Módulo HTTP

Este módulo fornece funções para realizar requisições HTTP a partir de scripts Lua. As funções retornam uma tabela com os resultados da requisição.

Funções Disponíveis

`http.request(method, url, body, headers, options)`

Função genérica para realizar requisições HTTP com qualquer método.

Parâmetros:

- `method` (string): Método HTTP (GET, POST, PUT, DELETE, PATCH, HEAD)
- `url` (string): URL do endpoint
- `body` (string, opcional): Corpo da requisição
- `headers` (tabela, opcional): Cabeçalhos HTTP como pares chave-valor
- `options` (tabela, opcional): Opções adicionais da requisição

Opções disponíveis:

- `verify` (boolean): Verificar certificados SSL (padrão: true)
- `charset` (string): Charset para decodificação da resposta (padrão: "utf-8")

Aviso de Segurança: Desabilitar a verificação SSL (`verify = false`) expõe a conexão a ataques "man-in-the-middle" e deve ser usado **apenas** em ambientes de desenvolvimento ou teste controlados. Nunca utilize esta opção em produção com servidores reais ou com dados sensíveis.

Valor de retorno: Retorna uma tabela com:

- `status` (number): Código de status HTTP
- `body` (string): Corpo da resposta

Exemplo:

```
local resultado = http.request("POST", "https://api.exemplo.com/dados",
    '{"nome": "teste", "valor": 123}',
```

```
    {["Content-Type"] = "application/json"},
    {verify = true, charset = "utf-8"}
)

print("Status:", resultado.status)
print("Resposta:", resultado.body)
```

http.get(url, body, headers, options)

Realiza uma requisição HTTP GET.

Parâmetros:

- `url` (string): URL do endpoint
- `body` (string, opcional): Corpo da requisição (raro para GET, mas suportado)
- `headers` (tabela, opcional): Cabeçalhos HTTP
- `options` (tabela, opcional): Opções adicionais

Exemplo:

```
local resultado = http.get("https://api.exemplo.com/usuarios/1",
    nil,
    {["Authorization"] = "Bearer token123"},
    {verify = true}
)

if resultado.status == 200 then
    print("Dados recebidos:", resultado.body)
else
    print("Erro:", resultado.status)
end
```

http.post(url, body, headers, options)

Realiza uma requisição HTTP POST.

Parâmetros:

- `url` (string): URL do endpoint
- `body` (string): Corpo da requisição (geralmente JSON ou dados de formulário)
- `headers` (tabela, opcional): Cabeçalhos HTTP
- `options` (tabela, opcional): Opções adicionais

Exemplo:

```
local dados_json = '{"nome": "Novo Usuário", "email": "usuario@exemplo.com"}'

local resultado = http.post("https://api.exemplo.com/usuarios",
    dados_json,
    {
        ["Content-Type"] = "application/json",
        ["Authorization"] = "Bearer token123"
    }
)

if resultado.status == 201 then
    print("Usuário criado com sucesso!")
    print("Resposta:", resultado.body)
else
    print("Falha ao criar usuário. Status:", resultado.status)
end
```

`http.put(url, body, headers, options)`

Realiza uma requisição HTTP PUT para atualizar recursos.

Parâmetros:

- `url` (string): URL do endpoint
- `body` (string): Corpo da requisição com dados atualizados
- `headers` (tabela, opcional): Cabeçalhos HTTP
- `options` (tabela, opcional): Opções adicionais

Exemplo:

```
local dados_atualizados = '{"nome": "Usuário Atualizado", "ativo": true}'

local resultado = http.put("https://api.exemplo.com/usuarios/1",
    dados_atualizados,
    {
        ["Content-Type"] = "application/json",
        ["Authorization"] = "Bearer token123"
    }
)
```

```
if resultado.status == 200 then
    print("Usuário atualizado com sucesso!")
else
    print("Falha na atualização. Status:", resultado.status)
end
```

http.delete(url, body, headers, options)

Realiza uma requisição HTTP DELETE para remover recursos.

Parâmetros:

- `url` (string): URL do endpoint
- `body` (string, opcional): Corpo da requisição (opcional para DELETE)
- `headers` (tabela, opcional): Cabeçalhos HTTP
- `options` (tabela, opcional): Opções adicionais

Exemplo:

```
local resultado = http.delete("https://api.exemplo.com/usuarios/1",
    nil,
    {["Authorization"] = "Bearer token123"}
)

if resultado.status == 204 then
    print("Usuário removido com sucesso!")
else
    print("Falha ao remover usuário. Status:", resultado.status)
end
```

http.head(url, body, headers, options)

Realiza uma requisição HTTP HEAD para obter apenas cabeçalhos.

Parâmetros:

- `url` (string): URL do endpoint
- `body` (string, opcional): Corpo da requisição
- `headers` (tabela, opcional): Cabeçalhos HTTP
- `options` (tabela, opcional): Opções adicionais

Exemplo:

```
local resultado = http.head("https://api.exemplo.com/usuarios/1",
    nil,
    {["Authorization"] = "Bearer token123"}
)

print("Status da verificação:", resultado.status)
-- A resposta HEAD geralmente não tem corpo
```

Informações Adicionais

URL Automática

Se a URL não contiver um esquema (como `http://` ou `https://`), o sistema automaticamente adiciona `http://` como prefixo.

```
-- Estas duas chamadas são equivalentes:
local r1 = http.get("api.exemplo.com/dados")
local r2 = http.get("http://api.exemplo.com/dados")
```

Verificação SSL

Por padrão, a verificação de certificados SSL está habilitada. Para desabilitar (útil em ambientes de desenvolvimento/teste):

Atenção: Desabilitar a verificação SSL (`verify = false`) expõe a conexão a ataques "man-in-the-middle" e deve ser usado **apenas** em ambientes de desenvolvimento ou teste controlados. Nunca utilize esta opção em produção com servidores reais ou com dados sensíveis.

```
local resultado = get("https://servidor-local.com",
    nil,
    nil,
    {verify = false}
)
```

Charsets

É possível especificar um charset diferente para decodificar a resposta:

```
local resultado = get("https://api.exemplo.com/dados",
    nil,
    nil,
    {charset = "iso-8859-1"}
)
```

Exemplo de Uso

```
-- Exemplo de monitoramento de API
function verificar_saude_api()
    local resultado = http.get("https://api.exemplo.com/health",
        nil,
        {["User-Agent"] = "MonstaAgent/1.0"}
    )

    if resultado.status == 200 then
        local dados = resultado.body
        -- Processar resposta JSON se necessário
        print("API está saudável")
        return true
    else
        print("API com problemas. Status:", resultado.status)
        return false
    end
end
```

Notas Importantes

1. **Timeout**: O timeout padrão é o timeout global configurado para scripts Lua no sistema.
2. **Performance**: Use conexões persistentes quando fizer múltiplas requisições para o mesmo servidor.
3. **Segurança**: Nunca desabilite a verificação SSL em produção sem necessidade.

Revision #1
Created 3 February 2026 16:42:46 by Marc
Updated 3 February 2026 16:43:06 by Marc