

# REST-Anwendungen - Eigenentwicklung und externe Nutzung

---

REST: Ein Architekturstil für das moderne Web

REST kurz für „Representational State Transfer“, ist ein Architektur-Stil, der im Laufe der Jahre aus einem einfachen Konzept zu einem Eckpfeiler der modernen Web-Entwicklung gewachsen ist. Der Schwerpunkt liegt in **Maschine-zu-Maschine** Kommunikation. Es werden Textnachrichten nach dem HTTP-Protokoll ausgetauscht. In einer RESTful-Architektur wird jede Ressource, auf die zugegriffen werden soll, durch eine eindeutige Adresse (URI –Uniform ResourceIdentifier) identifiziert. Das REST-Paradigma entwickelte sich aus dem 1994 von Roy Fielding entworfenen HTTP ObjectModel.

Wir werden den Apache2-Web-Server dazu benutzen einen einfachen Web-Service zu integrieren. Die Anfrage (Request) erfolgt mittels HTTP-Request und der Web-Service gibt ein JSON-Objekt zurück. JSON ist sehr einfach von einem eigenen Programm zu lesen und zu interpretieren, man kann aber auch andere Formate z. B. XML nutzen.

## Anwendung Temperaturumrechnung Celsius <--> Fahrenheit

Wir erstellen eine PHP-REST-Anwendung um eine Temperatur von Grad C° in Fahrenheit oder von Fahrenheit in Grad C° umzurechnen.

Dazu verwenden Sie folgende URL: `./tempUmrechner.php?wert=xxx.xx&einheit=celsius`.

Wir verwenden folgende Formel:  $^{\circ}\text{F} = ^{\circ}\text{C} \times 1,8 + 32$  oder  $^{\circ}\text{C} = (^{\circ}\text{F} - 32) / 1,8$ .

## Umsetzung

Erstellen Sie einen Ordner "rest" im Web-Verzeichnis. Erstellen Sie eine Datei `tempUmrechner.php` im neuen Verzeichnis und fügen folgenden Code ein:

```
<?php
    $einheit = "";
    $wert = "";

    // Definiere Funktion, um JSON-Fehler zurückzugeben
    function sendError($statusCode, $message) {
        header('Content-Type: application/json');
        http_response_code($statusCode);
        $error_result = array("status" => "error", "message" => $message);
        echo json_encode($error_result);
        exit;
    }

    // 1. Prüfen auf benötigte Parameter
    if (!isset($_GET['wert']) || !isset($_GET['einheit'])) {
        sendError(400, "Fehlende Parameter. Benötigt werden 'wert' und 'einheit'.");
    }
```

```

    }

    // 2. Prüfen auf gültigen Wert
    $wert = $_GET['wert'];
    if (!is_numeric($wert)) {
        sendError(400, "Der Parameter 'wert' muss numerisch sein.");
    }

    // 3. Prüfen auf gültige Einheit
    $einheit = strtolower($_GET['einheit']); // Einheit in Kleinbuchstaben
    konvertieren für robustere Prüfung
    if ($einheit !== 'celsius' && $einheit !== 'fahrenheit') {
        sendError(400, "Ungültiger Wert für 'einheit'. Erlaubt sind 'celsius' oder
        'fahrenheit'.");
    }

    if($einheit == "celsius"){
        header('Content-Type: application/json');
        http_response_code(200);
        $result = array(
            "einheit" => "fahrenheit",
            "wert" => ($wert * 1.8 + 32)
        );
        echo json_encode($result);
        exit;
    }elseif($einheit == "fahrenheit"){
        header('Content-Type: application/json');
        http_response_code(200);
        $result = array(
            "einheit" => "celsius",
            "wert" => (($wert - 32) / 1.8)
        );
        echo json_encode($result);
        exit;
    }
}

```

Beachten Sie, dass wir im Header den 'Content-Type: application/json' zurück geben und das Rückgabe-Array in ein JSON-Objekt umwandeln.

Jetzt können Sie die Anwendung testen. Rufen Sie dazu im Browser die Anwendung auf - <http://<ip-webcontainer>/rest/tempUmrechner.php?wert=20&einheit=celsius>. Wir fragen damit die Umrechnung von 20°Celsius in Fahrenheit ab.

Die Antwort ist:

```

{
    "einheit": "fahrenheit",
    "wert": 68
}

```

## REST-Client

Ein REST-Client kann in beliebiger Sprache geschrieben werden. Der Client muss sich nur an die Form des Request (HTTP oder HTTPS) halten und Kenntnis über den Aufbau der Antwort haben.

Hinweis: Die Beispiele können Sie auch auf Ihrem Client-PC ausführen.

