



DASAR-DASAR PEMROGRAMAN WEB

kuliah telegram
(mini ebook)



dosen:

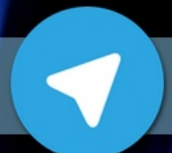
Pak Lebah / Mr. Bee

waktu:

**5 Mei 2017
20:00 WIB**



di: **t.me/pascalid**



DASAR-DASAR PEMROGRAMAN WEB

mini e-book

Topik Bahasan:

- Pengenalan Pemrograman di Awan dengan CodeAnywhere.
- Pengantar Pemrograman Web dengan Bahasa Pascal.

Tempat & Waktu:

- Telegram: t.me/pascalid
- Hari: Jumat, 5 Mei 2017
- Pukul: 20.00 WIB

Dosen:

Pak Lebah (Mr. Bee)

- telegram: @paklebah
- web: pak.lebah.web.id

Tagar:

[#kulgram](#) [#paklebah](#) [#dasarweb](#)

Daftar Isi

Aturan & Tata Tertib Kulgram	1
Tentang Dosen.....	1
I. Apa itu Program Web?.....	2
A. Internet dan Web	2
B. HTML dan Kawan-Kawan	4
C. Bagaimana Web Bekerja?	6
D. Program Web	9
II. Menyiapkan VPS.....	16
A. Menggunakan layanan CodeAnywhere.....	16
1. Mendaftar di layanan CodeAnywhere	16
2. Membuat VPS di CodeAnywhere	17
3. Kekurangan dan kelebihan.....	19
B. Memasang server web dan Pascal	20
1. Pengaturan awal VPS	20
2. Pemasangan server web	23
3. Pemasangan Free Pascal.....	24
III. Pascal dan Web	25
A. Hello World!	25
B. Pengaturan yang memudahkan	26
1. Pengaturan server web	26
2. Pengaturan Pascal	31
C. Menampilkan keluaran	32
1. Menampilkan informasi lingkungan.....	34
2. Catatan untuk proses awakutu	37
3. Membuat unit webUtils.pas	40

D. Menerima masukan	44
1. Menampilkan antarmuka masukan	45
2. Membaca data masukan web	51
3. Memilih dan memilah data masukan	54
E. Mengolah data	59
1. Mengolah data dengan memori bersama	59
2. Mengolah data masukan	61
3. Menggunakan sesi web	66
IV. Tanya Jawab & Diskusi	73
V. Penutup	74

...

Aturan & Tata Tertib Kulgram

I. Istilah

- Dosen adalah pemateri kulgram.
- Moderator adalah pengarah kulgram.
- Topik adalah materi bahasan kulgram.
- Peserta adalah anggota grup yang mengikuti kulgram.

II. Sesi Kulgram

1. Pembukaan.
 - a. Moderator membuka dengan penyampaian profil dosen dan topik kulgram.
 - b. Moderator menandai sesi kulgram dengan tagar [#kulgram](#) [#judul](#) [#dosen](#).
2. Kulgram topik oleh dosen.
 - a. Sesi kulgram bersifat searah. Dosen menyampaikan, peserta menyimak.
 - b. Peserta tidak boleh mengobrol dengan peserta lain atau dosen selama kulgram.
 - c. Dosen berhak membuka sesi tanya jawab terbatas di tengah kulgram.
 - d. Moderator berhak menyela dan menghentikan kulgram dosen jika diperlukan.
3. Tanya jawab (tanja) dosen dengan peserta.
 - a. Tanja bersifat dua arah. Peserta bertanya, dosen menjawab.
 - b. Jika tidak bisa atau tidak bersedia menjawab, dosen bisa memberi kesempatan pada peserta lain untuk membantu menjawab pertanyaan.
 - c. Peserta tidak boleh ikut menjawab pertanyaan tanpa ijin dan kesempatan dari dosen atau moderator.
 - d. Pertanyaan peserta harus sesuai dengan topik yang sedang dibahas.
 - e. Setiap pertanyaan peserta harus ditandai dengan tagar [#tanya](#).
 - f. Tanja dilaksanakan dalam suasana serius tapi santai dan sopan.
 - g. Moderator mengatur proses tanja agar berlangsung dengan tertib dan teratur.
 - h. Moderator berhak menegur dan menendang peserta yang dianggap tak mematuhi aturan dan tata tertib kulgram.
4. Penutup.

Moderator menutup kulgram dengan memberikan kesimpulan ringkas topik kulgram.

III. Tindak Lanjut

1. Panitia akan menerbitkan rekaman kulgram, berikut tanya jawab yang telah disunting seperlunya, sebagai bahan belajar untuk publik.
2. Setelah sesi kulgram selesai, bisa dilanjutkan dengan diskusi bebas tanpa dipandu moderator, baik dengan atau tanpa dosen.

...

Tentang Dosen

Pak Lebah (*Mr. Bee*) adalah seorang penggemar pemrograman. Namun kurang tepat jika Pak Lebah disebut seorang pemrogram profesional karena sudah cukup lama profesi utamanya bukan lagi membuat program. Pak Lebah lebih pantas disebut sebagai pemrogram iseng karena membuat program baginya lebih sebagai hobi yang dilakukan tidak dengan serius di waktu senggang.

Banyak bahasa pemrograman yang pernah dipelajari Pak Lebah, tapi bahasa pemrograman yang paling Pak Lebah sukai dan pahami adalah bahasa (*object*) Pascal. Dulunya Pak Lebah adalah seorang pengguna (Borland) Delphi, hingga kemudian bertemu dengan Free Pascal dan Lazarus IDE. Saat masih aktif sebagai pemrogram profesional dengan menggunakan Delphi, bidang pemrograman yang digelutinya adalah aplikasi *desktop* di Windows. Namun setelah itu dia berpindah ke aplikasi web, dengan menggunakan Free Pascal dan Lazarus IDE di Linux. Hingga akhirnya dia meninggalkan pemrograman untuk bekerja di bidang lainnya.

Walau pun demikian, Pak Lebah tetap menyukai pemrograman karena baginya membuat program —dia lebih suka menyebutnya “koding”— adalah suatu kegiatan kreatif yang menyenangkan dan merupakan “panggilan jiwa”. Karena itu jangan heran jika sedang stres, Pak Lebah justru bikin program agar pikirannya kembali segar. Kegemarannya selain bikin program adalah membaca, terutama seputar *software development* dan perkembangan dunia sains dan teknologi.

Saat ini, Pak Lebah sedang getol belajar bahasa pemrograman baru dari Apple, yaitu bahasa Swift. Sambil belajar, dia menuliskan juga apa yang sedang dipelajarinya ke blognya yang khusus untuk Swift di medium.com/@pak.lebah. Jika Anda juga tertarik pada bahasa baru ini, silakan kunjungi saja blog tersebut. Atau jika Anda seorang blogger, Anda bisa ikuti blog Pak Lebah di paklebah.tumblr.com.

Tanpa banyak basa-basi lagi, mari kita mulai saja kulgram hari ini oleh Pak Lebah dengan judul **Dasar-Dasar Pemrograman Web**. Sebagai penanda untuk pencarian, kulgram kali ini ditandai dengan tagar [#kulgram](#) [#dasarweb](#) [#paklebah](#).

Oh ya... dalam penyusunan dan penyampaian kulgram ini, Pak Lebah dibantu rekan-rekan dari Komunitas Pascal Indonesia, antara lain: Tigor Manurung, Moh. Riad, Dio Affriza, Mas Kofa, dan semua pihak yang tak bisa disebutkan satu-persatu di sini. Terima kasih.

...

I. Apa itu Program Web?

Sekarang rasanya nyaris tak ada orang yang tidak mengenal internet. Berdasarkan survei di akhir tahun 2016, dari sekitar 7,5 milyar manusia di Bumi, ada sekitar 3,7 milyar (atau sekitar 50%) yang telah mengakses internet. Di tahun 2020 nanti, diperkirakan ada sekitar 7,8 milyar manusia di Bumi, sekitar 5 milyarnya adalah pengguna internet (atau sekitar 64%). Dan ini diperkirakan akan terus meningkat. Artinya, internet telah menjadi jalur komunikasi manusia yang utama. Dan penguasaan teknologi internet akan penting bagi siapapun, bahkan bagi orang yang profesinya bukan di bidang teknologi informasi. Apalagi bagi orang yang profesinya di bidang teknologi informasi, jelas harus paham betul.

A. Internet dan Web

Internet itu berbeda dengan web. Internet lebih besar daripada web dan web hanya bagian dari internet. Pada prinsipnya, internet itu ibaratnya adalah jalan yang menghubungkan satu rumah dengan rumah lainnya di seluruh dunia. Di internet, “jalan” adalah kabel, dan “rumah” adalah komputer, serta “kendaraan” yang berlalu lalang adalah informasi.

Alamat IP

Alamat IP (*internet protocol [IP] address*) adalah alamat setiap komputer yang terhubung ke internet. Dengan alamat IP inilah komputer bisa saling bertukar informasi satu dengan yang lain. Saat ini tersedia dua macam alamat IP, yaitu:

1. IP versi 4 (IPv4) yang berupa deret 4 kelompok angka desimal dipisahkan tanda titik seperti `172.17.1.7` yang mampu menampung hingga 4,3 milyar ($4,3 \times 10^9$) alamat.
2. IP versi 6 (IPv6) yang berupa deret 8 kelompok angka heksadesimal dipisahkan tanda titik dua seperti `21da:00d3:0000:2f3b:02aa:00ff:fe28:9c5a` yang mampu menampung hingga $3,5 \times 10^{38}$ alamat.

Internet saat ini umumnya masih menggunakan alamat IPv4. Tetapi karena keterbatasan jumlah bilangan yang bisa ditampung oleh IPv4 maka internet ke depan akan beralih ke IPv6 secara bertahap.

Domain

Alamat IP yang berupa deretan angka-angka itu susah dibaca dan diingat oleh manusia. Karena itu kemudian dibuatlah *domain* yang fungsinya menerjemahkan alamat IP berupa angka menjadi teks yang mudah bagi manusia. Contoh, alamat IP `203.119.112.47` dapat diakses dengan alamat *domain* `www.pandi.id`.¹ Jadi lebih mudah bukan?

¹ pandi.id adalah badan resmi yang mengelola pendaftaran nama *domain* di Indonesia.

Domain terdiri dari 3 bagian utama, yaitu:

1. *Top Level Domain* (TLD) yaitu bagian akhir dari sebuah *domain*.
2. Nama *domain* yaitu bagian tengah dari sebuah *domain*.
3. *Subdomain* (awalan *domain*) yaitu bagian awal dari sebuah *domain*.

Misalnya pada contoh www.pandi.id sebelumnya, 'id' adalah *top level domain*, 'pandi' adalah nama *domain*, 'www' adalah *subdomain*. Bagian *subdomain* kadang tak harus disertakan jika mengakses laman utama atau beranda sebuah situs web. Komputer yang bertugas menerjemahkan *domain* disebut dengan DNS (*domain name server*).

Protocol

Protokol adalah metode atau cara berkomunikasi di internet. Ibaratnya seperti manusia bisa berkomunikasi dengan lisan (bicara), tulisan, bunyian, morse, dan sebagainya, yang masing-masing punya tujuan berbeda, maka seperti itulah fungsi protokol di internet. Ada banyak jenis protokol internet, namun cukup kita ketahui yang umum saja, antara lain:

- FTP (*file transfer protocol*) adalah metode untuk pertukaran berkas.
- SMTP (*simple mail transfer protocol*) adalah metode untuk mengirim surat elektronik.
- POP (*post office protocol*) adalah metode lain untuk menerima surat elektronik.
- IMAP (*internet message access protocol*) adalah metode untuk menerima surat elektronik.
- HTTP (*hyper-text transfer protocol*) adalah metode untuk mengambil berkas HTML (*hyper-text markup language*).

Beberapa protokol menyediakan pilihan metode yang lebih aman dari upaya pembajakan informasi di internet. Pengamanan informasi menggunakan teknik enkripsi (penyandian) dengan teknologi SSL (*secure sockets layer*). Protokol aman ditandai dengan akhiran 'S', misalnya FTPS² (FTP *with* SSL), HTTPS (HTTP *with* SSL), dan sebagainya.

URL

URL (*uniform resource locator*) adalah cara menulis alamat internet secara rinci. URL harus bersifat unik (hanya satu) di internet yang berfungsi untuk menunjukkan alamat komputer yang akan dituju dan bagaimana cara berkomunikasi dengannya. Struktur URL secara lengkap adalah sebagai berikut:

protocol://username:password@domain:port/path?query#fragment

Bagian yang berwarna biru adalah pemisah (*separator*), sedangkan yang berwarna merah adalah yang perlu diisi sesuai alamat atau nilai masing-masing. *Domain* bisa juga diisi dengan alamat IP jika alamat tersebut tidak memiliki *domain*.

Tak semua bagian URL harus diisi, bisa karena diisi secara otomatis atau bisa juga karena protokolnya tak membutuhkan. Contoh untuk mengakses berkas di internet dengan FTP, bisa ditulis <ftp://nama:sandi@172.17.1.7/folder> melalui peramban atau aplikasi FTP.

² FTPS berbeda dengan SFTP. SFTP adalah *SSH File Transfer Protocol* yaitu FTP di atas SSH.

Contoh untuk mengakses sebuah alamat web <http://www.pandi.id:80/index.html> cukup ditulis [pandi.id](http://www.pandi.id:80/index.html) saja. Ini karena sebagian alamat tersebut telah dilengkapi secara otomatis oleh peramban. URL disebut juga sebagai pranala atau tautan (*link*).

Web

Web adalah bagian dari internet yang berkomunikasi dengan protokol HTTP. Aplikasi yang umum digunakan untuk mengakses web disebut dengan peramban web (*browser*). Beberapa istilah umum seputar web, antara lain:

- Penyedia web (*web server*) adalah aplikasi yang melayani pertukaran informasi di internet dengan protokol HTTP.
- Laman web (*web page*) adalah berkas atau dokumen HTML yang disediakan oleh penyedia web (*web server*).
- Situs web (*web site*) adalah kumpulan laman web yang disediakan penyedia web dalam satu alamat atau *domain* yang sama.
- Layanan web (*web service*) adalah berkas atau dokumen yang disediakan penyedia web namun tidak dalam bentuk HTML melainkan berupa JSON atau XML.
- Aplikasi web (*web application*) adalah aplikasi penyedia berkas HTML yang dibuat secara dinamis dengan program komputer.
- Aplikasi pengguna web (*web client*) adalah aplikasi pengguna untuk bertukar informasi di internet dengan protokol HTTP.
- Peramban web (*web browser*) adalah aplikasi pengguna untuk menampilkan berkas HTML (atau bukan HTML) yang diambil dari penyedia web.

B. HTML dan Kawan-Kawan

Tidak ada web tanpa HTML. HTML adalah bahasa web dan satu-satunya jenis berkas yang isinya bisa dimengerti oleh peramban. HTML atau *hyper-text markup language* adalah struktur dokumen yang menjadi basis bentuk informasi di web. Namun HTML tak sendiri.

Berkas HTML

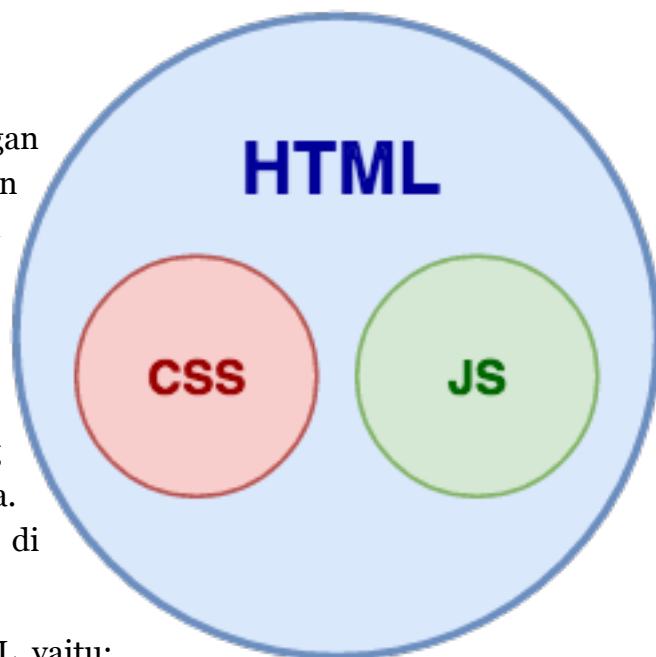
Berkas HTML adalah berkas teks dengan struktur tertentu dan biasanya dinamai dengan akhiran `.html` (atau `.htm` tapi ini tak disarankan). Struktur HTML menentukan bagaimana isi berkas ditampilkan oleh peramban. Bagian-bagian struktur HTML ditandai (*tag*) dengan pasangan `<tag>...</tag>`. Contoh, tanda `<p>` menunjukkan awal sebuah paragraf, sedang tanda `</p>` menunjukkan akhir dari paragraf.

Karena berkas HTML adalah berkas teks maka saat berkas HTML dibuka dengan aplikasi penyunting teks biasa, seluruh tanda tersebut muncul apa adanya. Namun saat berkas HTML yang sama dibuka dengan aplikasi peramban, maka berkas HTML tampil sebagai laman web dimana seluruh tanda HTML tidak lagi muncul sebab telah diterjemahkan sesuai dengan fungsinya masing-masing. Contoh, teks yang diapit tanda `<h1>...</h1>` akan ditampilkan sebagai judul besar (judul tingkat pertama).

Ada banyak tanda bagian HTML, mulai dari penanda judul, paragraf, tabel, gambar, dan lain sebagainya. Sebagai acuan, daftar tanda HTML bisa dipelajari di htmlreference.io yang juga disertai dengan contoh pemakaiannya.

Berkas Pendukung Gaya

Web saat ini tak semata hanya dibangun dengan HTML. HTML versi modern dilengkapi dengan pengaturan gaya (*style*) tampilan. Pengaturan gaya HTML disebut dengan CSS (*cascading style sheet*). Jika HTML menentukan bagian-bagian isi dokumen maka CSS menentukan bagaimana bagian-bagian tersebut ditampilkan atau dicetak, seperti warna teksnya, posisinya, dimensi panjang dan lebar, ukuran hurufnya, dan lain sebagainya. CSS mengatur bagaimana sebuah bagian (tanda di HTML) ditampilkan oleh peramban.



Ada 3 cara menerapkan CSS ke dalam berkas HTML, yaitu:

1. Menempel langsung ke tanda HTML (*inline*) dengan atribut `style`
Contoh: `<h1 style="color:blue;">Ini judul warna biru</h1>`
2. Sebagai bagian dalam berkas HTML (*internal*) dengan tanda `<style>...</style>`
Contoh: `<style>h1 {color: blue;}</style>`
3. Sebagai berkas terpisah yang dipanggil dari HTML (*external*) dengan tanda `<link>`
Contoh: `<link rel="stylesheet" href="my_styles.css">`
Aturan umum nama berkas CSS menggunakan akhiran `.css`

Ada banyak sekali kode pengaturan gaya di CSS. Cara penulisan kode CSS juga berbeda dengan HTML sehingga perlu waktu khusus untuk mempelajari CSS ini. Versi CSS terbaru bahkan telah mampu melakukan animasi. Sebagai acuan, daftar kode CSS bisa dipelajari di cssreference.io yang juga disertai dengan contoh pemakaiannya.

Berkas Pendukung Aksi

Selain dilengkapi dengan pengaturan gaya, HTML versi modern juga dilengkapi dengan pengaturan aksi. Pengaturan aksi dalam HTML menggunakan bahasa JavaScript (JS). Jika CSS memperkaya tampilan HTML maka JS menambah aksi (perilaku) pada HTML. JS modern sudah memiliki kemampuan yang cukup banyak dan rumit seperti *object oriented programming* (OOP), animasi, olah data, dan lain sebagainya.

Sebagaimana CSS, ada 3 cara menerapkan JS ke dalam berkas HTML, yaitu:

1. Menempel langsung ke tanda HTML (*inline*) dengan atribut aksinya.
Contoh: `<button type="button" onclick="alert('Hello!')">BUTTON</button>`
2. Sebagai bagian dalam berkas HTML (*internal*) dengan tanda `<script>...</script>`
Contoh: `<script>alert('Hello!');</script>`

3. Sebagai berkas terpisah yang dipanggil dari HTML (*external*) dengan atribut `src`

Contoh: `<script src="my_scripts.js"></script>`

Aturan umum nama berkas JS menggunakan akhiran `.js`

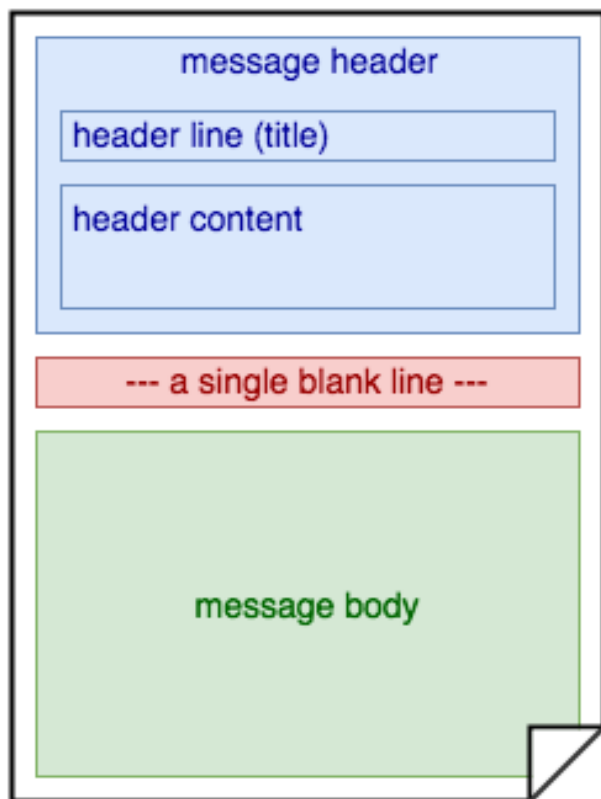
Mempelajari JS perlu waktu tersendiri yang di luar lingkup buku ini. Sebagai awalan, bisa dimulai dari www.w3schools.com/js

Perlu diingat bahwa CSS dan JS adalah pendukung terhadap HTML. Artinya CSS dan JS tidak bisa berdiri sendiri sebagai sebuah laman web, melainkan harus menempel pada induknya berupa berkas HTML.

