec rexx_commands[] = { ... };

/* Content Handlers */
static const handler_rec rexx_handlers[] = {
    { "application/x-httpd-rexx",      rexx_handler },
    { "rexx-handler",                  rexx_handler },
    { NULL }
};
```

```
/* module-structure */

module MODULE_VAR_EXPORT rexx_module = {
                STANDARD_MODULE_STUFF,
                rexx_initialize,          /* module initializer */
                rexx_create_dir_config, /* per-directory config
    creator */
                NULL,                     /* dir config merger */
                NULL,                     /* server config creator */
                NULL,                     /* server config merger */
                rexx_commands,            /* command table */
                rexx_handlers,            /* [9] list of handlers */
                NULL,                     /* [2] filename-to-URI
    translation */
                NULL,                     /* [5] check/validate user_id
    */
                NULL,                     /* [6] check user_id is valid
    *here* */
                NULL,                     /* [4] check access by host
    address */
                NULL,                     /* [7] MIME type
    checker/setter */
                NULL,                     /* [8] fixups */
                NULL,                     /* [10] logger */
NULL,                   /* [3] header parser */
NULL,                   /* process initializer */
NULL,                   /* process exit/cleanup */
NULL                    /* [1] post read_request handling */
};
```

The Rexx SAA API

- was standardized to ensure platform and interpreter independence
- enables applications to embed Rexx for macro/script processing

API functions

- Subcommand handler interface
- External function handler interface
- Execution of Rexx macros/scripts with RexxStart()
- Variable Pool interface
- System exit handler interface

Workflow

- Online search
  - for Apache/Rexx documentation, books
  - Apache, Regina, PHP, mod_perl source code
- Selection of development tools
  - Apache 1.3.14, Regina Rexx interpreter
- Prototyping
- mod_rexx development

mod_rexx characteristics

- Input/Output redirection
- GET/POST data conversion/transfer
- Multithreading
- Module configuration

Environment variables

Problem: Multithreading and

global variables / process environment

Solution: Storage of the process environment variables in Rexx variables for threadsave access

Program flow

- Initialization
- Configuration
- Request handling
  - Environment variable generation
  - Creation of the APACHE!ENVIRONMENT and APACHE!QUERY Rexx variable structures
  - POST−Request?: Upload of the request body and decoding of POSTed form variables
  - Storage of all request specific data (request_rec, variables) in a request_data structure
  - Registration of Rexx handlers (Initialization and I/O)
  - Configuration dependent: environment update
  - RexxStart() script execution
  - Deregistration of Rexx handlers

Generated Rexx Variables

- Environment:

Names:	APACHE!ENVIRONMENT.i.NAME
Values:	APACHE!ENVIRONMENT.i.VALUE
with 1 <= i <=	APACHE!ENVIRONMENT.0

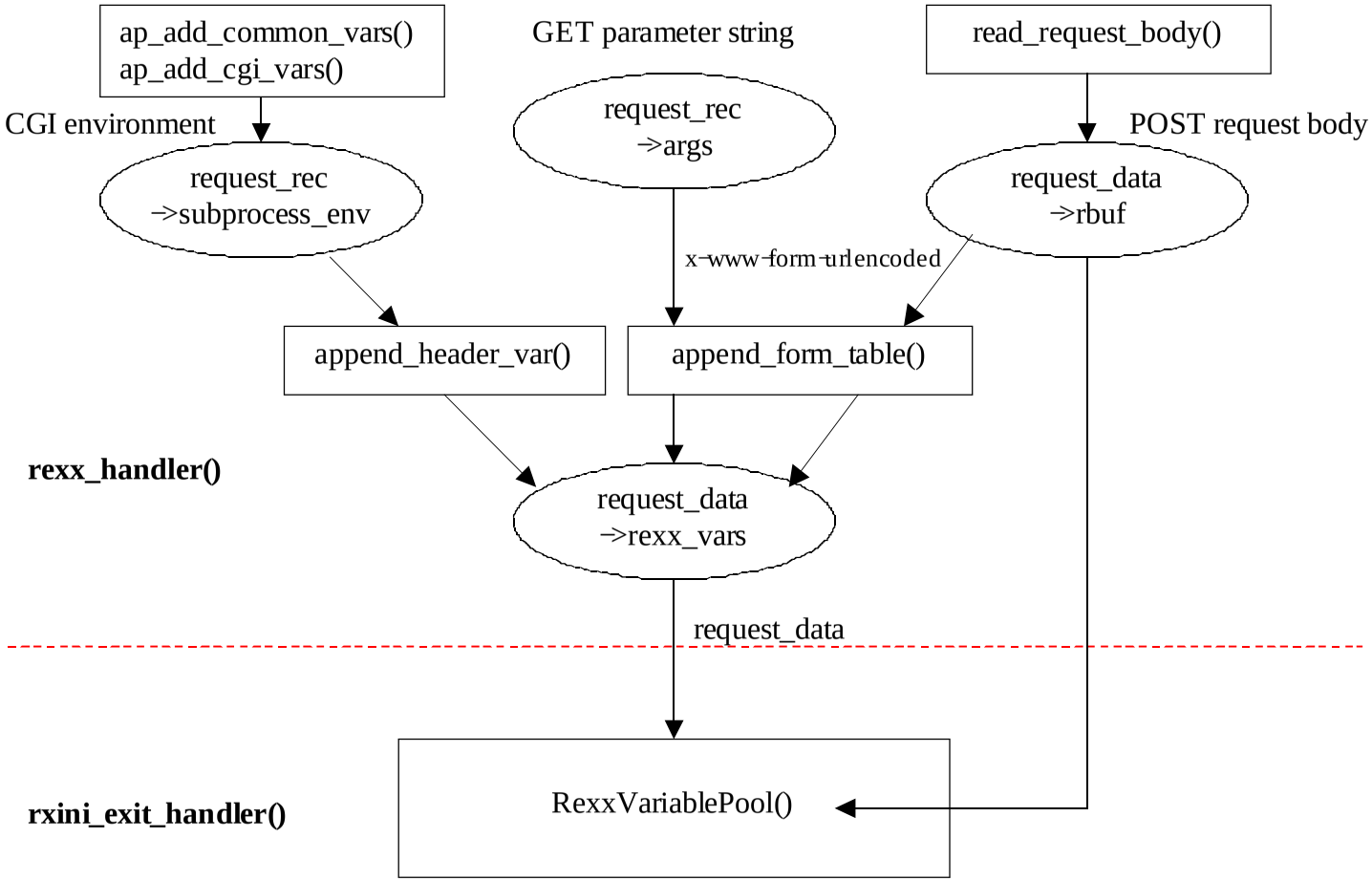
- Querystring: APACHE!QUERY_STRING

urldecoded variables:	APACHE!QUERY.varname
multivalued variables:	APACHE!QUERY.multivar.1
to	APACHE!QUERY.multivar.n
element count n:	APACHE!QUERY.multivar.0

- POST request body: APACHE!POST_DATA

urldecoded variables:	APACHE!FORM.varname
multivalued variables:	APACHE!FORM.multivar.1
to	APACHE!FORM.multivar.n
element count n:	APACHE!FORM.multivar.0

Request data flow



Module configuration example

apache−dir/conf/httpd.conf:

