```
/*-----------------------------------------------------
|   The "cgiParse"-class contains                     |
|   REXX Methods to handle CGI-Environment variables  |
|   e.g. parse the query-string, ...                  |
|   emasovic@wi-inf.uni-essen.de                      |
|   http://swt.wi-inf.uni-essen.de/~emasovic          |
|   based on CGIParse by Sacha Prins                  |
|   (slightly modified version) 24.11.2000            |
-----------------------------------------------------*/

::CLASS cgiParse PUBLIC                                 -- works only with CGI-GET-Method

   ::METHOD parseQueryString
      queryString = self~getEnv('QUERY_STRING')         --retrieve query string from the environment
      queryString= TRANSLATE(queryString, ' ', '+')     --retranslate '+'-Symbols to Spaces
      DO WHILE LENGTH(queryString) > 0
         varCouple= ''
         PARSE VAR queryString varCouple'&'queryString  --find parts between '&'-Symbol
         PARSE VAR varCouple varName'='varVal           --and split them into varName and varValue
         IF varName = '' | varVal= '' THEN ITERATE      --anything empty? => then find next part
         varName= 'cgi.' || self~urlDecode(varName)     --prepare stem-variable name, decode '%'-Expressions
         varVal=  self~urlDecode(varVal)                --prepare value and decode '%'-Expressions
         IF SYMBOL(varName) = 'BAD' THEN ITERATE        --varName cannot be used as a Rexx-Variable, find next part

                                                        --multiple assignments to the same variable?
                                                        --retain old value and append new, separated by 'Linefeed'
         IF VALUE(varName) \= TRANSLATE(varName) THEN call VALUE varName, VALUE(varName) || '0a'x || varVal
         ELSE call VALUE varName, varVal                --assign the variable to the Rexx-Stem-Variable
      END
      RETURN cgi.                                       --return Stem Variable

   /*****************************************************************/
      :: METHOD URLDecode PRIVATE
         IF ARG()\=1 THEN RETURN ''                     --only one argument permitted
         line= ARG(1)
         lineLen= LENGTH(line)
         newLine= ''
         i=1
```

```
39              DO WHILE i <= lineLen
40                  c= SUBSTR(line, i, 1)
41                  IF c \= '%' THEN newLine = newLine || c      --normal character? => then append
42                  ELSE IF i+2 <= lineLen THEN                  --is tested char = '%' (special char in query strings)?
43                      DO
44                          newLine= newLine || x2c(SUBSTR(line, i+1, 2)) --append decoded character
45                          i=i+2
46                      END
47                  i= i+1
48              END
49              RETURN newLine                                   --return completely decoded string
50
51      /*********************************************************************/
52          ::METHOD getEnv PRIVATE
53              RETURN VALUE(ARG(1),, 'ENVIRONMENT')             --retrieves variable from environment
54
55      /*********************************************************************/
56          ::METHOD putEnv PRIVATE                              --sets an environment variable
57              RETURN VALUE(ARG(1), ARG(2), 'ENVIRONMENT')      --(not used here)
58  /*********************************************************************/
```