C. Bagaimana Web Bekerja?

Walaupun terkesan besar dan rumit, cara kerja internet pada dasarnya masih menggunakan prinsip teknologi lama yaitu pertukaran pesan (informasi) antara klien dan server (*client and server*). Klien adalah pihak yang membutuhkan informasi, sedangkan server adalah pihak yang memiliki (menyimpan) informasi. Selaku pemilik informasi, server diharapkan selalu terhubung ke internet agar pengguna bisa mengambil informasi darinya kapan pun.

Demikian juga web. Bedanya, klien dan server di web berkomunikasi di internet dengan saling bertukar pesan dalam protokol HTTP. Secara umum, sisi klien web adalah aplikasi peramban (seperti Safari, Chrome, Firefox, Opera, Edge, dan sebagainya), sedang di sisi server adalah aplikasi server web (seperti Apache, Nginx, IIS, dan sebagainya).



HTTP Message Structure

Pesan HTTP

Pesan (*message*) HTTP adalah struktur pesan yang berlaku dalam protokol HTTP. Strukturnya seperti pada gambar di samping, terdiri atas 3 bagian, yaitu:

1. Kepala (*header*) yang terdiri dari:
 - judul (*line* atau *title*), dan
 - isi (*content*).
2. Pemisah pesan (*separator*) berupa sebuah baris kosong.
3. Badan (*body*) yang berisi pesan yang akan dikirimkan.

Di sisi pengguna, perambanlah yang bertugas menyusun dan mengirim pesan permintaan ini. Sedangkan di sisi penyedia, server web yang bertugas menyusun dan mengembalikan pesan balasan. Struktur pesan ini harus diikuti oleh kedua belah pihak agar komunikasi bisa terjalin dengan baik.

Sisi Depan Web

Lalu bagaimana alur pertukaran pesan antara peramban dengan server web? Mari kita perhatikan diagram berikut ini.

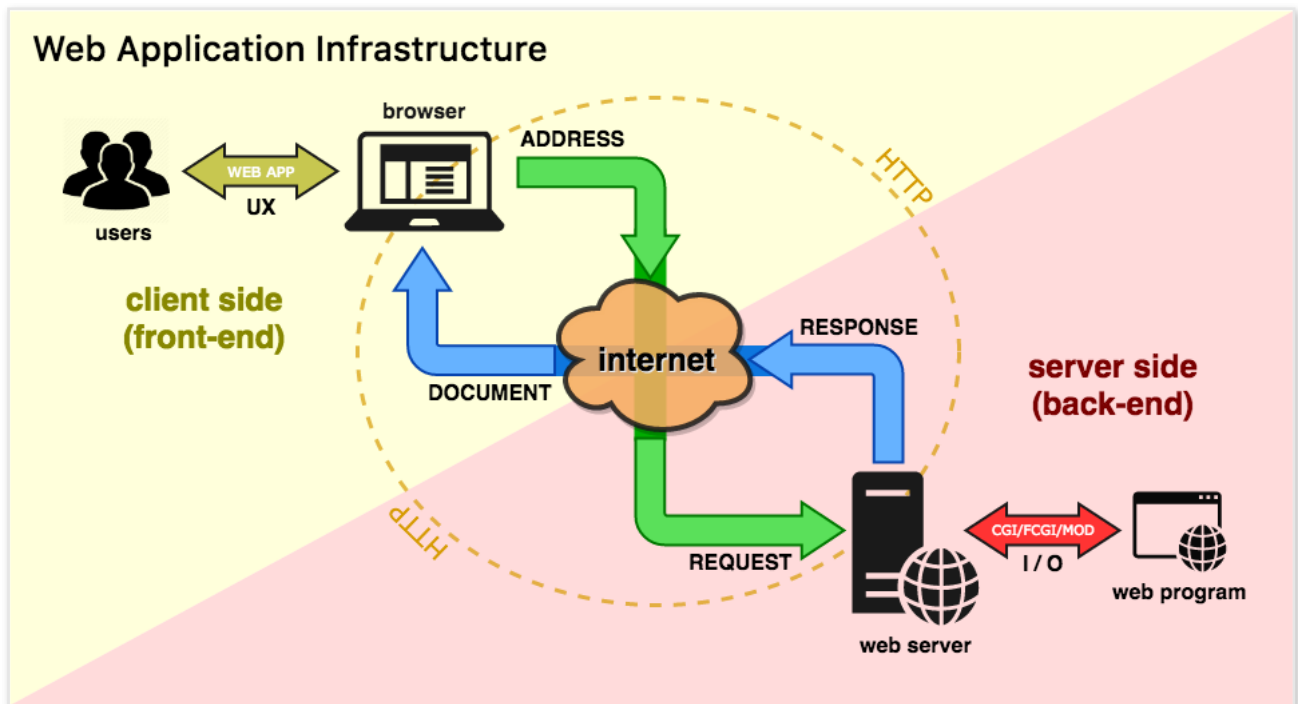


Diagram infrastruktur teknologi web.

Ada 2 (dua) sudut pandang dalam diagram di atas. Yang pertama adalah sudut pandang dari pengguna sisi depan (*front-end*), yang kedua adalah sudut pandang dari sisi belakang (*back-end*). Cara memahami kedua sudut pandang ini tentu berbeda.

Dari sisi depan, prosesnya seperti berikut:

1. Dimulai dengan pengguna memasukkan URL (alamat web) ke peramban.
2. Peramban menyusun pesan permintaan (*request message*) HTTP dan mengirimkannya ke alamat internet yang diberikan pengguna.
3. Peramban (dan pengguna) menunggu datangnya pesan balasan dari alamat web yang dituju. Peramban tidak tahu apa yang dilakukan komputer di alamat tersebut.
4. Ketika pesan balasan tiba, peramban membaca badan pesan yang berupa berkas HTML ke layar komputer. Jika HTML disertai dengan CSS maka gayanya diterapkan. Jika HTML disertai dengan JS maka aksinya dijalankan.
5. Jika pesan balasan tak kunjung tiba hingga waktu tertentu (*time-out*) atau terjadi kegagalan pengiriman pesan, peramban menampilkan pesan kesalahan pada pengguna.
6. Pengguna berinteraksi lebih lanjut dengan laman web yang dimunculkan peramban, atau pengguna menuju alamat web lainnya.
7. Demikian seterusnya berulang hingga pengguna berhenti menggunakan peramban.

Sudut pandang dari sisi depan ini seharusnya mudah dipahami karena semua orang jaman sekarang pasti pernah menggunakan peramban dan mengakses web.

Yang perlu diperhatikan adalah bahwa dari sudut pandang ini peramban yang melakukan proses penyusunan dan pembacaan isi pesan HTTP secara otomatis. Pengguna cukup mengetahui alamat dari situs web yang dituju dan menerima hasil berupa tampilan berkas HTML di peramban. Itu sebabnya dalam diagram di atas, di sisi depan hanya menyebut *address* (alamat web atau URL) dan *document* (berkas HTML).

Sisi Belakang Web

Melihat proses komunikasi dari sisi belakang akan nampak sedikit berbeda. Dari sisi ini nyaris tidak ada hal yang bisa dilihat mata. Proses di sisi belakang adalah seperti berikut:

1. Server web menunggu datangnya pesan permintaan HTTP (*request message*).
2. Ketika pesan datang, server web segera memproses pesan permintaan tersebut. Bagian badan dari pesan permintaan menunjukkan informasi atau data dari yang mengirim pesan (umumnya adalah peramban). Bagian judul kepala menunjukkan perintah yang terkait dengan informasi yang diminta. Sedangkan bagian isi kepala menunjukkan status terkait pesan yang dikirim.
3. Berdasarkan isi pesan permintaan yang diterima, server web kemudian menyiapkan pesan balasan (*response message*). Pesan balasan ini juga berupa pesan HTTP dengan struktur seperti di atas, hanya beda isi pesannya saja. Pada pesan balasan, bagian badan pesan berisi data berupa berkas HTML. Bagian judul kepala berisi status balasan terkait perintah dari pesan permintaan. Dan bagian isi kepala berisi status terkait data yang dikirim.
4. Jika informasi yang diminta adalah berkas HTML, server web akan menyertakan berkas tersebut ke dalam bagian badan pesan balasan. Jika berkas yang diminta tidak tersedia, server web akan membalas dengan pesan kesalahan.
5. Jika informasi yang diminta berasal dari program web, server web akan meneruskan pesan permintaan ke program tersebut, lalu menunggu program web memberikan balasannya. Metode komunikasi server web dengan program web tergantung pilihan jenis komunikasi program web yang dituju.
6. Jika balasan dari program web diterima, server web akan langsung meneruskannya ke alamat internet si pengirim pesan permintaan.
7. Jika balasan dari program web tak kunjung diterima hingga batas waktu tertentu atau terjadi kegagalan komunikasi dengan program web, server web mengirimkan pesan balasan dengan pesan kesalahan yang sesuai.
8. Demikian seterusnya berulang hingga server web berhenti bekerja.

Dari dua sudut pandang di atas, maka pemrograman web pun terdiri dari 2 bagian, yaitu: pemrograman web sisi depan (*front-end web programming*) dan pemrograman web sisi belakang (*back-end web programming*). Pemrograman web sisi depan berurusan dengan peramban dan interaksinya dengan pengguna (manusia), sementara pemrograman web sisi belakang lebih banyak berurusan dengan server web serta permintaan dan pengolahan informasi dari/ke pengguna.

Pemrograman web sisi depan lebih banyak mempelajari tentang penataan berkas HTML, pengaturan gaya laman yang bagus dengan CSS, aksi laman yang interaktif dengan JS, dan pertukaran data dengan server web (berdasarkan alamat). Sedangkan pemrograman web sisi belakang lebih banyak mempelajari tentang interaksi dan pertukaran data antara program web dengan server web, penyusunan berkas HTML (termasuk penyertaan CSS dan JS), serta pengolahan data.

Dalam buku ini, kita akan membahas dari sudut pandang sisi belakang, karena topik kita adalah pemrograman web dengan Free Pascal yang hanya tersedia di sisi belakang. Dan lebih khusus lagi pada langkah nomor 5 pada proses di atas, yaitu bagaimana program web berinteraksi dengan server web, membaca pesan permintaan, dan menyusun pesan balasan.

D. Program Web

Server web, sesuai namanya, hanya melayani permintaan dan mengembalikan tanggapan, dengan protokol HTTP. Server web tidak melakukan pengolahan informasi. Pengolahan informasi dilakukan dengan bantuan program lain, yang kita sebut dengan program web. Karena HTTP adalah protokol atau metode maka program web bisa dibuat dengan bahasa apa saja, tak ada ketentuan program web harus dibuat dengan bahasa tertentu. Program web bisa saja dibuat dengan *batch/shell script* (berkas `.bat/.sh`) di Windows/Linux.

Interaksi Program Web dengan Server Web

Yang terpenting adalah program web bisa berkomunikasi dengan server web. Bahasa pemrograman apapun yang bisa melakukan itu, bisa digunakan untuk membuat program web. Interaksi program web dengan server web secara umum ada tiga metode, yaitu:

- CGI (*common gateway interface*) yaitu server web menjalankan program setiap kali dibutuhkan. Ini adalah metode interaksi paling mudah dilakukan namun paling buruk kinerjanya.
- FCGI (*fast common gateway interface*) yaitu server web berkomunikasi dengan program web yang telah aktif menunggu (sebagai server aplikasi). Ini adalah metode interaksi yang lebih rumit dilakukan namun sangat baik kinerjanya.
- Sebagai tambahan yaitu program web yang dijalankan sebagai tambahan (*add-on* atau *extension*) terhadap server web. Misalnya, di server web Apache disebut sebagai *module*. Ini adalah metode yang cukup rumit dilakukan walaupun cukup cepat kinerjanya.

Karena lebih ditujukan pada pemula, dalam buku ini kita hanya akan membahas metode CGI saja. Metode lainnya bisa Anda pelajari setelah memahami CGI karena CGI adalah metode yang menjadi dasar bagi metode interaksi lainnya.

Pertukaran informasi antara server web dengan program web tersedia melalui dua jalur, yaitu melalui peubah lingkungan (*environment variable*) dan masukan/keluaran baku (*standard input/output*). Server web meneruskan pesan permintaan ke program web melalui peubah lingkungan dan/atau masukan baku, kemudian program web memberikan pesan balasan melalui keluaran baku.

Membaca Pesan Permintaan

Pesan permintaan (*request message*) yang dikirim peramban berisi banyak informasi yang menjelaskan permintaan. Pada bahasan sebelumnya telah dijelaskan bahwa pesan HTTP terdiri atas tiga bagian, yaitu: judul kepala (*header line*), isi kepala (*header content*), dan badan pesan (*request body*). Dalam pesan permintaan, informasi yang dikandung setiap bagian itu adalah sebagai berikut:

- Judul kepala berisi 3 informasi dalam susunan berikut:

[PERINTAH] [TUJUAN] HTTP/[VERSI]

- PERINTAH adalah perintah (*method*) yang menunjukkan apa yang dilakukan terhadap data dalam badan pesan. Perintah harus ditulis dalam huruf besar.
- TUJUAN adalah lokasi (*path*) berkas HTML atau program web yang dituju.
- VERSI adalah versi protokol HTTP yang digunakan. Versi yang umumnya berlaku saat ini adalah versi 1.1 sehingga bagian ini biasanya berisi teks HTTP/1.1

Contoh: GET /index.html HTTP/1.1

- Isi kepala berisi rincian permintaan HTTP dengan susunan berikut:

[Nama-Informasi]: [Informasi]

Ada banyak rincian permintaan HTTP, antara lain Host, Accept, Content-Type, Content-Length, Cookie, Referer, User-Agent, Connection, dan banyak lagi lainnya yang masing-masing mempunyai makna tersendiri. Contoh:

Host: www.google.com

User-Agent: Mozilla/5.0 AppleWebKit/537.36 Chrome/57.0.2987.98

Content-Type: application/x-www-form-urlencoded;

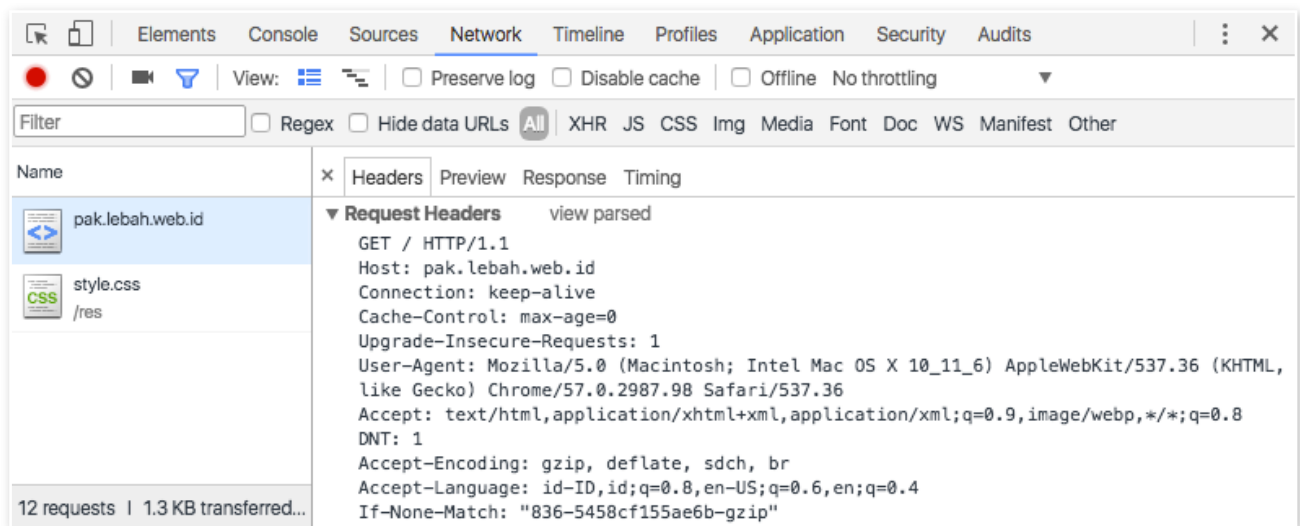
Content-Length: 114

Cookie: sessid=0888F207-1CB3-4C56-9E42-DFCF6B81B50E;

Daftar rincian permintaan sesuai ketentuan protokol HTTP versi 1.1 selengkapnya bisa dibaca pada laman berikut:

- <https://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html#sec4.5> (umum),
- <https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5.3> (permintaan),
- <https://www.w3.org/Protocols/rfc2616/rfc2616-sec7.html#sec7.1> (entitas).
- Badan pesan berisi data yang terkait dengan perintah permintaan. Jika hanya meminta berkas HTML saja, badan pesan boleh kosong. Jika meminta data ke program web, badan pesan bisa diisi dengan informasi data apa yang diinginkan, yang ditulis dengan susunan key=value dan tanda & (*ampersand*) sebagai pemisah jika informasi lebih dari satu. Contoh: username=simba&input_1=123

Jika kita menggunakan peramban maka seluruh informasi di atas secara otomatis diisi dan disusun oleh peramban. Kita bisa melihat pesan permintaan yang dikirim oleh peramban dengan cara menampilkan alat bantu pengembang. Di peramban umumnya menggunakan pintasan papan ketik **Ctrl+Alt+I** untuk menampilkan alat bantu pengembang (*developer tool*). Berikut adalah contoh tampilan pesan permintaan dalam alat bantu pengembang di peramban Chrome.



Pesan permintaan yang ditampilkan alat bantu pengembang di peramban Chrome.

Struktur pesan permintaan ini perlu dipahami pemrogram web sebab bisa jadi program yang kita buat berperan sebagai klien HTTP pengganti peramban. Jika demikian maka program kita harus bisa menyusun sendiri pesan permintaan dengan benar. Misalnya jika diperlukan untuk mengakses sebuah *web service* dari aplikasi kita sendiri.

Perintah Permintaan HTTP

Pesan permintaan HTTP mengandung perintah permintaan HTTP, disebut dengan *HTTP method* atau *HTTP verb*. Ada tujuh jenis pesan permintaan HTTP, yaitu:

1. GET yang berfungsi untuk mengambil (*read*) informasi dari server web.
2. POST yang berfungsi untuk membuat (*create*) informasi baru melalui server web.
3. PUT yang berfungsi untuk memperbarui (*update*) informasi melalui server web.
4. DELETE yang berfungsi untuk menghapus (*delete*) informasi melalui server web.
5. HEAD yang berfungsi seperti perintah GET namun hanya meminta bagian kepala (*header*) dari pesan balasan tanpa bagian badannya (berkas HTML tidak diperlukan).
6. TRACE yang berfungsi untuk menelusuri pergerakan informasi selama perjalanan dari klien web (peramban) ke server web.
7. OPTIONS yang berfungsi untuk mengetahui perintah HTTP apa saja yang bisa dilayani oleh server web yang dituju.

Dari 7 perintah di atas, yang paling umum digunakan adalah dua perintah pertama, yaitu GET dan POST. Perintah ketiga dan keempat, yaitu PUT dan DELETE, biasa digunakan untuk mengakses *web service*. 4 perintah HTTP ini menjadi dasar aksi CRUD (*create, read, update, delete*) dalam layanan web (*web service*) berbasis REST. Sedangkan perintah ke-5 hingga ke-7, biasanya untuk tujuan analisis atau pengujian sehingga jarang digunakan.

Pengiriman informasi melalui peramban hanya mendukung perintah GET dan POST. Itu sebabnya dua perintah tersebut yang paling banyak digunakan. Dan oleh sebab itu pula, kebanyakan aplikasi web menerapkan aksi memperbarui dan menghapus informasi juga menggunakan perintah POST (dengan keterangan) daripada perintah PUT dan DELETE.

