**// Warnung das bashttpd.conf erstellt wurde. Diese Nachricht wird im Terminal angezeigt nach der Ausführung der Datei**

|  |
| --- |
| EOF |
|  | warn "Created bashttpd.conf using defaults. Please review it/configure before running bashttpd again." |
|  | exit 1 |
|  | } |
|  |  |
|  | recv() { echo "< $@" >&2; } |
|  | send() { echo "> $@" >&2; |
|  | printf '%s\r\n' "$\*"; } |
|  |  |
|  | [[ $UID = 0 ]] && warn "It is not recommended to run bashttpd as root." |
|  | **//declaration von den header** |
|  | DATE=$(date +"%a, %d %b %Y %H:%M:%S %Z") |
|  | declare -a RESPONSE\_HEADERS=( |
|  | "Date: $DATE" |
|  | "Expires: $DATE" |
|  | "Server: Slash Bin Slash Bash" |
|  | ) |
|  |  |
|  | add\_response\_header() { |
|  | RESPONSE\_HEADERS+=("$1: $2") |
|  | } |
|  | **// Deklaration der Errors die dan Funktionen zugeordnet werden** |
|  | declare -a HTTP\_RESPONSE=( |
|  | [200]="OK" |
|  | [400]="Bad Request" |
|  | [403]="Forbidden" |
|  | [404]="Not Found" |
|  | [405]="Method Not Allowed" |
|  | [500]="Internal Server Error" |
|  | ) |
|  | **//Funktion Zur Antwort der Anfrage** |
|  | send\_response() { |
|  | local code=$1 |
|  | send "HTTP/1.0 $1 ${HTTP\_RESPONSE[$1]}" |
|  | for i in "${RESPONSE\_HEADERS[@]}"; do |
|  | send "$i" |
|  | done |
|  | send |
|  | while read -r line; do |
|  | send "$line" |
|  | done |
|  | } |
|  |  |
|  | send\_response\_ok\_exit() { send\_response 200; exit 0; } |
|  |  |
|  | fail\_with() { |
|  | send\_response "$1" <<< "$1 ${HTTP\_RESPONSE[$1]}" |
|  | exit 1 |
|  | } |
|  | **//Eine Datei zurückgeben die Typen css js werden unterstuzt** |
|  | serve\_file() { |
|  | local file=$1 |
|  |  |
|  | CONTENT\_TYPE= |
|  | case "$file" in |
|  | \*\.css) |
|  | CONTENT\_TYPE="text/css" |
|  | ;; |
|  | \*\.js) |
|  | CONTENT\_TYPE="text/javascript" |
|  | ;; |
|  | \*) |
|  | read -r CONTENT\_TYPE < <(file -b --mime-type "$file") |
|  | ;; |
|  | esac |
|  |  |
|  | add\_response\_header "Content-Type" "$CONTENT\_TYPE"; |
|  |  |
|  | read -r CONTENT\_LENGTH < <(stat -c'%s' "$file") && \ |
|  | add\_response\_header "Content-Length" "$CONTENT\_LENGTH" |
|  |  |
|  | send\_response\_ok\_exit < "$file" |
|  | } |
|  | **//Rückgabe von Directory plus tree (muss aber installiert sein)** |
|  | serve\_dir\_with\_tree() |
|  | { |
|  | local dir="$1" tree\_vers tree\_opts basehref x |
|  |  |
|  | add\_response\_header "Content-Type" "text/html" |
|  |  |
|  | # The --du option was added in 1.6.0. |
|  | read x tree\_vers x < <(tree --version) |
|  | [[ $tree\_vers == v1.6\* ]] && tree\_opts="--du" |
|  |  |
|  | send\_response\_ok\_exit < \ |
|  | <(tree -H "$2" -L 1 "$tree\_opts" -D "$dir") |