### Java-Beispiel

```
package va_rest;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class TempUmrechnerApiCaller {
    public static void main(String[] args) {
        try {
            // Ersetze mit der tatsächlichen URL deiner API
            String urlString = "http://<ip-webcontainer>/rest/tempUmrechner.php?wert=20&einheit=celsius";
            URL url = new URL(urlString);
            HttpURLConnection connection =
                (HttpURLConnection)url.openConnection();

            BufferedReader in = new BufferedReader(
                new InputStreamReader(connection.getInputStream()));

            int responseCode = connection.getResponseCode();
            System.out.println("Response Code: " + responseCode);

            String inputLine;
            StringBuffer content = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine);
            }
            in.close();

            // Hier kannst du den JSON-String parsen und das Ergebnis verarbeiten
            System.out.println(content.toString());

        } catch (IOException e) {
            System.out.println(e);
        }
    }
}
```

Zum Arbeiten mit JSON in Java benötigen Sie eine gesonderte Bibliothek; ich nutze GSON von Google.

## Python-Beispiel

```
import requests
import json
from pprint import pprint

# -----
# Konfiguration
# -----

# HINWEIS: Ersetzen Sie <ip-webcontainer> durch die tatsächliche IP-Adresse Ihres
# LXC-Containers
BASE_URL = "http://<ip-webcontainer>/rest/tempUmrechner.php"

def get_temperature_conversion(value, unit):
    """
    Ruft den REST-Service zur Temperaturumrechnung auf.

    :param value: Der umzurechnende Temperaturwert (float oder int).
    :param unit: Die Ausgangseinheit ('celsius' oder 'fahrenheit').
    :return: Das umgerechnete Ergebnis als Diktionär oder Fehlermeldung.
    """

    # URL-Parameter definieren
    params = {
        "wert": value,
        "einheit": unit
    }

    print(f"Sende Anfrage: {BASE_URL}?wert={value}&einheit={unit}")

    try:
        # Führe die GET-Anfrage aus
        response = requests.get(BASE_URL, params=params)

        # 1. Prüfe den HTTP Status Code
        if response.status_code == 200:
            print("Status: OK (200)")
            # Lese und gebe das JSON-Ergebnis zurück
            return response.json()

        elif response.status_code == 400:
            # Fehlerhafter Request, z.B. ungültige Einheit oder Wert
            print(f"Status: Bad Request ({response.status_code})")

            # Die API gibt im Fehlerfall einen erklärenden JSON-Body zurück (Ihre
            # Verbesserung!)
            try:
                error_data = response.json()
```

```

        print("API-Fehlermeldung:")
        # Nutze pprint für eine saubere Ausgabe des Fehlers
        pprint(error_data)
        return None
    except json.JSONDecodeError:
        print("API-Fehler konnte nicht als JSON gelesen werden.")
        return None

    else:
        # Unerwarteter Status Code (z.B. 500 Server Error)
        print(f"Unerwarteter Status: {response.status_code}")
        return None

    except requests.exceptions.ConnectionError:
        print("FEHLER: Konnte keine Verbindung zum Server herstellen. IP/Port
prüfen.")
        return None
    except requests.exceptions.RequestException as e:
        print(f"Ein Fehler ist aufgetreten: {e}")
        return None

# -----
# Beispiele
# -----

if __name__ == "__main__":

    print("--- 1. Erfolgreicher Aufruf (20°C in F) ---")
    result_ok = get_temperature_conversion(value=20, unit="celsius")
    if result_ok:
        print(f"Ergebnis: {result_ok['wert']} {result_ok['einheit'].upper()}")

    print("\n" + "="*50 + "\n")

    print("--- 2. Erfolgreicher Aufruf (68°F in C) ---")
    result_ok_reverse = get_temperature_conversion(value=68, unit="fahrenheit")
    if result_ok_reverse:
        print(f"Ergebnis: {result_ok_reverse['wert']}
{result_ok_reverse['einheit'].upper()}")

    print("\n" + "="*50 + "\n")

    print("--- 3. Fehlgeschlagener Aufruf (Ungültiger Wert) ---")
    # Der Wert "abc" ist nicht numerisch. Dies sollte 400 Bad Request auslösen.
    get_temperature_conversion(value="abc", unit="celsius")

    print("\n" + "="*50 + "\n")

    print("--- 4. Fehlgeschlagener Aufruf (Ungültige Einheit) ---")
    # Die Einheit ist unbekannt. Dies sollte 400 Bad Request auslösen.
    get_temperature_conversion(value=10, unit="kelvin")

```

## Fazit

Mit REST-Web-Services haben Sie ein mächtiges Werkzeug an der Hand das mit einem einfachen Protokoll (HTTP) arbeitet. Sie können ihre eigenen Service definieren, Daten aus der Datenbank für Antworten nutzen oder Daten austauschen.

Entscheidend ist, dass Programme unabhängig von einander arbeiten können - unablässig für Anwendungen mit Microservice-Architektur. Dies ist ein architektonischer Ansatz, bei dem eine große Anwendung in eine Sammlung kleiner, lose gekoppelter und unabhängig voneinander bereitstellbarer Dienste aufgeteilt wird

---

## Lizenz

Dieses Werk ist lizenziert unter der **Creative Commons - Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz**.

[Zum Lizenztext auf der Creative Commons Webseite](#)