Yang perlu diperhatikan, perintah HTTP ini sifatnya lebih sebagai informasi saja, bukan perintah mutlak yang harus dijalankan. Aksi yang sebenarnya akan dilakukan oleh server web atau program web bisa sama sekali berbeda dari perintah yang diterima. Walaupun tidak disarankan, namun bisa saja perintah yang diterima adalah menghapus (*delete*) tapi server web tidak menghapusnya, atau perintah membuat (*post*) tapi program web gunakan untuk menghapus informasi, dan sebagainya. Aksi yang dilakukan sepenuhnya berada di bawah kendali server web dan program web yang bekerja.

Menyusun Pesan Balasan

Setelah menerima pesan permintaan dari server web, seyogianya program web memberi pesan balasan (*response message*). Sebagaimana pesan permintaan, pesan balasan dari program web juga merupakan pesan HTTP, tetapi dengan isi yang berbeda. Dalam pesan balasan, kandungan pesan HTTP adalah sebagai berikut:

- Judul kepala berisi 3 informasi dalam susunan berikut:
HTTP/[VERSI] [KODE STATUS] [TEKS STATUS]
 - VERSI adalah versi protokol HTTP yang digunakan. Versi yang umumnya berlaku saat ini adalah versi 1.1 sehingga bagian ini biasanya berisi teks HTTP/1.1
 - KODE STATUS adalah tiga digit nomor kode status yang menunjukkan status pesan.
 - TEKS STATUS adalah teks keterangan kode status. Kode dan teks status ini telah ada aturan yang harus diikuti sehingga sisi klien juga mengerti maknanya.

Contoh: HTTP/1.1 404 Not Found

- Isi kepala berisi rincian balasan HTTP dengan susunan berikut:

[Nama-Informasi]: [Informasi]

Sebagaimana pesan permintaan, juga ada banyak rincian balasan HTTP, seperti Server, Content-Type, Content-Length, Set-Cookie, X-Powered-By, dan banyak lagi lainnya yang masing-masing mempunyai makna tersendiri. Contoh:

```
Content-Type: text/html;
Content-Length: 2316
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: FPC/3.0 ubuntu-x64
Set-Cookie: sessid=0888F207-1CB3-4C56-9E42-DFCF6B81B50E;
```

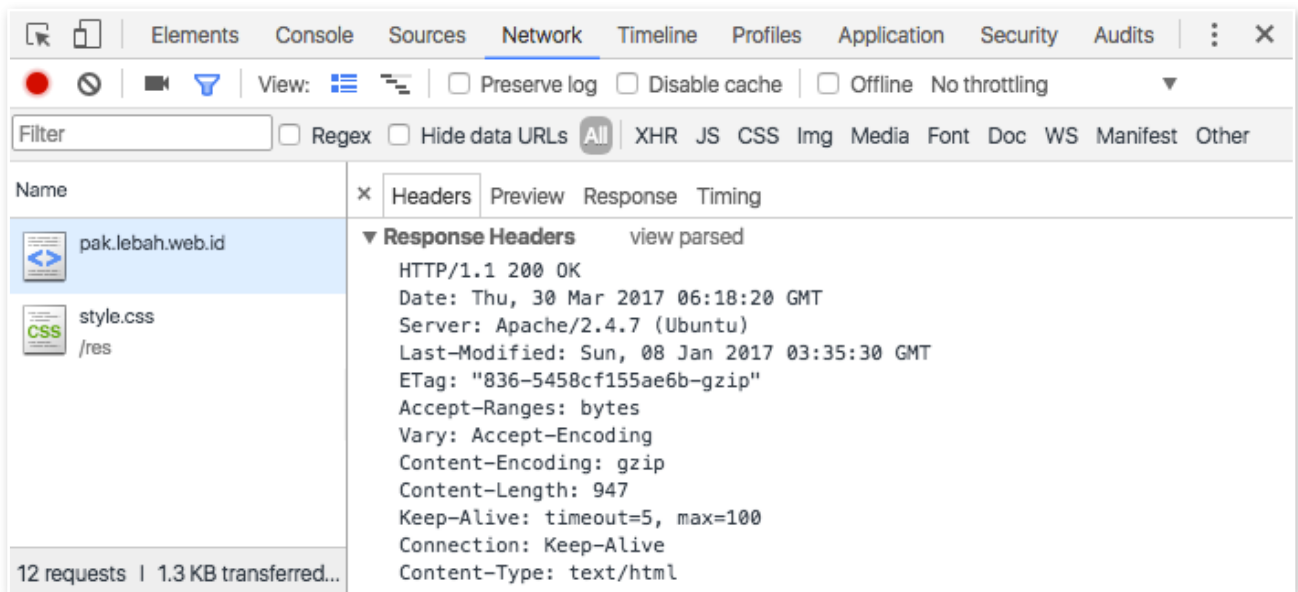
Ada satu rincian pesan balasan yang **wajib** atau harus selalu disertakan, yaitu rincian content-type karena rincian ini yang menentukan bagaimana klien web (peramban) memperlakukan dokumen di badan pesan.

Kecuali daftar rincian permintaan, daftar rincian umum dan entitas juga bisa digunakan dalam pesan balasan. Berikut adalah daftar rincian khusus untuk pesan balasan:

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.2>

- Badan pesan berisi data yang diminta oleh pesan permintaan. Di bagian inilah berkas HTML disertakan. Untuk layanan web (*web service*), bagian ini bisa diisi data dengan format lainnya seperti JSON atau XML.

Sebagaimana pesan permintaan, kita juga bisa melihat isi pesan balasan yang dikirim oleh server web ke peramban melalui alat bantu pengembang. Berikut adalah contoh tampilan pesan balasan di alat bantu pengembang di peramban Chrome.



Pesan balasan yang ditampilkan alat bantu pengembang di peramban Chrome.

Status Balasan HTTP

Jika mengakses suatu laman web melalui peramban maka peramban yang mengisi dan menyusun pesan permintaan. Program web yang berinteraksi dengan server web melalui interaksi CGI, status HTTP pada bagian judul kepala pesan balasan diisi oleh server web sebab kontrol interaksi ada di server web. Jika program web tak membalas atau gagal membalas terusan pesan permintaan dari server web maka server web yang memberikan pesan balasan ke peramban dengan kode status yang sesuai.

Ada lima kelompok status balasan HTTP, yaitu:

- 1xx : Status yang bersifat informasi, misalnya:
 - 101 (*switching protocol*) untuk pergantian atau pengalihan protokol.
- 2xx : Status sukses, misalnya:
 - 200 (*OK*) untuk pernyataan komunikasi HTTP sukses tanpa masalah.
- 3xx : Status pengalihan, misalnya:
 - 310 (*moved permanently*) untuk pemberitahuan laman telah pindah ke alamat lain.
- 4xx : Status kesalahan dari klien web, misalnya:
 - 404 (*not found*) untuk kesalahan tidak tersedianya laman atau berkas yang diminta.
- 5xx : Status kesalahan dari server atau program web, misalnya:
 - 500 (*internal server error*) untuk kegagalan program web dalam memberikan pesan balasan. Ini status kesalahan yang umum ditemui saat membuat program web.

Secara lebih lengkap, daftar status balasan HTTP bisa dipelajari di:

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10.1.2>

HTTP Bersifat Nirkeadaan

Protokol HTTP adalah hubungan yang bersifat nirkeadaan (*stateless protocol*) artinya setiap satu proses permintaan-balasan terlepas atau tidak tersangkut-paut baik dengan proses sebelumnya atau setelahnya. Setiap satu proses permintaan-balasan bersifat lepas, mandiri, dan tidak berkesinambungan dengan proses permintaan-balasan lainnya.

Kue HTTP

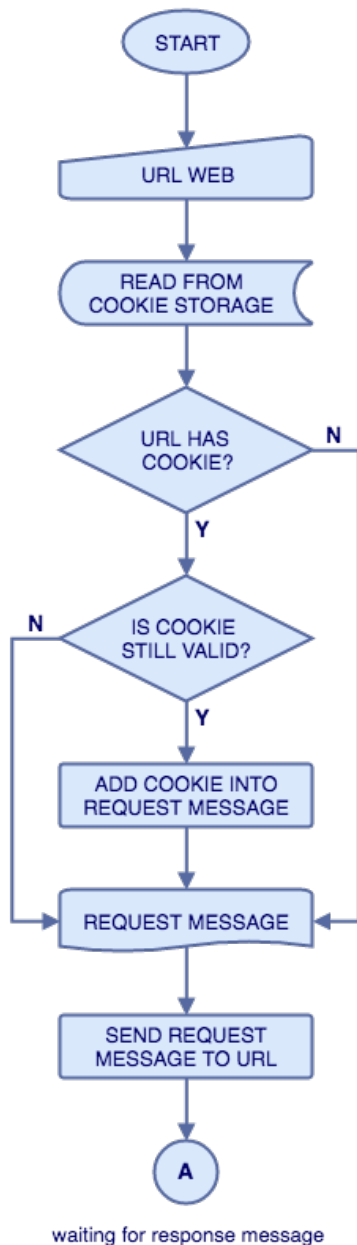
Dengan demikian, protokol HTTP tidak bisa mendukung komunikasi yang berurutan. Tapi HTTP mempunyai fitur yang disebut dengan kue HTTP (*HTTP cookie*). Kue HTTP adalah data yang diberikan oleh server web kepada klien web (peramban), yang mana data tersebut akan selalu disertakan di setiap pesan permintaan selanjutnya ke server web yang sama, hingga data tersebut tidak berlaku lagi (kedaluarsa). Fitur kue HTTP ini berada di sisi klien web sehingga memungkinkan sederet pesan permintaan membawa penanda (sementara) yang sama untuk masing-masing klien web.

Begini cara kerja kue HTTP di peramban:

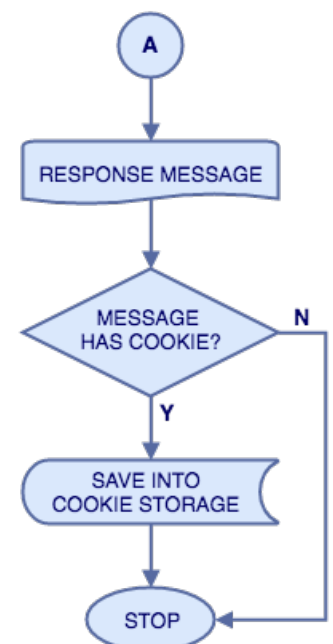
1. Saat peramban akan mengirimkan pesan permintaan ke sebuah alamat, peramban melihat apakah ada kue HTTP dari alamat tersebut dalam penyimpanan peramban.
2. Jika ada kue HTTP dari alamat yang akan dituju maka peramban mengecek apakah kue HTTP tersebut masih berlaku atau tidak.
3. Jika kue HTTP yang ada masih berlaku maka kue disertakan ke dalam kepala pesan permintaan.
4. Jika tidak ada kue HTTP atau kue HTTP yang ada sudah tidak berlaku maka tidak ada data penyertaan kue HTTP ke dalam pesan permintaan.
5. Kirim pesan permintaan ke alamat yang dituju.

Setelah dokumen HTML diterima, begini yang dilakukan peramban:

6. Saat peramban menerima pesan balasan, peramban mengecek apakah ada kue HTTP dalam pesan balasan.
7. Jika ada kue HTTP maka peramban menyimpannya.
8. Proses dokumen HTML yang diterima.
9. Saat ditutup, peramban menghapus kue HTTP yang tak berlaku.



receiving response message



Kue HTTP memiliki informasi masa berlaku. Masa berlaku kue HTTP ada dua jenis, yaitu:

1. Kue HTTP sementara.

Masa berlaku kue HTTP jenis ini mengikuti masa hidup peramban (klien web). Selama aplikasi peramban bekerja maka kue HTTP jenis ini tetap berlaku. Namun setelah aplikasi peramban ditutup maka kue HTTP jenis ini selesai masa berlakunya.

2. Kue HTTP tetap.

Masa berlaku kue HTTP jenis ditetapkan oleh server web dan disimpan oleh peramban (dalam berkas data yang terpisah) hingga masa berlakunya terlewati. Jika, misalnya, masa berlaku oleh server web diatur 1 tahun maka kue HTTP tersebut akan disimpan oleh peramban selama 1 tahun pula.

Sesi Web Mengatasi Nirkeadaan HTTP

Dengan adanya fitur kue HTTP, sebuah program web bisa mengatasi sifat nirkeadaan HTTP menjadi sadar-keadaan. Caranya dengan mencocokkan data kue HTTP yang dikirim klien web dengan data yang disimpan program web. Teknik inilah yang disebut dengan sesi web (*web session*). Dengan teknik ini maka setiap pesan permintaan bisa dikenali dan dikaitkan dengan pesan permintaan sebelumnya.

Perlu diingat bahwa kue HTTP disimpan peramban di sisi pengguna. Jika data kue HTTP ini dihapus oleh pengguna, baik disengaja atau tidak, maka pesan permintaan selanjutnya tak lagi membawa kue HTTP. Program web menjadi kehilangan jejak keterkaitan dengan pesan permintaan sebelumnya sehingga dianggap sebagai permintaan baru. Kesalahan ini umum disebut dengan sesi web yang rusak (*broken web session*).

Selain dengan kue HTTP, sesi web juga bisa dilakukan dengan cara lain. Misalnya dengan menambahkan peubah tersembunyi di setiap laman yang dibuat program web. Namun cara yang paling umum dan sering digunakan adalah dengan kue HTTP.

...

II. Menyiapkan VPS

Biasanya, pemrogram pemula yang akan belajar pemrograman web memulai dengan memasang web server di komputernya sendiri. Umumnya dengan memasang Apache (web server), MySQL (data server), dan PHP (*interpreter* bahasa PHP), atau disingkat dengan AMP, atau AMPP jika beserta Perl juga. Jika dipasang di sistem operasi Linux, disingkat LAMP (L = linux); jika dipasang di sistem operasi Windows, disingkat WAMP (W = windows); dan jika dipasang di sistem operasi Mac OS, disingkat MAMP (M = mac os). Karena terpasang di komputer sendiri, program web yang kita buat hanya bisa diakses di `localhost` (komputer lokal) sehingga hanya kita yang bisa menggunakannya. Program web yang tidak daring. Ini adalah cara lama dan kuno.

Cara baru dan modern adalah langsung belajar di VPS (*virtual private server*) daring. Sekarang banyak layanan yang menyediakan VPS secara gratis yang bisa kita gunakan untuk belajar pemrograman web. Program web yang kita buat langsung daring dan bisa diakses oleh semua orang di dunia, bahkan kita bisa belajar bersama-sama secara daring juga. Layaknya sedang menyunting dokumen di layanan Google Docs bersama-sama. Membuat program web juga bisa dilakukan cukup bermodal peramban seperti Chrome atau Firefox yang langsung terhubung ke VPS. Ini yang disebut dengan *programming on the cloud* atau pemrograman di awan.

A. Menggunakan layanan CodeAnywhere

Salah satu penyedia layanan VPS gratis yang terkenal adalah codeanywhere.com (kita singkat CA saja). Dalam kulgram ini, kita akan menggunakan layanan CA tersebut.

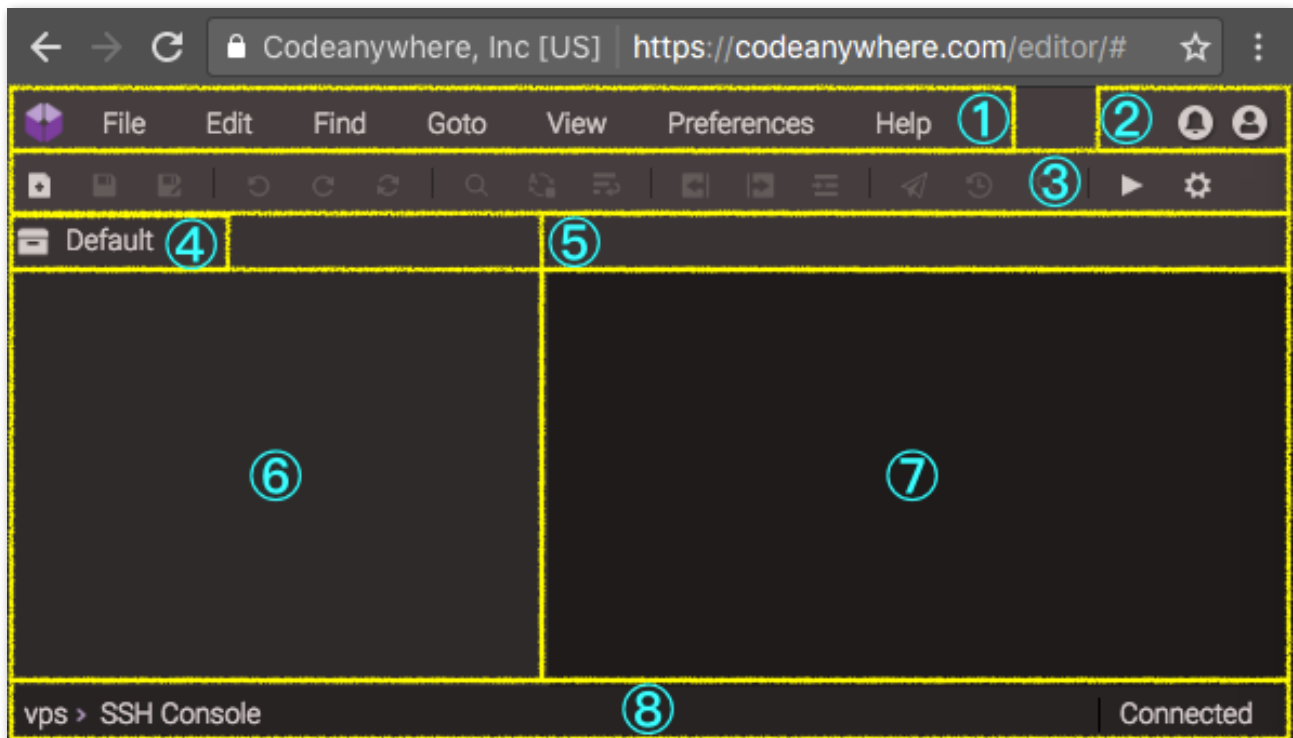
1. Mendaftar di layanan CodeAnywhere

Ikuti langkah-langkah berikut untuk mendaftar ke layanan CA:

1. Buka peramban dan ketik alamat berikut: codeanywhere.com/pricing (atau bisa juga ke codeanywhere.com lalu klik menu **Pricing** di sisi atas).
2. Pilih layanan gratis (*free* atau \$0) di sisi paling kiri, lalu klik **Sign Up** (mendaftar) untuk membuat akun CA baru.
3. Di laman pendaftaran yang muncul, masukkan alamat surel (*email*) dan kata kunci (*password*) Anda, kemudian klik **Register** (daftarkan).
4. Buka akun surel yang Anda gunakan, tunggu datangnya surel konfirmasi akun dari layanan CA. Akun CA belum bisa digunakan sebelum akun tersebut dikonfirmasi.
5. Setelah surel konfirmasi tiba —biasanya tak lebih dari 5 menit— ikuti tautan yang disertakan dalam surel untuk proses pengesahan akun baru CA Anda tadi.

6. Buka kembali CA di peramban dan masuk (*login*) menggunakan akun CA yang telah disahkan tadi.

Jika semua langkah di atas dilakukan dengan benar dan sukses, seharusnya kita sekarang sedang berhadapan dengan IDE (*integrated development environment*) daring dari CA.



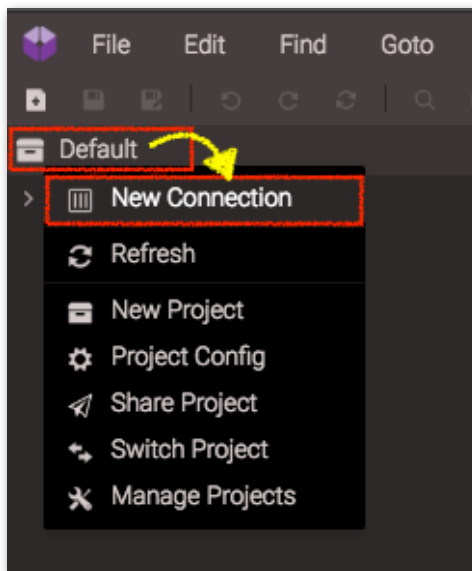
Tampilan IDE CodeAnywhere dan bagian-bagiannya.

Bagian-bagian IDE CA:

1. Menu utama, berisi daftar menu untuk akses ke berbagai fitur IDE.
2. Pengaturan akun dan pemberitahuan (*notification*), tempat untuk mengelola informasi akun dan proyek serta pemberitahuan dari layanan CA.
3. Baris perangkat (*toolbar*), berisi daftar tombol pintasan untuk berbagai fitur IDE.
4. Baris proyek, berisi nama proyek dan akses ke menu fitur proyek.
5. Baris *tab* berkas, berisi deretan *tab* berkas yang dibuka atau terminal Linux.
6. Pohon berkas (*file tree*), berisi susunan pohon berkas di *folder* yang diacu oleh IDE.
7. Ruang kerja (*workspace*), adalah tempat kerja penyunting dan terminal Linux.
8. Baris keterangan (*status bar*), berisi berbagai keterangan terkait hal yang terkait, misal berkas yang sedang aktif dikerjakan (sisi kiri) dan status koneksi VPS (sisi kanan).