|  | }  **//Rückgabe von Directory mittels ls - a** |
|  |  |
|  | serve\_dir\_with\_ls() |
|  | { |
|  | local dir=$1 |
|  |  |
|  | add\_response\_header "Content-Type" "text/plain" |
|  |  |
|  | send\_response\_ok\_exit < \ |
|  | <(ls -la "$dir") |
|  | }  **// Rückgabe von Directory** |
|  |  |
|  | serve\_dir() { |
|  | local dir=$1 |
|  |  |
|  | # If `tree` is installed, use that for pretty output. |
|  | which tree &>/dev/null && \ |
|  | serve\_dir\_with\_tree "$@" |
|  |  |
|  | serve\_dir\_with\_ls "$@" |
|  |  |
|  | fail\_with 500 |
|  | } |
|  | **//Gibt eine File oder eine Directory zurruck** |
|  | serve\_dir\_or\_file\_from() { |
|  | local URL\_PATH=$1/$3 |
|  | shift |
|  |  |
|  | # sanitize URL\_PATH |
|  | URL\_PATH=${URL\_PATH//[^a-zA-Z0-9\_~\-\.\/]/} |
|  | [[ $URL\_PATH == \*..\* ]] && fail\_with 400 |
|  |  |
|  | # Serve index file if exists in requested directory |
|  | [[ -d $URL\_PATH && -f $URL\_PATH/index.html && -r $URL\_PATH/index.html ]] && \ |
|  | URL\_PATH="$URL\_PATH/index.html" |
|  |  |
|  | if [[ -f $URL\_PATH ]]; then |
|  | [[ -r $URL\_PATH ]] && \ |
|  | serve\_file "$URL\_PATH" "$@" || fail\_with 403 |
|  | elif [[ -d $URL\_PATH ]]; then |
|  | [[ -x $URL\_PATH ]] && \ |
|  | serve\_dir "$URL\_PATH" "$@" || fail\_with 403 |
|  | fi |
|  |  |
|  | fail\_with 404 |
|  | }  **//Gibt string zurruck** |
|  |  |
|  | serve\_static\_string() { |
|  | add\_response\_header "Content-Type" "text/plain" |
|  | send\_response\_ok\_exit <<< "$1" |
|  | }  **//Uberpruft den link** |
|  |  |
|  | on\_uri\_match() { |
|  | local regex=$1 |
|  | shift |
|  |  |
|  | [[ $REQUEST\_URI =~ $regex ]] && \ |
|  | "$@" "${BASH\_REMATCH[@]}" |
|  | } |
|  |  |
|  | unconditionally() { |
|  | "$@" "$REQUEST\_URI" |
|  | } |
|  |  |
|  | # Request-Line HTTP RFC 2616 $5.1 |
|  | read -r line || fail\_with 400 |
|  |  |
|  | # strip trailing CR if it exists |
|  | line=${line%%$'\r'} |
|  | recv "$line" |
|  |  |
|  | read -r REQUEST\_METHOD REQUEST\_URI REQUEST\_HTTP\_VERSION <<<"$line" |
|  |  |
|  | [ -n "$REQUEST\_METHOD" ] && \ |
|  | [ -n "$REQUEST\_URI" ] && \ |
|  | [ -n "$REQUEST\_HTTP\_VERSION" ] \ |
|  | || fail\_with 400 |
|  |  |
|  | # Only GET is supported at this time |
|  | [ "$REQUEST\_METHOD" = "GET" ] || fail\_with 405 |
|  |  |
|  | declare -a REQUEST\_HEADERS |
|  |  |
|  | while read -r line; do |
|  | line=${line%%$'\r'} |
|  | recv "$line" |
|  |  |
|  | # If we've reached the end of the headers, break. |
|  | [ -z "$line" ] && break |
|  |  |
|  | REQUEST\_HEADERS+=("$line") |
|  | done |
|  |  |
|  | source "${BASH\_SOURCE[0]%/\*}"/bashttpd.conf |
|  | fail\_with 500 |