```
# Loading the Rexx dynamic module on Windows
LoadFile c:/regina/regina.dll
LoadModule rexx_module modules/ApacheModuleRexx.dll

# Loading the Rexx dynamic module on Unix systems
LoadModule rexx_module modules/mod_rexx.so
AddModule mod_rexx.c
...
<IfModule mod_mime.c>
AddType application/x−httpd−rexx .rexx
</IfModule>

<Location /rexx−scripts>
SetHandler rexx−handler
</Location>
```

Module configuration

```
/* Data structure to the storage of the module configuration */
typedef struct {
int createEnvironment;
} rexx_dir_config;

/* Reservation of storage for the module configuration */
/* and setting of defaults */
static char *rexx_create_dir_config(pool *p, char *path) {
rexx_dir_config *cfg = (rexx_dir_config *)
ap_palloc(p, sizeof(rexx_dir_config));
cfg->createEnvironment = 1;
return (void *) cfg;
}
/* Command function for RexxCreateEnvironment */
static const char *rexx_cmd_createEnvironment(cmd_parms
        *parms,
void *mconfig, char *yesno) {
rexx_dir_config *cfg = (rexx_dir_config *) mconfig;
/* check parameter validity and update configuration */
if ( (!strcasecmp(yesno, "yes")) || (!strcasecmp(yesno, "on")) ) {
cfg->createEnvironment = 1;
} else if ( (!strcasecmp(yesno, "no"))
|| (!strcasecmp(yesno, "off")) ) {
cfg->createEnvironment = 0;
} else {
return "parameter yes,no,on or off allowed";
}
return NULL;
}
```

```
static const command_rec rexx_commands[] = {
{
"RexxCreateEnvironment",        /* name of the command */
rexx_cmd_createEnvironment,  /* accompanying function */
NULL,                           /* pointer to a data field to */
 /* be delivered to the function*/
ACCESS_CONF,                    /* valid areas within the */
/* configuration */
TAKE1,                          /* type <=> numb. of arguments
        */
"yes/no as parameter, default is yes" /* command description */
},
{ NULL }
};

static int rexx_handler(request_rec *r) {
...
rexx_dir_config *config;        /* per dir configuration */
...
config = (rexx_dir_config *) ap_get_module_config(
r->per_dir_config, &rexx_module);
if (config->createEnvironment) {
...
}
...
}
```

Output redirection

- Registration of the Rexx I/O Exit Handler
- Redirection of standard/error output
- Handling of response headers