2. Membuat VPS di CodeAnywhere

Sekarang kita telah memiliki akun di CA, namun kita belum mempunyai VPS. CA bisa menyediakan banyak VPS dalam satu akun, dimana masing-masing VPS disebut dengan istilah wadah (*container*). Akun gratis CA hanya bisa membuat satu wadah saja dengan kemampuan perangkat keras yang cukup terbatas. Namun buat belajar pemrograman web di awan, satu wadah tersebut sudah lebih dari cukup.

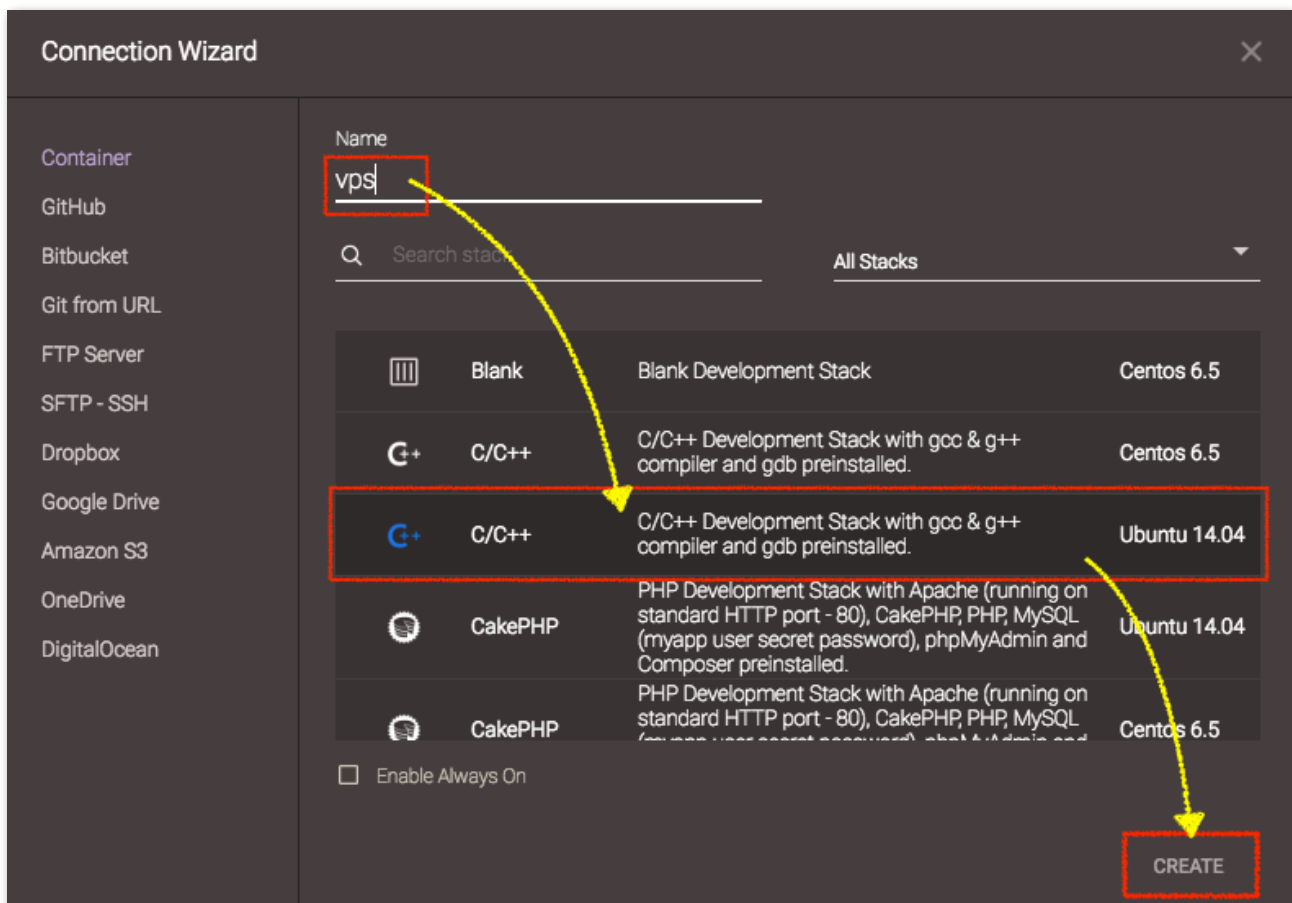


Berikut langkah-langkah membuat VPS di akun CA (dan perhatikan kotak merah di gambar):

1. Klik kanan pada ikon proyek **Default** (sisi kiri atas, di bawah menu utama IDE), akan memunculkan daftar menu proyek.
2. Klik pilihan menu **New Connection** (pilihan pertama pada menu proyek), akan memunculkan jendela **Container Wizard**.
3. Pada jendela **Container Wizard**, cari dan klik pilihan wadah **C/C++** dengan sistem operasi Linux **Ubuntu 14.04** seperti pada gambar di bawah ini.

← Menu proyek.

4. Agar memudahkan, beri nama wadah sesingkat mungkin. Dan karena di akun gratis hanya boleh punya satu wadah saja, wadah kita beri nama “vps” (tanpa petik). Ketikkan **vps** pada kolom masukan **Name**, seperti pada gambar berikut.



Pilihan dan penamaan wadah di jendela **Connection Wizard**.

5. Setelah dipastikan pilihan dan isian nama VPS benar, klik tombol **Create** (kanan bawah jendela). Setelah itu CA akan membuat VPS yang kita pesan. Apa yang sedang terjadi selama proses pembuatan dimunculkan dalam jendela kecil. Proses pembuatan VPS ini butuh waktu sekitar 1 hingga 3 menit.

6. Jika proses pembuatan VPS berhasil, IDE akan menampilkan informasi VPS di sebuah *tab* dengan nama sesuai dengan nama VPS, dalam hal ini adalah **vps**. seperti yang ditunjukkan pada gambar di bawah.

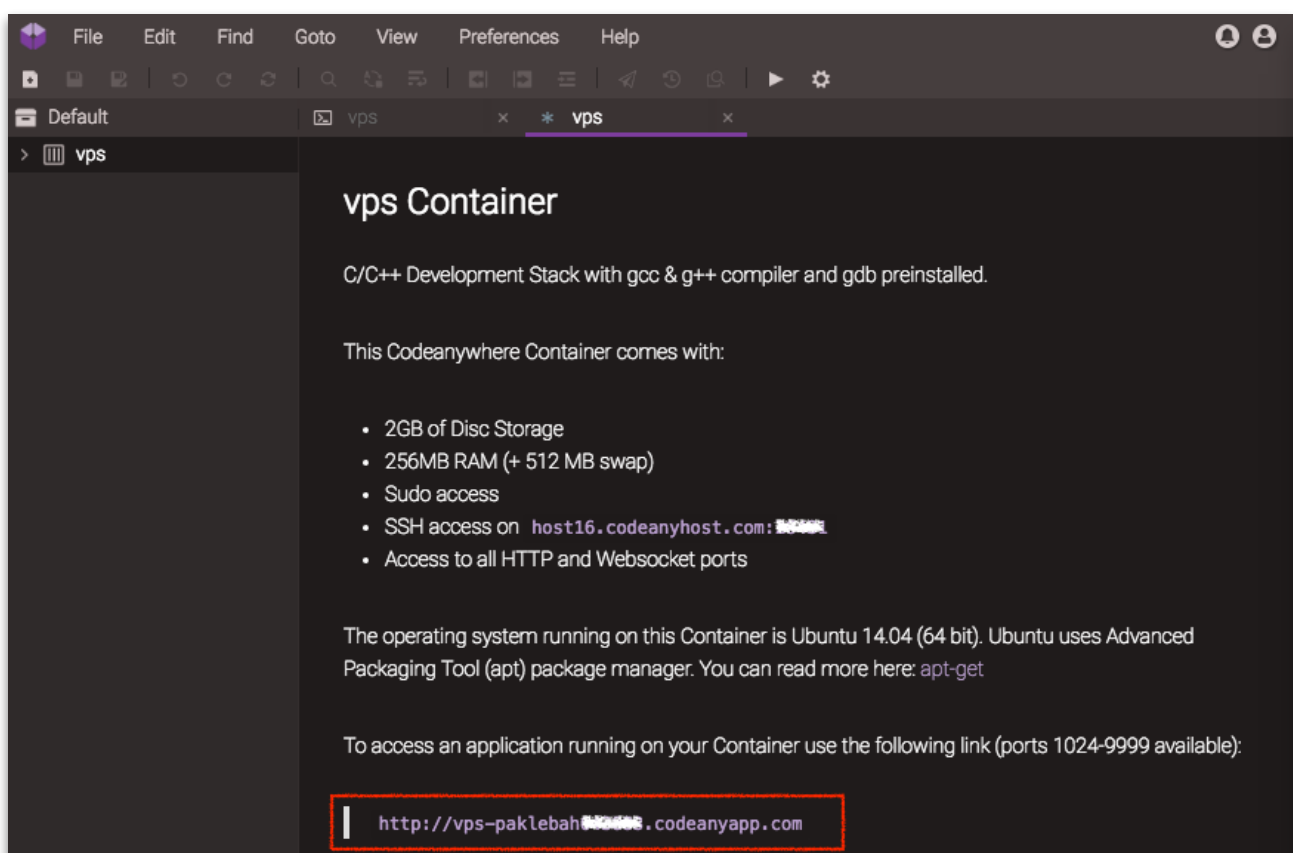
Dari informasi yang ditampilkan, ada satu informasi penting yaitu alamat web VPS kita yang bisa diakses dari internet. Alamat web VPS adalah sebagai berikut:

[http://\[nama_vps\]-\[nama_akun\]\[nomor_akun\].codeanyapp.com](http://[nama_vps]-[nama_akun][nomor_akun].codeanyapp.com)

Misalnya, pada akun saya adalah:

<http://vps-paklebahXXXXXX.codeanyapp.com>

dengan XXXXXX adalah nomor akun yang berbeda untuk setiap akun. Selain itu, ada juga informasi cara mengakses VPS melalui SSH, bagi yang ingin akses VPS langsung.



Tampilan informasi VPS yang telah dibuat.

Kini VPS telah aktif, namun masih kosong. Oleh karena itu, jika pada tahap ini kita coba akses ke alamat web VPS kita, akan muncul pesan kesalahan “*This Container is currently inaccessible*” (wadah ini sedang tak bisa diakses) dari layanan CA karena walaupun VPS telah aktif tapi VPS masih kosong, dalam artian belum ada aplikasi atau laman web yang tersedia. Masih ada beberapa hal yang perlu disiapkan lagi sebelum kita memulai belajar pemrograman web.

3. Kekurangan dan kelebihan

Layanan yang diberikan CA sangat bermanfaat. Yang gratis juga tetap memberikan banyak fasilitas penting. Fasilitas yang dulu harus dibayar cukup mahal, kini bisa kita manfaatkan

tanpa biaya, atau dengan biaya yang cukup ringan (terjangkau). Tapi tentu saja akun gratis juga memiliki batasan-batasan. Beberapa batasan akun gratis, antara lain:

1. Hanya bisa memiliki 1 (satu) wadah VPS yang kemampuannya cukup terbatas. Berikut kemampuan perangkat keras di akun gratis:
 - *Processor* berinti tunggal dengan kecepatan 1,1 GHz 64-bit,
 - Pengingat RAM dengan kapasitas 256 MB, dan
 - Penyimpan (bukan SSD) berkapasitas 2 GB.
2. Waktu aktif VPS gratis dibatasi. VPS hanya menyala selama kita (pengguna) melakukan aktivitas di komputer yang mengakses IDE. Jika pengguna melakukan:
 - keluar (*logout*) dari IDE, atau
 - menutup *tab* atau jendela peramban, atau
 - meninggalkan komputer dalam diam (*idle*) lebih dari 60 menit,maka secara otomatis VPS akan dipadamkan. Intinya, selama kita masih sibuk menulis program di IDE CA, selama itu pula VPS kita menyala dan bisa diakses.
3. Serta batasan lain sebagaimana disebutkan di situs layanan CodeAnywhere [ini](#). Namun dua batasan di atas adalah batasan yang paling terasa membatasi.

Walaupun ada batasan, tapi fasilitas di akun gratis juga tidak bisa diremehkan. Misalnya:

1. Kendali penuh atas sistem operasi (Linux) di wadah VPS, melalui akses `sudo` (hak akses setara *root*) dan terminal. Ini juga bisa kita manfaatkan untuk belajar Linux.
2. Akses ke VPS melalui SSH (+SFTP). Bagi yang ingin menggunakan IDE sendiri, seperti Lazarus IDE misalnya, bisa memanfaatkan jalur SSH.
3. Fasilitas berbagi (*share*) akses ke sesama pengguna CA. Dengan fasilitas ini, kita bisa membuat program bersama (*pair programming*) di VPS.

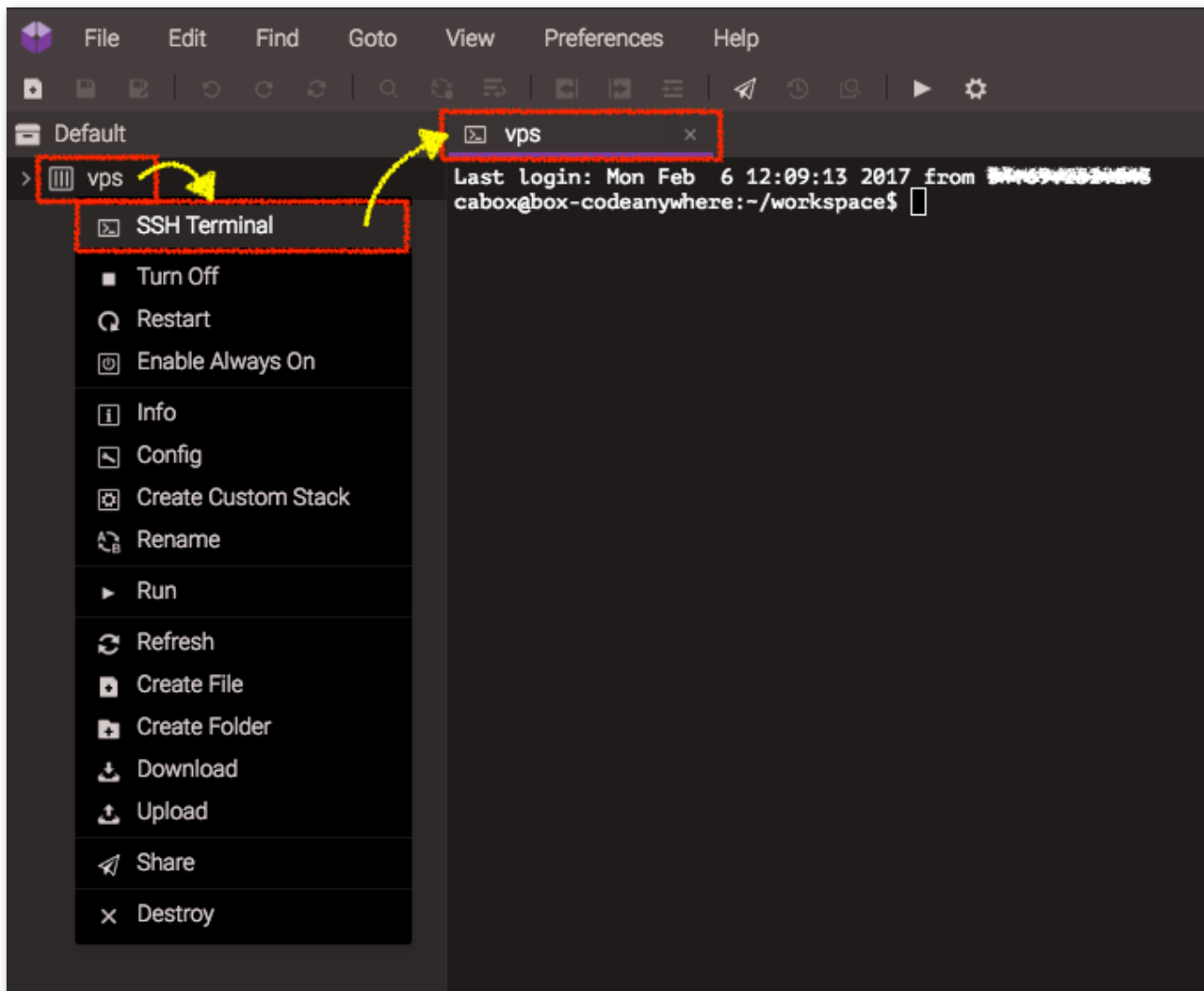
B. Memasang server web dan Pascal

VPS Linux dari CA telah siap digunakan. Selanjutnya adalah memasang server web dan Pascal di VPS. Untuk server web, kita gunakan yang umum dan mudah digunakan, yaitu Apache. Dan untuk Pascal-nya, tentu saja kita gunakan Free Pascal. Sementara ini Delphi belum mendukung Linux, pun jika mendukung nanti harganya mahal dengan kemampuan yang tak lebih baik dari Free Pascal yang gratis. Jadi, lupakan dulu Delphi.

1. Pengaturan awal VPS

Sebelum memasang server web dan Pascal, ada beberapa hal yang perlu diatur pada VPS untuk kemudahan selanjutnya. Ikuti saja langkah-langkah berikut:

1. Pada IDE, klik kanan pada ikon wadah kita yang bernama **vps** (sisi kiri atas, di bawah ikon proyek), akan memunculkan daftar menu wadah.
2. Klik **SSH Terminal** (pilihan pertama pada menu wadah), akan memunculkan terminal akses ke Linux, seperti pada gambar di bawah ini.



Tampilan menu wadah dan terminal akses Linux.

3. Klik di *tab* terminal kemudian jalankan perintah berikut secara berurutan:

```
$ cd ~  
$ sudo apt-get update  
$ sudo apt-get install nano  
$ sudo apt-get install lynx
```

Perintah-perintah di atas untuk memasang (*install*) aplikasi tambahan yang nanti kita perlukan. Jika saat proses pemasangan muncul pertanyaan konfirmasi, jawab *Yes* saja.

4. Setelah proses di atas selesai, selanjutnya adalah menyunting program awalan (*startup script*) dengan menjalankan perintah berikut:

```
$ nano .bashrc
```

Perintah di atas akan membuka dan menampilkan kode program awalan. Selanjutnya, ubah kode program di **baris ke-4**, dari yang semula berisi:

```
cd ~/workspace
```

ubah menjadi:

```
cd ~
```

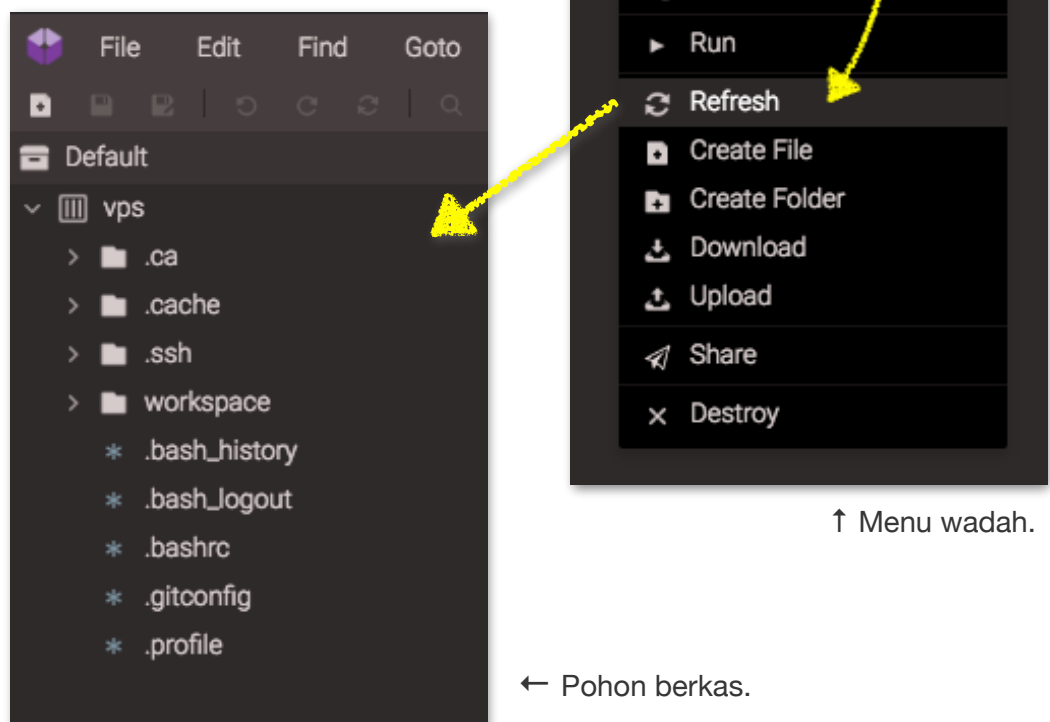
yang bertujuan untuk mengubah tujuan pohon berkas ke *folder* rumah (*home*).

Selanjutnya geser kursor ke bagian paling akhir/bawah, lalu tambahkan baris berikut:

```
alias pas=". ~/pas.sh"
```

yang bertujuan untuk memasang alias pada perintah pemanggilan Free Pascal melalui *shell script* yang akan kita bahas nanti.

