```
/*
  Beschreibung
  Ein einfacher Webserver, der ein paar statische (nicht veraennderliche) Informationen  und die Zeit ausgibt.
  Das Ethernet-Shield (der Ethernet-Huckepackbausein) ist ueber die Pins 10, 11, 12, 13 angeschlossen.
  Zusaetzlich wird noch Pin 4 fuer die SD-Karte verwendet. Daher stehen diese Pins fuer Anwendungen nicht zur Verfuegung.
*/

// Einbindung einiger Bibliotheken
#include <SPI.h>
#include <Ethernet.h>

// Geben Sie eine MAC-Adresse ( = media access control address, Medien Zugangssteuerungsadresse) ein, am besten die,
// die auf dem Ethernet-Shield steht, ansonsten eine beliebige.
// Die MAC-Adresse ist eine Hardwareadresse, die auf der Welt nur einmal vergeben wird.
// Sie wird normalerweise vom Hersteller des Geraets fest in dieses eingebrannt und kann nicht veraendert werden.
// Sie darf aber im Heimnetzwerk sonst nicht vorkommen.
// Bei unserem Shield kann sie frei vergeben werden. (Chinesische Massenware)
// Die IP-Adresse haengt von Ihrem Heimnetzwerk ab.
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
// Aendern Sie das letzte Byte (0xED) in 0xF0, 0xF1, 0xF2, ...
IPAddress ip(192, 168, 178, 25);   // Aendern Sie die Zehl 25 in eine passende!!!

// Initialisierung (=erstmalige Inbetriebnahme) der Ethernet server Bibliothek
// mit der IP Adresse und dem Port (der Eingangspforte), die Sie wuenschen
// (port 80 ist die Voreinstellung fuer HTTP):
EthernetServer server(80);

long zeit; // Hilfsvariable, um die Startzeit zu speichern
boolean currentLineIsBlank;   // Hilfsvariable, um das Ende der HTTP-Anfrage zu erkennen
char c;                       // Hilfsvariable, um ein gelesenes Zeichen zwischenzuspeichern
// Diese Variablenvereinbarungen muessen vor dem Setup stehen!

void setup() {
  // Der folgende Code wird einmal am Anfang ausgefuehrt:
  zeit = millis();
  // Oeffne die serielle Schnittstelle und warte, bis sie bereit ist:
  Serial.begin(9600);
```

```
  while (!Serial) {    // Verweile hier, solange die serielle Schnittstelle nicht (! = logisches "Nicht") bereit ist.
    ;   // Tue nichts (leerer Befehl, nur bestehend aus einem ;)
        // Wird nur fuer den Arduino Leonardo gebraucht, schadet aber auch sonst nicht
  }

  // Starte die Ethernet Verbindung und den Server:
  Ethernet.begin(mac, ip);
  server.begin();
  // Gib die IP-Adresse ber die serielle Schnittstelle aus
  Serial.print("Der Server hat die IP-Adresse: ");
  Serial.println(Ethernet.localIP());
  Serial.println("-----------------------------------------------");
}

void loop() {
  // Lausche, ob jemand (client = Kunde) die Seite aufrufen will
  EthernetClient client = server.available();
  if (client) {
    Serial.println("Ein neuer Client (=Kunde) fragt an:");
    // Eine http-Anfrage wird mit einer leeren Zeile abgeschlossen
    currentLineIsBlank = true;                    // Noch steht nichts in der (empfangenen) neuen Zeile
    while (client.connected()) {                  // Solange der Client angeschlossen ist
      if (client.available()) {                   // Wenn noch Zeichen vom Client empfangen werden koennen
        c = client.read();                        // Lies ein einzelnes Zeichen
        Serial.write(c);                          // Gib dieses Zeichen ueber die Serielle Schnittstelle aus
        //  ----------------------------------------------------------------
        // Was bis zu den naechsten --- steht, wird nur einmalig am Ende der HTTP-Anfrage ausgefuehrt
        // Solange der Server (=Arduino) noch Zeichen vom Client empfaengt, wird dieser Teil uebersprungen

        if (c == '\n' && currentLineIsBlank) {    // Ein Zeilenvorschub "\n", gefolgt von einer leeren Zeile ist das Ende der An
          // Das Ende der Anfrage ist nun erkannt, das Lauschen ist zu Ende, jetzt ist der Server dran
          Serial.println("Das ist das Ende der Anfrage");
          Serial.println("Nun sendet der Arduino an den Browser die Webseite");
          // Sende einen Standard http response header (Standard Antwortkopf)
          // Der Kopf ist kein Seiteninhalt, sondern sagt dem Browser nur, nach welchen Regeln er die Sache darzustellen hat.
          client.println("HTTP/1.1 200 OK");          // Ich (Arduino) sende nun (an den Browser) nach dem HTTP/1.1 Standard
          client.println("Content-Type: text/html");  // Der Inhalt dessen, was der Arduino sendet, ist HTML-Text
          client.println("Connection: close");        // Nach Beendigung der Antwort wird der Server (=Arduino) die Verbindung sc
          client.println("Refresh: 10");              // Die Seite wird automatisch alle 10 Sekunden aktualisiert
```

```
          client.println();
          // Mit dieser Leerzeile ist der Antwortkopf abgeschlossen. Jetzt folgt der eigentliche Seiteninhalt
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          client.println("Hier ist der Arduino<br>&Ouml;ffnen Sie auch den seriellen Monitor beim Arduino.");
          client.println("<br>Viele Gr&uuml;&szlig;e ... Norbert ");
          client.println("<br />Das Programm l&auml;uft seit ");
          client.print(String((millis()-zeit)/1000.0));
          client.println(" Sekunden<br>");
          client.println("</html>");
          break;
        }
        //  ---------------------------------------------------------------

        // Jede vom Client empfangene Textzeile endet mit \r\n ("\r" = Wagenruecklauf (reurn), "\n" = neue Zeile)
        if (c == '\n') {
          // Das Zeichen fuer neue Zeile wurde empfangen, d.h. noch ist die neue Zeile leer
          currentLineIsBlank = true;
        }
        else if (c != '\r') {     // Das Zeilenende ist noch nicht erreicht,
          // also war c ein Textzeichen, und damit ist die Zeile nicht leer
          currentLineIsBlank = false;
        }
      }         // Ende von "if (client.available())"
    }           // Ende von "while (client.connected())"

    delay(1);        // Gib dem Web-Browser Zeit, die Daten zu empfangen
    // Schliesse die Verbindung:
    client.stop();        // Damit schliesst der Arduino (=Server) die Verbindung zum Client (=Kunden)
    Serial.println("client disconnected");
    Serial.println("<><><><><><><><><><><><><><><><><><><><><><><><><><><><>");
    Serial.println();
  }   // Ende der Client-Schleife
  Serial.println("Ich (Arduino) drehe Ehrenrunden");
  delay(1000);
}  // Ende des Loop
```