```
LONG APIENTRY rxsio_exit_handler(LONG ExitNumber,
LONG Subfunction, PEXIT ParmBlock)
{
   USHORT        query_flag;
   request_data  *handler_info[2];
   request_data  *rdata;
   request_rec   *request;

   RXSTRING      str;
   char                              *hkey, *hval;

   if (ExitNumber != RXSIO) return RXEXIT_NOT_HANDLED;

   /* get request information */
   RexxQueryExit("rxsio_handler", NULL, &query_flag,
(PUCHAR) handler_info);
   rdata = handler_info[0];
   request = rdata->request;

   if (!request) return RXEXIT_RAISE_ERROR;

   switch (Subfunction) {
      case RXSIOTRC:
         /* trace- or error messages are redirected */

         if (!rdata->headers_sent) {
            ap_send_http_header (request);
            rdata->headers_sent = 1;
         }

         str = ((RXSIOTRC_PARM *)ParmBlock)->rxsio_string;

         ap_rputs("</TD></TR></TABLE><B>", request);
         ap_rwrite(str.strptr, str.strlength, request);
         ap_rputs("</B>\n", request);

case RXSIOSAY:
         /* text output by SAY is redirected */

         str = ((RXSIOSAY_PARM *)ParmBlock)->rxsio_string;

         if (!rdata->headers_sent) {
            if (str.strlength) {
               /* Zur Headerliste hinzufgen: */
               hkey = ap_pstrndup(request->pool, str.strptr,
str.strlength);
               if (hval = strchr(hkey, ':')) {
                  *hval = '\0';
                  while (*(++hval) == ' ');
                  if (! strcmp(hkey, "Content-type")) {
                     request->content_type = hval;
                  } else {
                     ap_table_set (request->headers_out, hkey, hval);
                  }
               }
            } else {
               /* empty line ==> send headers */
               ap_send_http_header (request);
               rdata->headers_sent = 1;
            }
         } else {
            ap_rwrite(str.strptr, str.strlength, request);
            ap_rputc('\n', request);
         }
         break;

      case RXSIOTRD:
        /* ... */
      case RXSIODTR:
      default:
         return RXEXIT_NOT_HANDLED;
   }
```

Problems

- CGI compliance
  - Environment <=> Multithreading
  - Rexx Standard I/O via LINEIN/LINEOUT
  - Output redirection for shell commands (SYSTEM subcommands)

Future enhancements

- Adding more features
  - script caching, session support ...
- Developing or porting a Rexx toolkit to mod_rexx for interface independent script development

hello_world.rexx

```
SAY 'Content-type: text/html'
SAY ''
SAY '<html>'
SAY '<head>'
SAY '    <title>Easy example for text output</title>'
SAY '</head>'
SAY '<body>'
SAY '    <div align=center>'
SAY '        <h1>Hello World</h1>'
SAY '        powered by mod_rexx'
SAY '    </div>'
SAY '</body> '
SAY '</html>'
```

env.rexx

```
SAY 'Content-type: text/html'
SAY ''
SAY '<html>'
SAY '<head>'
SAY '    <title>Output of the CGI environment variables</title>'
SAY '</head>'
SAY '<body>'
SAY '    <div align=center>'
SAY '        <h1>Current Environment:</h1>'
SAY '        <table border=1 cellspacing=0 cellpadding=2>'

DO i=1 to APACHE!ENVIRONMENT.0
SAY '            <tr><td>'APACHE!ENVIRONMENT.i.NAME'</td>'
SAY '                <td>'APACHE!ENVIRONMENT.i.VALUE'</td></tr>'
END

SAY '        </table>'
```

getpost.html

```
<html>
<head>
    <title>HTML-Form for getpost.rexx</title>
</head>
<body>
    <div align=center>
        <h1>POST Form:</h1>
        <form action="getpost.rexx?id=1" method=post>
            Please type in your first name: <input type=text name="Name">
            <input type=submit name="Send" value="Send">
        </form>
    </div>
</body>
</html>
```

getpost.rexx

```
SAY 'Content-type: text/html'
SAY ''
SAY '<html>'
SAY '<head>'
SAY '    <title>Access to GET- und POST-variables</title>'
SAY '</head>'
SAY '<body>'
SAY '    <div align=center>'
SAY '        <h1>Hello 'APACHE!FORM.NAME',</h1>'
SAY '        Your ID is 'APACHE!QUERY.ID
SAY '    </div>'
SAY '</body>'
SAY '</html>'
```

The End