5. Kemudian, lanjutkan dengan menekan tombol **Ctrl+X** lalu tombol **Y** lalu **Enter**. Berkas disimpan, penyunting ditutup, dan kita kembali ke terminal Linux.
6. Tutup *tab* terminal dengan klik tanda silang di sisi kanan atas *tab*.
7. Kemudian jalankan perintah **Restart** VPS melalui menu wadah. Perintah *restart* ini bekerja seperti *restart* komputer. Tunggu hingga proses *restart* selesai, butuh waktu sekitar satu menit.
8. Jika pohon daftar berkas (di sisi kiri IDE) tidak berubah setelah *restart*, jalankan perintah **Refresh** dari menu wadah. Perintah *refresh* ini untuk menyegarkan tampilan isi daftar berkas.
9. Setelah proses *restart* dan *refresh* selesai, seharusnya muncul daftar berkas seperti pada gambar di bawah ini.



Pengaturan awal VPS telah selesai. Kini pohon berkas telah mengarah ke *home folder*. Perlu diketahui, di layanan CA, kita otomatis terdaftar di Linux sebagai pengguna dengan nama **cabox** grup **cabox** dengan hak-akses **sudo** (akses sebagai *administrator* atau **root**). Sementara server web bekerja atas nama pengguna **www-data** grup **www-data**.

2. Pemasangan server web

Memasang server web Apache di Ubuntu sangat mudah. Ikuti langkah-langkah berikut:

1. Jika *tab* terminal belum terbuka, buka terminal dari menu wadah, seperti yang telah ditunjukkan sebelumnya.

2. Ketikkan perintah berikut:

```
$ sudo apt-get install apache2
```

yang akan memasang server web Apache. Jika muncul pertanyaan konfirmasi saat proses pemasangan, jawab *Yes* saja. Tunggu hingga proses pemasangan selesai.

3. Selanjutnya, aktifkan modul Apache yang kita butuhkan. Ketikkan perintah berikut:

```
$ sudo a2enmod cgid
```

```
$ sudo a2enmod rewrite
```

yang akan mengaktifkan modul CGI dan *rewrite*.

4. Kemudian, aktifkan pengaturan CGI dengan perintah berikut:

```
$ sudo nano /etc/apache2/sites-available/000-default.conf
```

yang akan membuka berkas pengaturan Apache. Gerakkan kursor hingga **baris ke-32** (menjelang akhir berkas), yaitu pada baris berisi teks ini:

```
#Include conf-available/serve-cgi-bin.conf
```

lalu hapus **#** (tanda pagar) di awal/kiri baris tersebut. Penghapusan ini bertujuan untuk mengaktifkan fitur CGI yang secara bawaan dinon-aktifkan.

5. Lanjutkan menyimpan berkas, dengan menekan tombol **Ctrl+X** lalu tombol **Y** lalu **Enter**. Berkas disimpan, penyunting ditutup, dan kita kembali ke terminal Linux.

6. Karena mengaktifkan modul dan mengubah pengaturan, kita perlu melakukan *restart* server web agar modul dan pengaturan baru diterapkan. Ketikkan perintah berikut:

```
$ sudo apache2ctl restart
```

7. Pemasangan server web Apache selesai. Untuk mengetahui status server web, gunakan perintah berikut:

```
$ sudo apache2ctl status
```

yang akan menampilkan informasi status server web.

Setelah server web terpasang baik, mari kita lihat apakah VPS kita sudah bisa diakses dari internet. Lakukan hal berikut:

1. Buka *tab* peramban baru, lalu bukalah alamat web VPS Anda, seperti yang ditunjukkan pada laman info VPS.
2. Jika *tab* info belum terbuka (atau sudah ditutup), bukalah melalui menu wadah, lalu pilih menu **Info**, lalu klik tautan alamat web VPS Anda.
3. Jika server web berhasil terpasang dan aktif, maka akan tampil laman bawaan server web Apache dengan tulisan “*Apache2 Ubuntu Default Page*” di bagian judul.

3. Pemasangan Free Pascal

Karena gudang (*repository*) Ubuntu masih lambat memperbarui paketnya, pemasangan Free Pascal (selanjut kita singkat menjadi FPC saja) tak semudah memasang server web di atas, namun juga tidak sulit. Ikuti saja langkah-langkah berikut ini:

1. Jika *tab* terminal belum terbuka, buka terminal dari menu wadah.

2. Ketikkan perintah berikut:

```
$ wget ftp://freepascal.stack.nl/pub/fpc/dist/3.0.0/x86_64-linux/fpc-3.0.0.x86_64-linux.tar
```

yang akan mengunduh berkas FPC versi 3 untuk Linux, langsung dari situs resmi FPC. Tunggu hingga proses pengunduhan selesai, butuh waktu sekitar 3 hingga 5 menit saja.

3. Setelah pengunduhan selesai, ketikkan perintah berikut:

```
$ tar xf fpc-3.0.0.x86_64-linux.tar
```

yang akan membongkar berkas hasil unduhan, kemudian lanjutkan dengan:

```
$ cd fpc-3.0.0.x86_64-linux
```

yang akan memasuki *folder* pemasangan FPC.

4. Lalu ketikkan perintah berikut:

```
$ sudo ./install.sh
```

yang akan menjalankan aplikasi pemasang (*installer*) FPC. Saat aplikasi bekerja, untuk mudahnya ikuti saja jawaban yang telah disediakan, dengan menekan **Enter**.

5. Kemudian berturut-turut ketikkan perintah berikut:

```
$ cd ..
```

```
$ rm -rf fpc-3.0.0.x86_64-linux
```

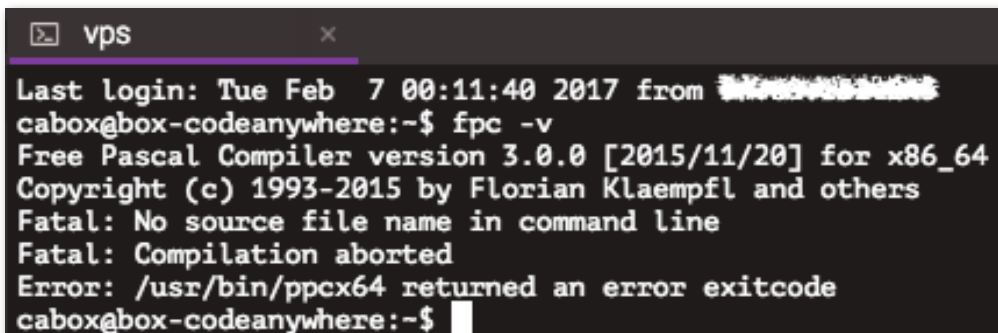
```
$ rm fpc-3.0.0.x86_64-linux.tar
```

yang akan menghapus unduhan dan aplikasi pemasang FPC yang sudah tidak kita butuhkan lagi, lagi-lagi untuk menghemat ruang penyimpanan VPS.

6. Untuk menguji apakah FPC telah sukses terpasang, ketikkan perintah berikut:

```
$ fpc -v
```

untuk menampilkan versi FPC, seperti pada gambar di samping ini.



```
vps
Last login: Tue Feb  7 00:11:40 2017 from [REDACTED]
cabox@box-codeanywhere:~$ fpc -v
Free Pascal Compiler version 3.0.0 [2015/11/20] for x86_64
Copyright (c) 1993-2015 by Florian Klaempfl and others
Fatal: No source file name in command line
Fatal: Compilation aborted
Error: /usr/bin/ppcx64 returned an error exitcode
cabox@box-codeanywhere:~$
```

VPS sudah aktif. Server web Apache sudah terpasang. Free Pascal Compiler juga telah terpasang. Sekarang waktunya kita memulai belajar pemrograman web yang sebenarnya dengan bahasa Pascal.

III. Pascal dan Web

Seperti yang telah dijelaskan di awal, program web pada dasarnya hanya saling bertukar masukan dan keluaran dengan server web. Sehingga membuat program web dengan bahasa Pascal —atau bahasa apa pun— tidaklah sulit, sepanjang bisa mengakses masukan dan keluaran, yang pasti bisa dilakukan semua bahasa pemrograman. Di Pascal, cukup bermodal prosedur klasik `read/ln` dan `write/ln` saja, sudah bisa. Jadi, anggapan bahwa membuat program web dengan Pascal itu sulit adalah bohong alias *hoax*.

A. Hello World!

Tradisi belajar pemrograman yang terkenal adalah membuat program *Hello World* sebagai program pertama. Mari kita coba. Untuk itu, mari kita kembali ke IDE CA kita, dan lakukan langkah-langkah berikut ini:

1. Buat berkas melalui menu wadah, pilih **Create File**, dan tuliskan `test1.pas` pada jendela yang muncul, lalu klik tombol **OK**.
2. IDE akan membuat berkas baru dan langsung membukanya untuk disunting. Silakan ketikkan kode program berikut ke dalam berkas `test1.pas` tersebut.

```
program webTest;

begin
  writeln('content-type: text/html;');
  writeln; // penting!
  writeln('<!DOCTYPE html>');
  writeln('<html><head>');
  writeln('<meta charset="utf-8">');
  writeln('<meta name="viewport" content="width=device-width,initial-scale=1">');
  writeln('<title>Hello World</title>');
  writeln('</head><body>');
  writeln('<h1>Hello World!</h1>');
  writeln('This page is created using Free Pascal v3 and hosted on CodeAnywhere.');
```

```
  writeln('</body></html>');
end.
```

3. Setelah selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau dengan pintasan **Ctrl+S**. Tanda sebuah berkas belum disimpan adalah adanya tanda bulat berwarna putih pada *tab* berkas di tempat tanda silang berada.
4. Jika *tab* terminal belum terbuka, buka terminal dari menu wadah.

5. Di terminal, ketikkan perintah berikut:

```
$ fpc test1.pas
```

yang akan mengkompilasi berkas `test1.pas` dengan FPC. Jika proses kompilasi berhasil, lanjutkan dengan:

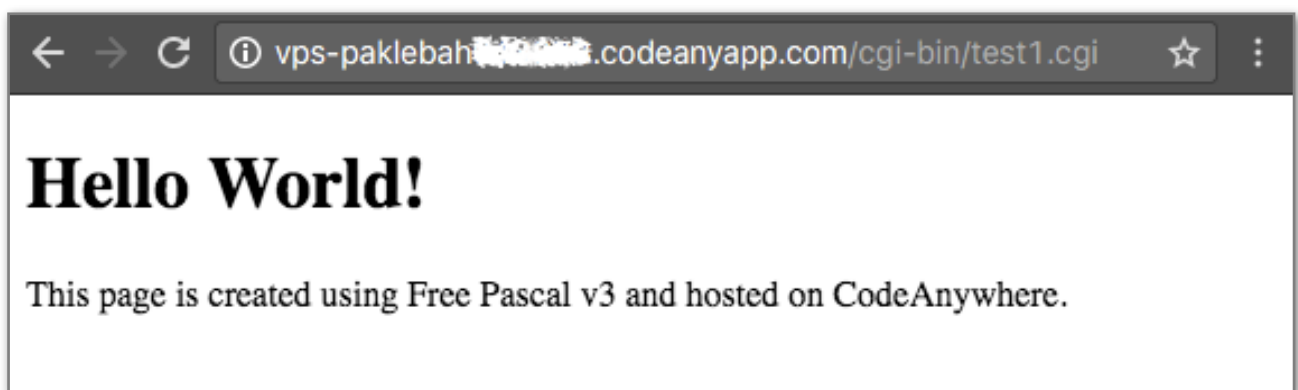
```
$ sudo mv test1 /usr/lib/cgi-bin/test1.cgi
```

yang akan memindahkan program (*executable*) hasil kompilasi ke *folder* `cgi-bin` agar program bisa diakses dari internet.

6. Untuk melihat aksi program, buka *tab* peramban baru (atau gunakan *tab* yang sudah ada) lalu ketikkan alamat web VPS Anda diikuti dengan `/cgi-bin/test1.cgi`
Contoh, pada akun saya adalah:

```
http://vps-paklebahXXXXXX.codeanyapp.com/cgi-bin/test1.cgi
```

7. Jika tak ada kesalahan, seharusnya muncul laman seperti pada gambar di bawah ini.



Tampilan Hello World dari program `test1.pas`.

Bagaimana, cukup mudah bukan membuat aplikasi web dengan Pascal? Hanya dalam waktu sekitar 30 menit saja, kita telah mengaktifkan server web yang benar-benar *online* yang bisa diakses semua orang, membuat program web sederhana dengan Free Pascal secara *programming on the cloud* yang hanya bermodal *browser*, dan tanpa sepeser pun biaya yang kita keluarkan.

B. Pengaturan yang memudahkan

Walaupun langkah-langkah di atas cukup mudah dilakukan, tapi masih cukup panjang. Dengan pengaturan tambahan kita bisa membuat proses di atas menjadi lebih mudah dan pendek lagi. Berikut caranya...

1. Pengaturan server web

Pengaturan bawaan server web Apache mengharuskan kita untuk meletakkan berkas-berkas aplikasi web (seperti berkas program CGI, html, javascript, css, gambar, dan lain sebagainya) di *folder* khusus yang membutuhkan akses khusus pula. Ini tujuannya untuk keamanan server dan aplikasi agar tak bisa dijangkau orang yang tak berhak, baik dari dalam maupun dari luar sistem.

Tapi karena kita sedang belajar, hal seperti itu cukup merepotkan. Karena itu mari kita ubah pengaturan server web agar kita lebih nyaman buat belajar. Caranya tak sulit, cukup ikuti saja langkah-langkah berikut:

1. Jika *tab* terminal belum terbuka, buka terminal dari menu wadah, seperti yang telah ditunjukkan sebelumnya.

2. Kemudian ketikkan perintah berikut:

```
$ mv workspace/ web/
```

yang akan mengubah nama *folder* `workspace` menjadi `web`.

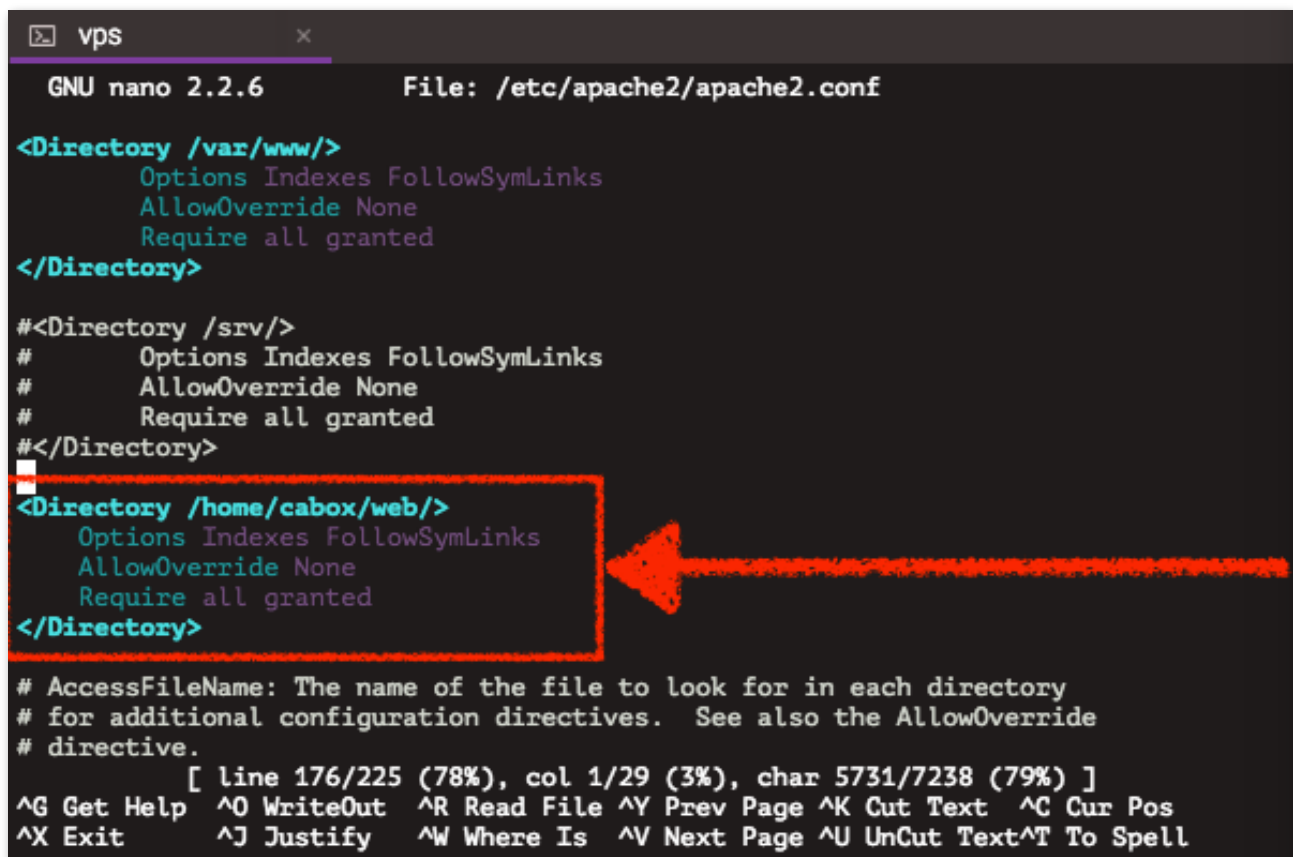
3. Kemudian ketikkan perintah berikut:

```
$ sudo nano /etc/apache2/apache2.conf
```

yang akan membuka berkas utama pengaturan server web Apache. Geser kursor ke bawah mendekati akhir berkas. Pada **baris ke-176**, tambahkan teks berikut ini:

```
<Directory /home/cabox/web/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

sehingga tampilan di sekitar baris tersebut menjadi seperti gambar di bawah ini:



```
vps
GNU nano 2.2.6      File: /etc/apache2/apache2.conf

<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

#<Directory /srv/>
#     Options Indexes FollowSymLinks
#     AllowOverride None
#     Require all granted
#</Directory>

<Directory /home/cabox/web/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

# AccessFileName: The name of the file to look for in each directory
# for additional configuration directives.  See also the AllowOverride
# directive.
[ line 176/225 (78%), col 1/29 (3%), char 5731/7238 (79%) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Tampilan berkas `apache2.conf` setelah ditambah pengaturan baru.

4. Lanjutkan menyimpan berkas, dengan menekan tombol **Ctrl+X** lalu tombol **Y** lalu **Enter**. Berkas disimpan, penyunting ditutup, dan kita kembali ke terminal Linux.

5. Pengaturan ini membuat *folder web* (dalam *folder rumah*) juga bisa diakses oleh server web sehingga berkas dan program web bisa diletakkan di tempat yang bisa kita akses.
6. Selanjutnya, ketikkan perintah berikut:

```
$ sudo nano /etc/apache2/sites-available/000-default.conf
```

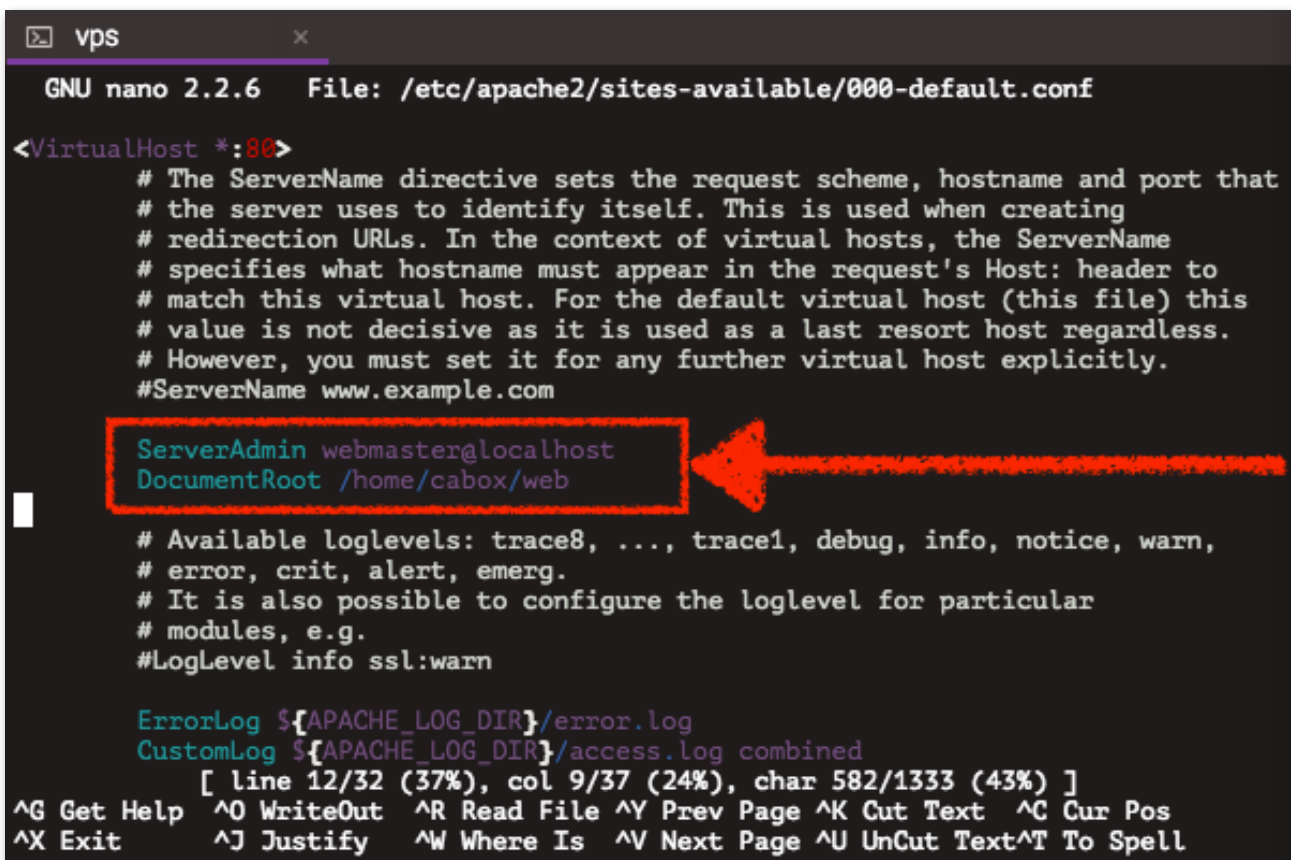
yang akan membuka berkas lain pengaturan server web Apache. Geser kursor ke bawah hingga **baris ke-12**, lalu ubah teks dari yang semula:

```
/var/www/html
```

ubah menjadi:

```
/home/cabox/web
```

seperti yang ditunjukkan dalam gambar di bawah ini:



```
GNU nano 2.2.6 File: /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /home/cabox/web

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
[ line 12/32 (37%), col 9/37 (24%), char 582/1333 (43%) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Tampilan berkas 000-default.conf setelah diubah.

7. Lanjutkan menyimpan berkas, dengan menekan tombol **Ctrl+X** lalu tombol **Y** lalu **Enter**. Berkas disimpan, penyunting ditutup, dan kita kembali ke terminal Linux.
8. Pengaturan ini membuat *folder web* (dalam *folder rumah*) menjadi tempat utama berkas aplikasi web yang diacu oleh server web Apache. Apa pun berkas yang ada di *folder* tersebut akan bisa diakses dari internet.
9. Selanjutnya, ketikkan perintah berikut:

```
$ sudo nano /etc/apache2/conf-available/serve-cgi-bin.conf
```

yang akan membuka berkas lain pengaturan server web Apache. Geser kursor ke bawah mendekati akhir berkas. Pada **baris ke-20**, tambahkan teks berikut ini:

```

<Directory /home/cabox/web/>
    AddHandler cgi-script .cgi
    AllowOverride None
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    Require all granted
</Directory>

```

seperti yang ditunjukkan dalam gambar di bawah ini:

```

GNU nano 2.2.6 File: /etc/apache2/conf-available/serve-cgi-bin.conf

<IfModule mod_cgid.c>
    Define ENABLE_USR_LIB_CGI_BIN
</IfModule>

<IfDefine ENABLE_USR_LIB_CGI_BIN>
    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Require all granted
    </Directory>
</IfDefine>
</IfModule>
<Directory /home/cabox/web/>
    AddHandler cgi-script .cgi
    AllowOverride None
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    Require all granted
</Directory>

[ line 20/28 (71%), col 1/29 (3%), char 410/631 (64%) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell

```

Tampilan berkas `serve-cgi-bin.conf` setelah ditambah pengaturan baru.

10. Lanjutkan menyimpan berkas, dengan menekan tombol **Ctrl+X** lalu tombol **Y** lalu **Enter**. Berkas disimpan, penyunting ditutup, dan kita kembali ke terminal Linux.
11. Pengaturan ini membuat *folder web* (dalam *folder rumah*) juga bisa untuk meletakkan program CGI sehingga program CGI yang kita buat tak perlu disalin ke *folder cgi-bin*.
12. Setelah pengaturan baru disimpan, langkah terakhir adalah melakukan *restart* server web Apache agar pengaturan baru di atas diterapkan. Jalankan perintah berikut:


```
$ sudo apache2ctl restart
```

Menguji pengaturan baru

Selanjutnya, kita perlu memastikan apakah pengaturan baru di atas telah diterapkan dan bekerja dengan benar. Untuk itu, lakukan langkah-langkah berikut:

1. Di pohon berkas, klik kanan pada ikon *folder web*, lalu pilih **Create File**, dan tuliskan `index.html` pada jendela yang muncul, lalu klik tombol **OK**.

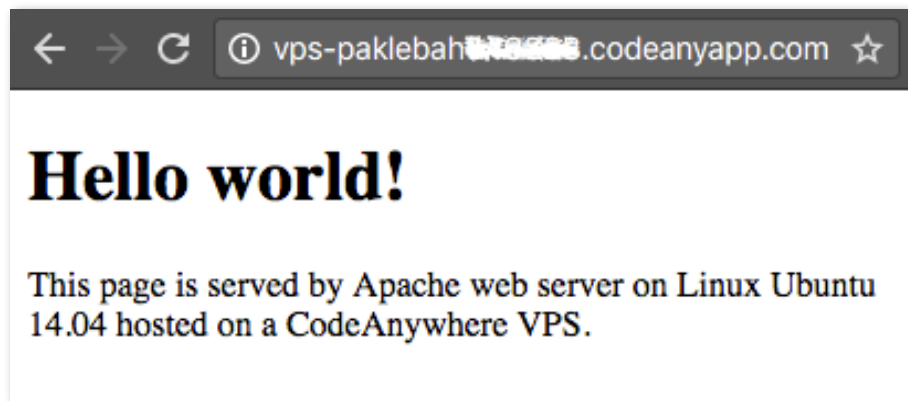
2. IDE akan membuat berkas baru dan langsung membukanya untuk disunting. Silakan ketikkan kode program berikut ke dalam berkas `index.html` tersebut.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>My VPS @ CodeAnywhere</title>
  </head>
  <body>
    <h1>Hello world!</h1>
    <p>This page is served by Apache web server on
    Linux Ubuntu 14.04 hosted on a CodeAnywhere VPS.</p>
  </body>
</html>
```

3. Setelah selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau pintasan **Ctrl+S**. Perhatikan pohon berkas, pastikan `index.html` ada dalam *folder web*, bukan di *folder* rumah.
4. Buka *tab* peramban baru (atau gunakan *tab* lain yang sudah ada) lalu ketikkan alamat web VPS Anda. Contoh, pada akun saya adalah:

`http://vps-paklebahXXXXXX.codeanyapp.com/`

Jika pengaturan baru berhasil, seharusnya Anda akan melihat laman web dengan judul “Hello world!” sebagaimana gambar di samping ini.



5. Kemudian kita menguji program CGI kita. Mari kembali lagi ke *tab* terminal. Jika *tab* terminal tadi sudah ditutup, buka *tab* terminal baru dari menu wadah.
6. Di terminal, ketikkan perintah berikut:

```
$ fpc test1.pas -otest1.cgi
```

yang akan mengkompilasi kode program `test1.pas` tadi menjadi program `test1.cgi` di *folder* saat ini (rumah), dilanjutkan dengan:

```
$ mv test1.cgi web/
```

yang akan memindahkan program `test1.cgi` ke *folder web*.

7. Buka *tab* peramban baru (atau gunakan *tab* lain yang sudah ada) lalu ketikkan alamat web VPS Anda. Contoh, pada akun saya adalah:

`http://vps-paklebahXXXXXX.codeanyapp.com/test1.cgi`

↑ Tampilan laman utama yang membuka berkas `index.html`

Perhatikan bahwa kita tak perlu mengarahkan CGI ke *folder* `/cgi-bin/` lagi, seperti di awal tadi. Jika pengaturan CGI berhasil, seharusnya Anda akan melihat laman web dengan tampilan sebagaimana program CGI di atas tadi.

Jika semua pengujian di atas, baik `index.html` atau `test1.cgi`, berhasil maka kita cukup bekerja di *folder* rumah. Dan berkas apapun yang ada di *folder* `web` (dalam *folder* rumah) bisa diakses dari luar melalui server web.

2. Pengaturan Pascal

Perintah kompilasi program CGI di atas masih cukup merepotkan karena kita mengetik nama berkas kode program sebanyak dua kali, kemudian menyalin berkas program ke *folder* CGI. Mengetik 1-2 kali mungkin tak masalah, tapi ketika dilakukan berulang kali, penghematan beberapa kata akan terasa. Untuk itu kita perlu bantuan *shell script* agar perintah kompilasi tadi menjadi lebih ringkas.

Untuk itu, lakukan langkah-langkah berikut ini:

1. Buat berkas melalui menu wadah, pilih **Create File**, dan tuliskan `pas.sh` pada jendela yang muncul, lalu klik tombol **OK**.
2. IDE akan membuat berkas baru dan langsung membukanya untuk disunting. Silakan ketikkan kode program berikut ke dalam berkas `pas.sh` tersebut.

```
# pisah nama dan akhiran berkas
ext=".pas"
fname=$(basename "$1")
bname=$(basename $fname $ext)
# folder tempat cgi
cgipath="/home/cabox/web/"

# kompilasi program
echo "Compiling:" fpc -XXs -CX -O3 -S2achi "$1" -o$bname.cgi "..."
fpc -XXs -CX -O3 -S2achi "$1" -o$bname.cgi

# uji hasil kompilasi
if [ -f $bname.o ]; then
    # pindahkan program ke foldernya
    echo "Deploying:" mv $bname.cgi $cgipath "..."
    mv $bname.cgi $cgipath
    # hapus berkas sampah
    echo "Cleaning:" rm $bname.o "..."
    rm $bname.o
    echo "Done."
else
    # kompilasi gagal
    echo "Error!"
    return 1
fi
```

3. Setelah selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau dengan pintasan **Ctrl+S**.
4. Buka atau kembali ke terminal, lalu ketikkan perintah berikut:

```
$ pas test1.pas
```

maka *shell script* `pas.sh` akan mengkompilasi berkas `test1.pas` dengan FPC (berikut berbagai *parameter* kompilasinya) serta hasilnya langsung bernama `test1.cgi` yang siap diakses. Kita bisa memberikan perintah lain ke dalam *script* `pas.sh` sesuai dengan yang kita butuhkan. Lebih lanjut tentang *shell script* bisa dibaca di [sini](#)³.

Berkas *shell script* `pas.sh` bisa langsung dipanggil dengan singkat karena di awal tadi kita telah mendaftarkan alias untuk berkas `pas.sh` di program awalan (*startup script*).

C. Menampilkan keluaran

Selanjutnya kita akan membuat program web dengan keluaran yang lebih ramai daripada sekedar teks *Hello World*. Langsung saja, mari kita kembali ke IDE CA kita, dan lakukan langkah-langkah berikut ini:

1. Buat berkas melalui menu wadah, pilih **Create File**, dan tuliskan `test2.pas` pada jendela yang muncul, lalu klik tombol **OK**.
2. IDE akan membuat berkas baru dan langsung membukanya untuk disunting. Silakan ketikkan kode program berikut ke dalam berkas `test2.pas` tersebut.

```
program webTest;
```

```
const
```

```
  Po = '<p>';    // buka paragraf
```

```
  Pc = '</p>';  // tutup paragraf
```

```
// menulis kepala html
```

```
procedure writeHeader(const aTitle: string; const loadCSS: string = '');
```

```
begin
```

```
  writeln('content-type: text/html;');
```

```
  writeln; // penting!
```

```
  writeln('<!DOCTYPE html>');
```

```
  writeln('<html><head>');
```

```
  writeln('<meta charset="utf-8">');
```

```
  writeln('<meta name="viewport" content="width=device-width,initial-scale=1">');
```

```
  if loadCSS <> '' then writeln('<link rel="stylesheet" href="",loadCSS,"">');
```

```
  writeln('<title>',aTitle,'</title>');
```

```
  writeln('</head><body>');
```

```
  writeln('<!-- isi laman mulai dari sini -->');
```

```
end;
```

```
// menulis kaki html
```

```
procedure writeFooter;
```

³ linuxcommand.org/learning_the_shell.php

```

begin
  writeln('<!-- isi laman berhenti di sini -->');
  writeln('</body></html>');
end;

// menulis judul html
procedure writeTitle(const aTitle: string; aLevel: integer = 1);
begin
  if aLevel < 1 then aLevel := 1; // nilai terendah
  if aLevel > 6 then aLevel := 6; // nilai tertinggi
  writeln('<h',aLevel,'>',aTitle,'</h',aLevel,'>');
end;

// menulis span html
function span(const aText:string; const aClass:string=''; const aID:string=''): string;
begin
  result := '<span';
  if aClass <> '' then result += ' class="'+aClass+'"' ;
  if aID <> '' then result += ' id="'+aID+'"' ;
  result += '>'+aText+'</span>';
end;

(** program utama **)
var
  s: string;
begin
  writeHeader('Hello world!','test.css');
  writeTitle('Hello world!');
  s := span('Free Pascal','bold');
  writeln(Po,'This page is created using ',s,' v3 and hosted on CodeAnywhere.','Pc');
  writeFooter;
end.

```

3. Kode program `test2.pas` terdiri dari 55 baris (termasuk baris kosong dan komentar) yang merupakan perbaikan dari kode program `test1.pas` dengan memisahkan bagian-bagian penulisan keluaran HTML menjadi prosedur dan fungsi yang sesuai.
4. Setelah selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau dengan pintasan **Ctrl+S**.
5. Di pohon berkas, klik kanan pada ikon *folder* **web**, lalu pilih **Create File**, dan tuliskan `test.css` pada jendela yang muncul, lalu klik tombol **OK**.
6. Ketikkan kode CSS berikut ke dalam berkas `test.css` tersebut.

```

body {
  font-family: Helvetica, Arial;
  font-size: 11pt;
  line-height: 1.2em;
  margin: 8px 12px;
}

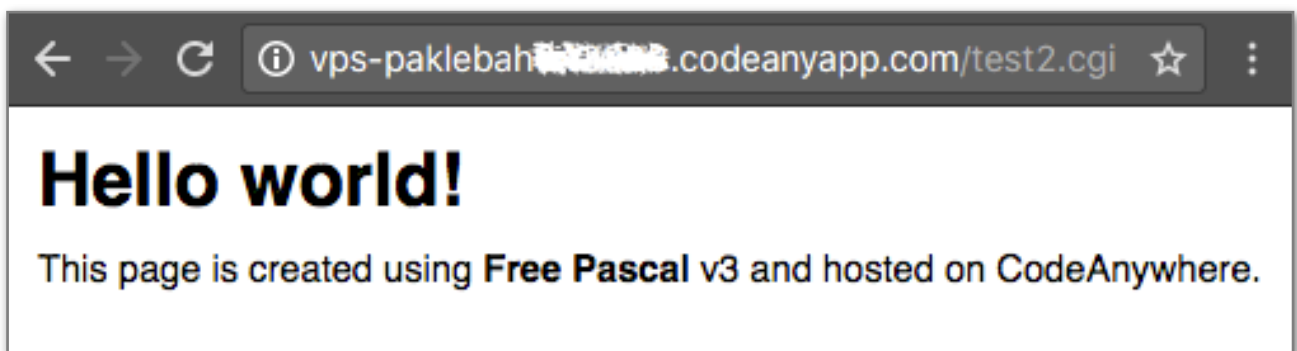
span.bold { font-weight: bold; }

```


- Setelah selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau dengan pintasan **Ctrl+S**. Perhatikan pohon berkas, pastikan `test.css` ada dalam *folder web*, bukan di *folder rumah*.
- Buka atau kembali ke terminal, lalu ketikkan perintah berikut:

```
$ pas test2.pas
```

yang akan mengkompilasi berkas `test2.pas` menjadi `test2.cgi` yang siap diakses.
- Buka *tab* peramban baru (atau *tab* yang sudah ada) lalu ketikkan alamat web VPS Anda diikuti `/test2.cgi` untuk melihat keluaran program. Keluaran `test2.cgi` seharusnya seperti gambar berikut ini.



Walau teksnya sama, perhatikan bahwa tampilan `test2.cgi` agak berbeda dari `test1.cgi` sebelumnya. Hal tersebut karena berkas `test.css` yang memperkaya gaya keluaran HTML. Untuk melihat isi berkas HTML peramban, klik-kanan di laman, pilih menu **View Source**. Berikut isi berkas HTML keluaran program `test2.cgi` di atas.

Untuk mempelajari HTML dan CSS lebih dalam, silakan ikuti salah satu panduan belajar web terbaik di internet, yaitu [w3schools.com](https://www.w3schools.com). Untuk referensi lain, bisa juga menggunakan situs htmlreference.io dan cssreference.io.

Contoh program di atas hanya menampilkan teks dari program. Selanjutnya kita membuat program untuk menampilkan informasi lingkungan sistem. Mari kita mulai...

1. Buat berkas melalui menu wadah, pilih **Create File**, dan tuliskan `test3.pas` pada jendela yang muncul, lalu klik tombol **OK**.
2. Silakan ketikkan kode program berikut ke dalam berkas `test3.pas` tersebut.

```
program webTest;

uses SysUtils;

const
  Po = '<p>';      // buka paragraf
  Pc = '</p>';     // tutup paragraf
  BR = '<br/>';    // baris baru

// menulis kepala html
procedure writeHeader(const aTitle: string; const loadCSS: string = '');
begin
  writeln('content-type: text/html;');
  writeln; // penting!
  writeln('<!DOCTYPE html>');
  writeln('<html><head>');
  writeln('<meta charset="utf-8">');
  writeln('<meta name="viewport" content="width=device-width,initial-scale=1">');
  if loadCSS <> '' then writeln('<link rel="stylesheet" href="",loadCSS,"">');
  writeln('<title>',aTitle,'</title>');
  writeln('</head><body>');
  writeln('<!-- isi laman mulai dari sini -->');
end;

// menulis kaki html
procedure writeFooter;
begin
  writeln('<!-- isi laman berhenti di sini -->');
  writeln('</body></html>');
end;

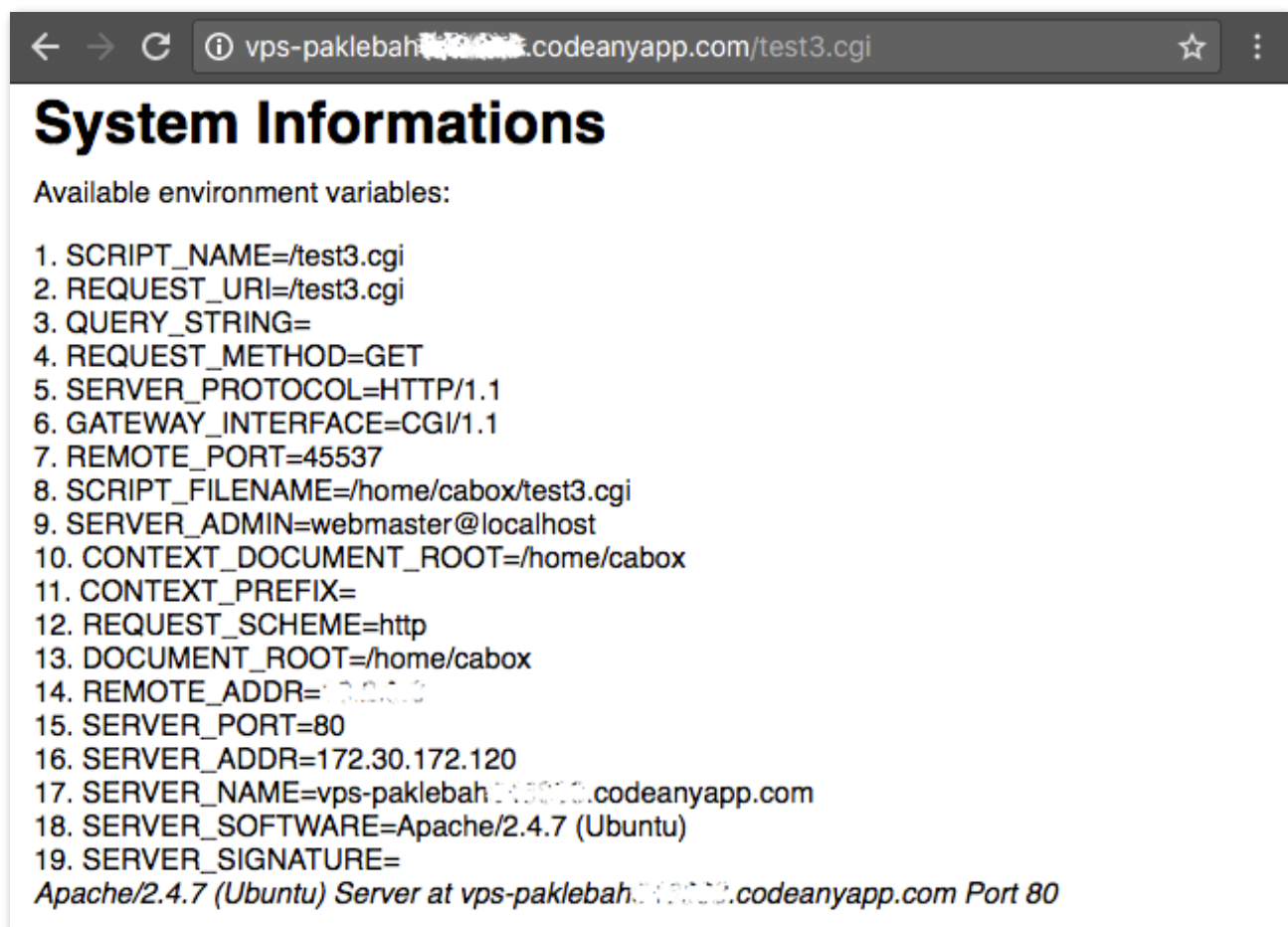
// menulis judul html
procedure writeTitle(const aTitle: string; aLevel: integer = 1);
begin
  if aLevel < 1 then aLevel := 1; // nilai terendah
  if aLevel > 6 then aLevel := 6; // nilai tertinggi
  writeln('<h',aLevel,'>',aTitle,'</h',aLevel,'>');
end;

// menulis span html
function span(const aText:string; const aClass:string=''; const aID:string=''): string;
begin
  result := '<span';
  if aClass <> '' then result += ' class="'+aClass+'";';
  if aID <> '' then result += ' id="'+aID+'";';
  result += '>'+aText+'</span>';
end;
```

```
// menulis isi laman
procedure writeContent;
var
  i: integer;
begin
  writeln(Po,'Available environment variables:',Pc);
  for i := 1 to getEnvironmentVariableCount do
    writeln(i:2,'. ',getEnvironmentString(i),BR);
end;

(** program utama **)
begin
  writeHeader('System Informations','test.css');
  writeTitle('System Informations');
  writeContent;
  writeFooter;
end.
```

3. Kode program `test3.pas` terdiri dari 65 baris dari pengembangan `test2.pas`, dengan tambahan prosedur `writeContent` untuk membaca dan menampilkan informasi sistem.
4. Setelah selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau dengan pintasan **Ctrl+S**.
5. Buka atau kembali ke terminal, lalu ketikkan perintah berikut:
\$ `pas test3.pas`



Tampilan keluaran program `test3.pas`.

6. Buka *tab* peramban baru (atau *tab* yang sudah ada) lalu ketikkan alamat web VPS Anda diikuti `/test3.cgi` untuk melihat keluaran program. Keluaran `test3.cgi` seharusnya seperti pada gambar di atas.

Keluaran `test3.cgi` adalah informasi dari server web yang diberikan kepada program CGI. Setiap baris informasi di atas mempunyai makna masing-masing. Penjelasan lebih rinci informasi sistem untuk program CGI bisa dibaca di [sini](#)⁴. Perlu diperhatikan bahwa urutan dan nilai keluaran program tak sama di tiap server web.

2. Catatan untuk proses awakutu

Berbeda dengan aplikasi meja (*desktop*) yang bisa dijalankan oleh IDE dan berinteraksi dengan IDE secara langsung, program CGI dijalankan oleh server web yang tergantung pada adanya permintaan peramban. Karena itu, proses awakutu (*debugging*) program CGI harus dilakukan manual dengan cara pencatatan (*logging*). Untungnya, cara ini juga tidak sulit karena Free Pascal telah menyediakan perintah untuk itu. Langsung saja kita simak caranya berikut ini...

1. Buat berkas melalui menu wadah, pilih **Create File**, dan tuliskan `test4.pas` pada jendela yang muncul, lalu klik tombol **OK**.
2. IDE akan membuat berkas baru dan langsung membukanya untuk disunting. Silakan ketikkan kode program berikut ke dalam berkas `test4.pas` tersebut.

```
program webTest;

uses
  SysUtils, DateUtils, EventLog;

const
  Po = '<p>';    // buka paragraf
  Pc = '</p>';   // tutup paragraf
  BR = '<br/>';  // baris baru

var
  log: TEventLog;    // objek catatan
  start: TDateTime;  // awal pewartuan

// menulis kepala html
procedure writeHeader(const aTitle: string; const loadCSS: string = '');
begin
  start := now;
  writeln('content-type: text/html;');
  writeln; // penting!
  writeln('<!DOCTYPE html>');
  writeln('<html><head>');
  writeln('<meta charset="utf-8">');
  writeln('<meta name="viewport" content="width=device-width,initial-scale=1">');
  if loadCSS <> '' then writeln('<link rel="stylesheet" href="',loadCSS,'">');
```

⁴ www.zytrax.com/tech/web/env_var.htm

```

        writeln('<title>',aTitle,'</title>');
        writeln('</head><body>');
        writeln('<!-- isi laman mulai dari sini -->');
        log.debug('#'+{$I %LINE%}+' : done writing http and html header');
    end;

// menulis kaki html
procedure writeFooter;
var
    d: integer;
begin
    d := trunc(milliSecondSpan(start,now));
    writeln('<p><small>This page is served in ',d,' ms.</small></p>');
    writeln('<!-- isi laman berhenti di sini -->');
    writeln('</body></html>');
    log.debug('#'+{$I %LINE%}+' : done writing html footer');
end;

// menulis judul html
procedure writeTitle(const aTitle: string; aLevel: integer = 1);
begin
    if aLevel < 1 then aLevel := 1; // nilai terendah
    if aLevel > 6 then aLevel := 6; // nilai tertinggi
    writeln('<h',aLevel,'>',aTitle,'</h',aLevel,'>');
    log.debug('#'+{$I %LINE%}+' : done writing html page title');
end;

// menulis span html
function span(const aText:string; const aClass:string=''; const aID:string=''): string;
begin
    result := '<span';
    if aClass <> '' then result += ' class="'+aClass+'";';
    if aID <> '' then result += ' id="'+aID+'";';
    result += '>'+aText+'</span>';
    log.debug('#'+{$I %LINE%}+' : done writing html span tag');
end;

// menulis isi laman
procedure writeContent;
var
    i: integer;
begin
    writeln(Po,'Available environment variables:',Pc);
    for i := 1 to getEnvironmentVariableCount do
        writeln(i:2,'. ',getEnvironmentString(i),BR);
    log.debug('#'+{$I %LINE%}+' : done reading environment vars');
end;

(*** program utama ***)
begin
    // menyiapkan objek catatan
    log := TEventLog.Create(nil);

```

```

log.logType := ltFile;
log.fileName := '/home/cabox/cgi.log';
log.identification := ExtractFilename(ParamStr(0))+':';
log.appendContent := true;
log.active := true;
// jalankan aplikasi
writeHeader('System Informations','test.css');
writeTitle('System Informations');
writeContent;
writeFooter;
// bebaskan objek catatan
log.free;
end.

```

3. Kode program `test4.pas` terdiri dari 90 baris yang merupakan pengembangan dari program `test3.pas`, dengan penambahan objek log untuk menyimpan catatan (*log*).
4. Setelah program selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau dengan pintasan **Ctrl+S**.
5. Kembali ke terminal, kemudian ketikkan perintah berikut secara berurutan:

```

$ touch cgi.log
$ sudo chown cabox:www-data cgi.log
$ sudo chmod 664 cgi.log

```

yang akan membuat berkas catatan bernama `cgi.log` di *folder* rumah yang bisa diakses oleh server web (dalam grup pengguna `www-data`). Berkas catatan harus dibuat terlebih dahulu sebelum program web menggunakannya.

6. Masih di terminal, ketikkan perintah berikut:


```
$ pas test4.pas
```
7. Sebelum membuka peramban dan mengakses program, di terminal ketikkan perintah:


```
$ tail -f cgi.log
```

yang akan terus menampilkan penambahan catatan terbaru di berkas `cgi.log` ke layar. Untuk menghentikan perintah `tail`, tekan **Ctrl+C** di terminal.
8. Buka *tab* peramban baru (atau *tab* yang sudah ada) lalu ketikkan alamat web VPS Anda diikuti `/test4.cgi` untuk menjalankan program `test4.cgi`.
9. Perhatian kita kali ini tidak lagi pada keluaran program `test4.cgi` di peramban karena sama saja dengan keluaran `test3.cgi` di atas. Yang perlu diperhatikan adalah keluaran di terminal dari perintah di langkah 6, seperti yang ditunjukkan di gambar berikut ini.

```

vps x * test4.pas x
cabox@cabox-codeanywhere:~$ tail -f cgi.log
test4.cgi: [2017-02-15 12:13:31.674 Debug] #27: done writing http and html header
test4.cgi: [2017-02-15 12:13:31.675 Debug] #44: done writing html page title
test4.cgi: [2017-02-15 12:13:31.675 Debug] #65: done reading environment vars
test4.cgi: [2017-02-15 12:13:31.675 Debug] #35: done writing html footer

```

Tampilan catatan program `test4.pas`.

Program membaca informasi sistem dengan prosedur `GetEnvironmentString()`. Baris program `log.debug('#'+{$I %LINE%}+' : text')` berfungsi untuk mencatat nama program, waktu, mode catatan, nomor baris, dan teks (bisa diisi apa saja) ke dalam berkas `cgi.log` dengan kelas `TEventLog`. Lebih jauh tentang `TEventLog` bisa dibaca di [sini](#)⁵. Kita bisa mencatatkan apapun ke berkas catatan saat program bekerja. Catatan inilah —bersama dengan perintah `tail`— yang bisa membantu kita dalam proses awakutu.

Setelah beberapa lama, berkas catatan akan membengkak dan membuat program bekerja lebih lambat. Untuk mengosongkan berkas catatan, jalankan perintah berikut di terminal:

```
$ > cgi.log
```

Memang teknik catatan ini kurang nyaman dan cukup merepotkan, tak seperti mekanisme awakutu di Delphi atau Lazarus IDE. Ini merupakan salah satu kelemahan program CGI. Jika Anda ingin mekanisme awakutu yang lebih nyaman seperti pada program meja, Anda harus membuat program web dengan metode *fast* CGI (FCGI) atau server web sendiri (*embedded*). Semoga metode ini bisa kita bahas di *kulgram* atau buku-el selanjutnya.

Selain itu, program `test4.pas` ini bisa mengukur berapa waktu yang dibutuhkan program untuk menyusun keluaran HTML. Caranya sederhana saja, silakan Anda perhatikan tambahan kode program di prosedur `writeHeader()` dan `writeFooter()` dan bandingkan dengan kode program `test3.pas` sebelumnya.

3. Membuat unit `webUtils.pas`

Program selanjutnya akan semakin panjang seiring dengan semakin banyak hal yang akan dilakukan. Karena itu, kita perlu membuat sebuah `unit` untuk menampung fungsi/prosedur yang sering digunakan dalam program utama. Lebih rinci tentang unit, bisa dibaca di [sini](#)⁶. Kita namakan unit ini `webUtils.pas` karena gunanya untuk membantu pembuatan program web, yang isinya sebagai berikut:

```
unit webUtils;

{$MODE OBJFPC} {$H+} {$J-}

interface

uses
  SysUtils, DateUtils, EventLog;

const
  // tanda elemen html
  Po  = '<p>';      // buka paragraf
  Pc  = '</p>';     // tutup paragraf
  BR  = '<br/>';    // baris baru
  HR  = '<hr/>';    // garis horisontal
  DIVc = '</div>';  // tutup div
```

⁵ freepascal.org/docs-html/current/fcl/eventlog/teventlog.html

⁶ <http://wiki.freepascal.org/Unit>

```

// nama berkas catatan, harus sudah dibuat
LOG_FILE = '/home/cabox/cgi.log';

var
  log: TEventLog; // pencatatan

procedure writeHeader(const aTitle:string; const loadCSS:string=''; const loadJS:string='');
procedure writeFooter(const loadCSS: string = ''; const loadJS: string = '');
procedure writeTitle(const aTitle:string; const aLevel:integer=1; const aClass:string='');

function span(const aText:string; const aClass:string=''; const aID:string=''): string;
function paraOpen(const aClass: string = ''; const aID: string = ''): string;
function divOpen(const aClass: string = ''; const aID: string = ''): string;

implementation

var
  start : TDateTime;
  isLogged: boolean = false; // pengaturan pencatatan unit

// menulis kepala html
procedure writeHeader(const aTitle:string; const loadCSS:string=''; const loadJS:string='');
begin
  start := now; // awal pewaktuan
  writeln('content-type: text/html;');
  writeln; // penting!
  writeln('<!DOCTYPE html>');
  writeln('<html><head>');
  writeln('<meta charset="utf-8">');
  writeln('<meta name="viewport" content="width=device-width,initial-scale=1">');
  if loadCSS <> '' then writeln('<link rel="stylesheet" href="",loadCSS,"">');
  if loadJS <> '' then writeln('<script src="",loadJS,""></script>');
  writeln('<title>',aTitle,'</title>');
  writeln('</head><body>');
  writeln('<!-- isi laman mulai dari sini -->');
  if isLogged then log.debug('#'+{$I %LINE%}+' : [webUtils.writeHeader] done. ');
end;

// menulis kaki html
procedure writeFooter(const loadCSS: string = ''; const loadJS: string = '');
var
  elapse: integer;
begin
  if loadCSS <> '' then writeln('<link rel="stylesheet" href="",loadCSS,"">');
  if loadJS <> '' then writeln('<script src="",loadJS,""></script>');
  elapse := trunc(milliSecondSpan(start,now)); // akhir pewaktuan
  writeln('<p><small>This page is served in ',elapse,' ms.</small></p>');
  writeln('<!-- isi laman berhenti di sini -->');
  writeln('</body></html>');
  if isLogged then log.debug('#'+{$I %LINE%}+' : [webUtils.writeFooter] done. ');
end;

```

```

// menulis judul laman
procedure writeTitle(const aTitle:string; const aLevel:integer=1; const aClass:string='');
begin
    if aClass <> '' then
        writeln('<h',aLevel,' class=""',aClass,'">',aTitle,'</h',aLevel,'>')
    else
        writeln('<h',aLevel,'>',aTitle,'</h',aLevel,'>');
    if isLogged then log.debug('#'+{$I %LINE%}+' : [webUtils.writeTitle] done. ');
end;

// membuat tanda span
function span(const aText:string; const aClass:string=''; const aID:string=''): string;
begin
    result := '<span';
    if aClass <> '' then result += ' class=""'+aClass+'""';
    if aID <> '' then result += ' id=""'+aID+'""';
    result += '>'+aText+'</span>';
    if isLogged then log.debug('#'+{$I %LINE%}+' : [webUtils.span] done. ');
end;

// membuat bukaan paragraf
function paraOpen(const aClass: string = ''; const aID: string = ''): string;
begin
    result := '<p';
    if aClass <> '' then result += ' class=""'+aClass+'""';
    if aID <> '' then result += ' id=""'+aID+'""';
    result += '>';
    if isLogged then log.debug('#'+{$I %LINE%}+' : [webUtils.paraOpen] done. ');
end;

// menulis bukaan tanda div
function divOpen(const aClass: string = ''; const aID: string = ''): string;
begin
    result := '<div';
    if aClass <> '' then result += ' class=""'+aClass+'""';
    if aID <> '' then result += ' id=""'+aID+'""';
    result += '>';
    if isLogged then log.debug('#'+{$I %LINE%}+' : [webUtils.divOpen] done. ');
end;

(***) awalan dan akhiran unit (***)

initialization
    // memastikan berkas catatan tersedia
    if fileExists(LOG_FILE) then
    begin
        log := TEventLog.Create(nil);
        log.logType := ltFile;
        log.fileName := LOG_FILE;
        log.identification := ExtractFilename(ParamStr(0))+':';
        log.appendContent := true;
        log.active := true;
    end;

```

```

    log.debug('#'+{$I %LINE%}+' : [webUtils.initialization] done.');
```

```
end;
```

```
finalization
  if fileExists(LOG_FILE) then
  begin
    log.debug('#'+{$I %LINE%}+' : [webUtils.finalization] done.');
```

```
    log.free;
```

```
  end;
```

```
end.
```

Dapat diperhatikan bahwa isi unit webUtils ini berasal dari program `test4.pas` dengan sedikit penyesuaian dan tambahan pengaturan pencatatan unit melalui peubah `isLogged` sebagai tanda keaktifan pencatatan fungsi/prosedur dalam unit webUtils. Selain itu juga dilakukan pengecekan keberadaan berkas catatan untuk menghindari kesalahan berkas. Selanjutnya, simpan unit ini dengan langkah-langkah berikut:

1. Buat berkas melalui menu wadah, pilih **Create File**, dan tuliskan `webUtils.pas` pada jendela yang muncul, lalu klik tombol **OK**.
2. IDE akan membuat berkas baru dan langsung membukanya untuk disunting. Silakan salin-tempel (*copy-paste*) kode program di atas ke dalam berkas `webUtils.pas` tersebut.
3. Kode program `webUtils.pas` terdiri dari 131 baris, pastikan proses salin-tempel tak mengurangi jumlah baris program.
4. Setelah selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau dengan pintasan **Ctrl+S**.

Selanjutnya kita perlu menguji unit webUtils ini dengan program baru. Untuk mudahnya, kita buat program yang serupa dengan `test4.pas` tapi dengan memanfaatkan unit webUtils ini. Ikuti langkah-langkah berikut:

1. Buat berkas melalui menu wadah, pilih **Create File**, dan tuliskan `test5.pas` pada jendela yang muncul, lalu klik tombol **OK**.
2. IDE akan membuat berkas baru dan langsung membukanya untuk disunting. Silakan ketikkan kode program berikut ke dalam berkas `test5.pas` tersebut.

```

program webTest;
```

```
uses
```

```
  SysUtils, webUtils;
```

```
// menulis isi laman
```

```
procedure writeContent;
```

```
var
```

```
  i: integer;
```

```
begin
```

```
  writeln(Po,'Available environment variables:',Pc);
```

```
  for i := 1 to getEnvironmentVariableCount do
```

```
    writeln(i:2,'. ',getEnvironmentString(i),BR);
```

```

    log.debug('#'+{$I %LINE%}+' : [main.writeContent] done.');
```

```
end;
```

```

(***) program utama (***)
begin
    writeHeader('System Informations','test.css');
    writeTitle('System Informations');
    writeContent;
    writeFooter;
end.
```

3. Kode program `test5.pas` terdiri dari hanya 23 baris saja karena fungsi/prosedur untuk penyusunan kode-kode HTML telah dipindahkan ke dalam unit `webUtils`.
4. Setelah selesai diketik, simpan berkas. Cara menyimpan berkas bisa melalui menu IDE, atau dengan pintasan **Ctrl+S**.
5. Kembali ke terminal, ketikkan perintah berikut:

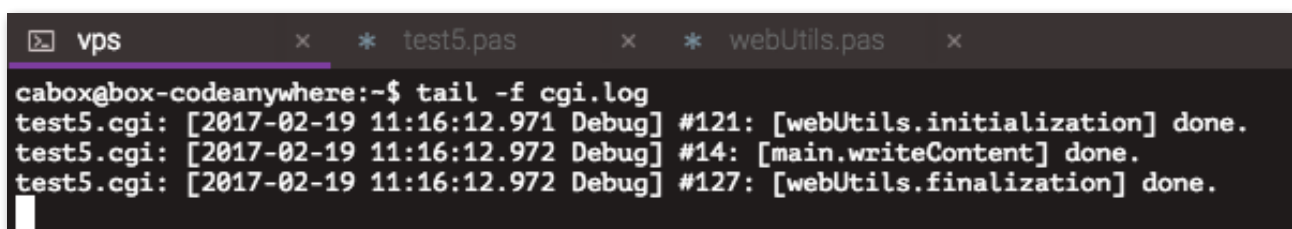
```
$ pas test5.pas
```

6. Sebelum membuka peramban dan mengakses program, di terminal ketikkan perintah:

```
$ tail -f cgi.log
```

yang akan menampilkan tambahan catatan terbaru di berkas `cgi.log` ke layar. Untuk menghentikan perintah `tail`, tekan **Ctrl+C** di terminal.

7. Buka `tab` peramban baru (atau `tab` yang sudah ada) lalu ketikkan alamat web VPS Anda diikuti `/test5.cgi` untuk menjalankan program `test5.cgi`.
8. Jika semua langkah di atas dijalankan dengan benar, seharusnya kita mendapatkan hasil seperti yang ditunjukkan pada gambar di bawah ini.



```

vps x * test5.pas x * webUtils.pas x
cabox@box-codeanywhere:~$ tail -f cgi.log
test5.cgi: [2017-02-19 11:16:12.971 Debug] #121: [webUtils.initialization] done.
test5.cgi: [2017-02-19 11:16:12.972 Debug] #14: [main.writeContent] done.
test5.cgi: [2017-02-19 11:16:12.972 Debug] #127: [webUtils.finalization] done.
```

Tampilan catatan program `test5.pas`.

Dengan menggunakan unit, program utama menjadi lebih ringkas dan kita tak perlu selalu melihat kode program yang sama berulang-ulang di kulgram/buku-el ini. Lalu, berhubung langkah-langkah membuat dan menyimpan berkas, kompilasi program, melihat catatan program, dan menjalankan program dari peramban telah berulang kali dijelaskan di atas, maka pada penjelasan selanjutnya keempat proses tersebut masing-masing saya ringkas menjadi satu langkah saja.

D. Menerima masukan

Membuat program tentu tak hanya bisa menampilkan keluaran tapi juga bisa menerima masukan. Pada program web, masukan diterima melalui permintaan (*request*) HTTP. Tapi sebelum program kita bisa menerima masukan, kita harus menyediakan antarmuka bagi

pengguna untuk memberi masukan ke program kita. HTML telah menyediakan banyak antarmuka agar pengguna bisa berinteraksi dengan program web. Berbagai jenis antarmuka masukan HTML yang tersedia bisa dibaca di [sini](http://htmlreference.io/forms/)⁷. Antarmuka masukan HTML dibungkus dalam pasangan tanda `<form>...</form>` untuk memicu pengiriman data masukan pengguna dari peramban ke server web. Data masukan dari peramban akan diterima program web dengan perantara server web melalui dua saluran berikut:

1. Menampilkan antarmuka masukan

Contoh program di atas hanya menampilkan teks dari program. Selanjutnya kita membuat program untuk menampilkan informasi lingkungan sistem. Langsung saja, kita kembali ke IDE CA kita, dan mulai menulis program baru.

1. Buat berkas program baru bernama `test6.pas` dan salin program berikut ini (124 baris):

```
program webTest;

uses
  SysUtils, webUtils;

const
  FORMc = '</form>'; // tutupan form

var
  inputIndex: integer = 0; // penghitung antarmuka masukan

// uji kondisi 2 teks
function switch(const condition: boolean; const ifTrue, ifFalse: string): string;
begin
  if condition then result := ifTrue else result := ifFalse;
end;

// membuat garis pemisah horisontal
function separator: string;
begin
  result := '<hr class="separator"/>'
end;

// membuat bukaan form
function formOpen(const action: string; const isPost: boolean = false): string;
begin
  result := '<form action="' + action + '" method="' + switch(isPost, 'post', 'get') + '">';
end;

// membuat ruang kosong sebelum masukan
function putSpace(const aLabel: string = ''): string;
begin
  result := '<span class="input">' + aLabel + '</span>';
end;

// membuat teks keterangan masukan
function putLabel(const aLabel: string; const forID: string = '_'): string;
```

⁷ <http://htmlreference.io/forms/>


```

begin
    result := '<label class="input"' ;
    if forID = '_' then
        result += ' for="input_'+IntToStr(inputIndex+1)+'">'
    else if forID <> '' then
        result += ' for="'+forID+'"'>'
    else
        result += '>';
    result += aLabel+'</label>';
end;

// membuat masukan teks
function inputText(const aValue: string = ''): string;
begin
    inputIndex := inputIndex+1;
    result := '<input type="text"' +
        ' id="input_'+IntToStr(inputIndex)+'"' +
        ' name="input_'+IntToStr(inputIndex)+'"' +
        ' switch(aValue='',',', value="'+aValue+'")>';
end;

// membuat masukan bilangan
function inputNumber(const aValue: integer = -1): string;
begin
    inputIndex := inputIndex+1;
    result := '<input type="number"' +
        ' id="input_'+IntToStr(inputIndex)+'"' +
        ' name="input_'+IntToStr(inputIndex)+'"' +
        ' switch(IntToStr(aValue)=-1',',',', value="'+IntToStr(aValue)+'")>';
end;

// membuat masukan logika
function inputBool(const aCaption: string; const aValue: boolean = false): string;
begin
    inputIndex := inputIndex+1;
    result := '<label><input type="checkbox"' +
        ' id="input_'+IntToStr(inputIndex)+'"' +
        ' name="input_'+IntToStr(inputIndex)+'"' +
        ' value="true"' +
        ' switch(aValue,' checked',',')+
        '> '+aCaption+' </label>';
end;

// membuat masukan memo
function inputMemo(const aValue: string = ''): string;
begin
    inputIndex := inputIndex+1;
    result := '<textarea' +
        ' id="input_'+IntToStr(inputIndex)+'"' +
        ' name="input_'+IntToStr(inputIndex)+'">'+
        aValue+'</textarea>';
end;

// membuat masukan tombol

```

```

function inputButton(const aCaption: string; const action: string = '';
                    const isReset: boolean = false): string;
begin
    inputIndex := inputIndex+1;
    result := '<button'+
        ' type="'+switch(isReset,'reset','submit')+'"' +
        ' id="button_'+IntToStr(inputIndex)+'"' +
        ' name="input_'+IntToStr(inputIndex)+'"' +
        switch(action='',',', ' formaction="'+action+'"' )+
        ' value="clicked">'+aCaption+'</button>';
end;

// menulis isi laman
procedure writeContent;
begin
    writeln(Po,formOpen(ExtractFilename(ParamStr(0))));
    writeln(putLabel('String: '),inputText,BR);
    writeln(putLabel('Number: '),inputNumber,BR);
    writeln(putLabel('Boolean: ', ''),inputBool('Yes, I agree'),BR);
    writeln(putLabel('Memo: '),inputMemo,BR);
    writeln(separator);
    writeln(putSpace,inputButton(' SUBMIT '),BR);
    writeln(FORMc,Pc,HR);
    log.debug('#'+{$I %LINE%}+: [main.writeContent] done. ');
end;

(*** program utama ***)
begin
    writeHeader('Read Input','test.css');
    writeTitle('READ INPUT',3);
    writeContent;
    writeFooter;
end.

```

2. Untuk menata tampilan supaya lebih bagus, buka kembali berkas [test.css](#) dan ganti isinya dengan kode-kode CSS berikut ini (berisi 77 baris):

```

body {
    font-family: Helvetica, Arial;
    font-size: 11pt;
    line-height: 1.8em;
    margin: 8px 12px;
}

hr {
    border: none;
    height: 1px;
    background-color: lightgray;
}

input { font-size: 11pt; }

select {
    border-color: lightgray;
    font-size: 11pt;
    width: 170px;
}

```

```

select[multiple] { margin: 4px 0 0 0; }

textarea {
  border-color: lightgray;
  margin: 4px 0 0 0;
  height: 100px;
  width: 165px;
  font-size: 10pt;
}

button { font-size: 10pt; }

table {
  font-size: 10pt;
  margin: 8px 0;
  border-bottom: 1px solid darkgray;
  border-collapse: collapse;
}

table th {
  background-color: lightgray;
  border-bottom: 1px solid darkgray;
  padding: 1px 6px;
}

table td {
  padding: 1px 6px;
  vertical-align: top;
}

table tr:nth-child(odd) {
  background-color: #f1f1f1;
}

table tr:hover {
  background-color: lightskyblue;
}

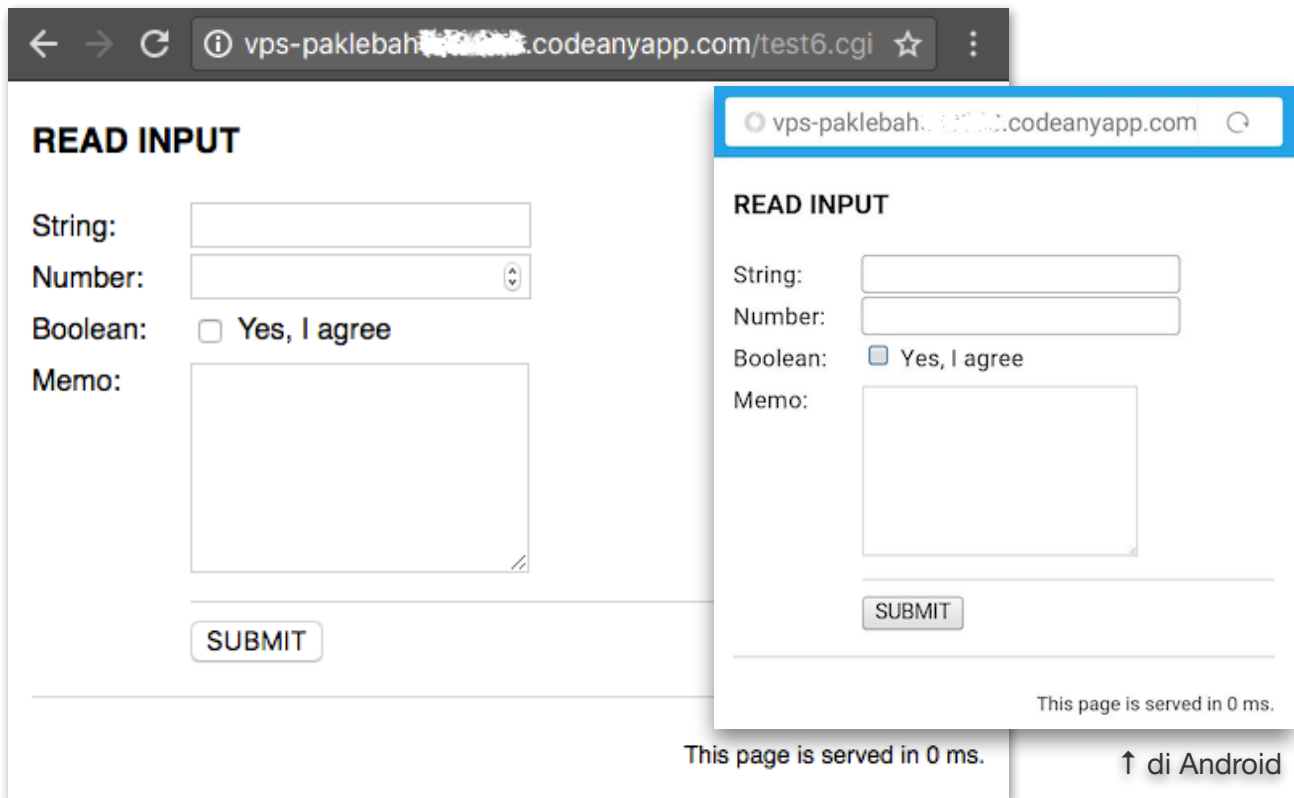
span.bold { font-weight: bold; }

hr.separator {
  border: none;
  height: 1px;
  margin: 6px 0 6px 80px;
  /* margin: 6px 0; */
  background-color: lightgray;
}

.input {
  vertical-align: top;
  display: inline-block;
  /* display: block; */
  width: 80px;
}

```

3. Simpan kedua berkas tersebut, kemudian kompilasi programnya, lalu buka program [test6.cgi](#) di peramban.
4. Jika semua di atas dilakukan benar maka seharusnya Anda akan melihat tampilan di peramban seperti pada gambar di bawah ini.



Tampilan keluaran program test6.pas.

Mari kita bedah apa yang dilakukan program `test6.pas` ini. Hal pertama adalah membuat susunan kode HTML untuk setiap antarmuka masukan. Ini dilakukan oleh fungsi-fungsi yang berawalan `inputXXX` seperti `inputText()` yang menampilkan kode HTML antarmuka masukan teks, dan sebagainya. Untuk menampilkannya, cukup panggil fungsi-fungsi yang dibutuhkan ke dalam prosedur `writeln()` seperti yang telah dicontohkan dalam prosedur `writeContent()`. Yang telah saya sediakan di atas hanya sedikit contoh dari sekian banyak jenis antarmuka masukan HTML yang tersedia, selanjutnya silakan Anda lengkapi sendiri untuk jenis antarmuka lainnya.

Kedua yaitu penghitung antarmuka masukan pada peubah `inputIndex`. Setiap antarmuka harus memiliki nama atau pengenalan yang unik agar masing-masing bisa dibaca nilainya. Cara yang paling mudah adalah dengan memberinya nomor urut. Itu sebabnya pada setiap fungsi `inputXXX` dilakukan penambahan nilai peubah `inputIndex`.

Kode HTML dan CSS tidak akan saya jelaskan karena itu di luar lingkup pembahasan kita. Jika Anda ingin memahami apa isi berkas `test.css` di atas, silakan pelajari CSS pada tautan yang telah diberikan sebelumnya. Termasuk juga bagaimana susunan kode HTML untuk berbagai jenis antarmuka, silakan pelajari sendiri. Pelajari juga bagaimana kode HTML yang dihasilkan oleh setiap prosedur/fungsi dalam program `test6.pas` di atas.

Lalu, apa yang akan terjadi ketika kita mengklik tombol SUBMIT pada program `test6.cgi` di atas? Ya, tidak terjadi apa-apa. Apa pun yang kita ketikkan ke dalam antarmuka masukan, tidak menghasilkan apa-apa. Karena memang program ini hanya menampilkan antarmuka saja, belum membaca data masukan pengguna yang dikirimkan oleh peramban. Ini akan kita bahas pada subbab selanjutnya.

Mengembangkan unit webUtils.pas

Untuk menyingkat dan mempermudah penulisan kode program kita dalam buku/kulgram ini maka setiap selesai satu contoh program maka Anda perlu memindahkan hal-hal baru di program utama ke dalam berkas `webutils.pas`. Dengan demikian, seiring berjalannya pembahasan, kemampuan unit `webUtils` juga akan semakin banyak dan ukurannya akan semakin besar, tapi tak perlu menuliskannya di sini berulang-ulang.

Contoh, sekarang kita telah selesai dengan program `test6.pas` di atas, maka seluruh peubah, konstanta, dan fungsi/prosedur baru kita pindahkan ke unit `webUtils`. Artinya, program `test6.pas` hanya akan berisi baris program, `uses`, prosedur `writeContent()`, serta bagian program utama, seperti berikut:

```
program webTest;

uses
  SysUtils, webUtils;

// menulis isi laman
procedure writeContent;
begin
  writeln(Po,formOpen(ExtractFilename(ParamStr(0))));
  writeln(putLabel('String: '),inputText,BR);
  writeln(putLabel('Number: '),inputNumber,BR);
  writeln(putLabel('Boolean: ',''),inputBool('Yes, I agree'),BR);
  writeln(putLabel('Memo: '),inputMemo,BR);
  writeln(separator);
  writeln(putSpace,inputButton(' SUBMIT '),BR);
  writeln(FORMc,Pc,HR);
  log.debug('#'+{$I %LINE%}+' : [main.writeContent] done.');
```

```
end;

(*** program utama ***)
begin
  writeHeader('Read Input','test.css');
  writeTitle('READ INPUT',3);
  writeContent;
  writeFooter;
end.
```

Pastikan setelah sebagian kode program utama dipindah ke unit `webUtils`, program utama tetap bisa dikompilasi dengan baik dan program bekerja dengan benar, persis sebagai mana sebelum pemindahan. Saya anggap Anda telah paham bagaimana mengolah kode program dalam sebuah unit. Silakan Anda pelajari struktur unit Pascal di [sini](#)⁸ dan [sini](#)⁹.

Setiap kita masuk ke pembahasan baru, maka saya menganggap Anda telah melakukan pemindahan tersebut sehingga di program baru kita hanya akan menulis kemampuan baru saja di program utama. Dan unit `webUtils` yang semakin kaya kemampuannya. Pada akhir buku/kulgram ini, kita akan mendapat unit `webUtils` yang lengkap.

⁸ <http://www.freepascal.org/docs-html/ref/refse105.html>

⁹ <http://wiki.freepascal.org/Unit>

2. Membaca data masukan web

Setelah bisa menampilkan antarmuka masukan web, selanjutnya adalah bagaimana cara membaca data masukan yang diterima melalui antarmuka masukan tersebut. Seperti yang telah dijelaskan di awal, ada 7 perintah permintaan HTTP. Namun jika melalui antarmuka, hanya tersedia 2 perintah saja, yaitu GET dan POST. Setidaknya ada 3 data masukan yang bisa kita baca, yaitu:

1. Data *query* dari URL untuk perintah GET yang tersedia melalui variabel sistem,
2. Data masukan dari perintah POST yang tersedia melalui jalur masukan baku (*std-in*),
3. Data kue (*cookie*) HTTP baik perintah GET atau POST yang tersedia di variabel sistem.

Bagaimana cara membaca data masukan tersebut? Mari kita buat program baru. Ikuti saja langkah-langkah berikut ini:

1. Buat berkas program baru bernama `test7.pas` dan salin program berikut ini (87 baris):

```
program webTest;

uses
  SysUtils, webUtils;

var
  webInputs  : string = '';    // data input http
  webQueries : string = '';    // data query http
  webCookies : string = '';    // data kue http
  allWebInput: string = '';    // seluruh data masukan http
  webHasInput: boolean = false; // status ketersediaan masukan

// baca data dari masukan baku (std-in)
function readWebInput: string;
var
  ch: char;
begin
  result := '';
  while not EOF(input) do
  begin
    read(ch);
    result += ch;
  end;
end;

// baca data query dari variabel sistem
function readWebQuery: string;
begin
  result := GetEnvironmentVariable('QUERY_STRING');
end;

// baca data kue dari variabel sistem
function readWebCookie: string;
begin
  result := GetEnvironmentVariable('HTTP_COOKIE');
end;
```



```

// baca dan gabungkan seluruh data masukan web
function readAllWebInput(const includeCookies: boolean = false): string;
begin
    webInputs    := readWebInput;
    webQueries   := readWebQuery;
    webHasInput  := (webQueries <> '') or (webInputs <> '');
    result       := switch(webQueries='', switch(webInputs='', '', webInputs),
                           switch(webInputs='', webQueries, webQueries+'&'+webInputs));
    if includeCookies then
    begin
        webCookies := readWebCookie;
        result      := switch(webCookies='', switch(result='', '', result),
                               switch(result='', webCookies, webCookies+'&'+result));
    end;
    webHasInput := (result <> '');
end;

// menulis isi laman
procedure writeContent;
begin
    // tampilkan antarmuka masukan
    writeln(Po, formOpen(ExtractFilename(ParamStr(0))));
    writeln(putLabel('String: '), inputText, BR);
    writeln(putLabel('Number: '), inputNumber, BR);
    writeln(putLabel('Boolean: ', ''), inputBool('Yes, I agree'), BR);
    writeln(putLabel('Memo: '), inputMemo, BR);
    writeln(separator);
    writeln(putSpace, inputButton(' SUBMIT '), BR);
    writeln(FORMc, Pc, HR);

    // tampilkan masukan, jika ada
    if webHasInput then
    begin
        writeln(Po, span('Accepted input:', 'bold'), BR);
        writeln('Input: ', switch(webInputs = '', '[empty]', webInputs), BR);
        writeln('Query: ', switch(webQueries = '', '[empty]', webQueries), BR);
        writeln('Cookie: ', switch(webCookies = '', '[empty]', webCookies), BR);
        writeln(Pc, HR);
    end;
    log.debug('#'+{$I %LINE%}+' : [main.writeContent] done. ');
end;

(** program utama **)
begin
    writeHeader('Read Input', 'test.css');
    writeTitle('READ INPUT', 3);
    allWebInput := readAllWebInput;
    writeContent;
    writeFooter;
end.

```

2. Simpan berkas, kompilasi program, lalu buka [test7.cgi](#) di peramban. Kemudian isilah setiap antarmuka masukan yang tampil, lalu klik tombol **SUBMIT**.
3. Jika semua di atas dilakukan benar maka seharusnya Anda akan melihat tampilan di peramban seperti pada gambar di bawah ini.

The screenshot shows a web browser window with the address bar displaying `vps-paklebah[REDACTED].codeanyapp.com/test7.cgi?input_1=abc&input_2=123&input_3=true&input_4=this+is+a+long+text&input_5=clicked`. The page content is divided into two sections. The top section, titled "READ INPUT", contains a form with four fields: "String:" with a text input containing "abc", "Number:" with a text input containing "123", "Boolean:" with a checkbox labeled "Yes, I agree" that is checked, and "Memo:" with a large text area. Below these fields is a "SUBMIT" button. The bottom section, titled "Accepted input:", displays the data received: "Input: [empty]", "Query: input_1=abc&input_2=123&input_3=true&input_4=this+is+a+long+text&input_5=clicked", and "Cookie: [empty]". At the bottom right of the page, it says "This page is served in 0 ms."

Tampilan program `test7.pas` dengan masukan `abc`, `123`, centang, dan `this is a long text`.

Perhatikan apa yang muncul di kolom alamat peramban setelah Anda mengklik tombol **SUBMIT**. Ya, data yang kita masukkan menjadi bagian dari URL alamat program web `test7.cgi`, yaitu sebagai *query*. Hal ini karena program kita menggunakan perintah GET. Lihat lagi isi fungsi `formOpen()` di unit `webUtils` yang secara bawaan menggunakan GET.

Untuk mengubahnya menjadi POST, cukup atur parameter kedua fungsi `formOpen()` yaitu parameter `isPost` menjadi bernilai `true`. Ubahlah pemanggilan `formOpen()` dalam program `test7.pas` di baris ke-59 menjadi seperti berikut:

```
writeln(Po, formOpen(ExtractFilename(ParamStr(0)), true));
```

Kemudian simpan berkas, kembali ke terminal, kompilasi program `test7.pas`, lalu buka `test7.cgi` di peramban. Jika semua benar maka bagian *query* pada URL menjadi kosong dan data masukan akan muncul pada baris **Input** (semula kosong), bukan lagi pada baris **Query**. Karena permintaan dikirim dengan perintah POST maka data tersedia pada lajur masukan baku (*standard input*).

Jika Anda ingin menjadikan perintah POST sebagai bawaan (*default*) fungsi `formOpen()` maka ubahlah parameter `isPost` di judul fungsi `formOpen()` di unit `webUtils.pas` menjadi bernilai `true` seperti berikut:

```
function formOpen(const action: string; const isPost: boolean = true): string;
```

Sebelum kita meneruskan pembahasan, seperti yang telah saya jelaskan di atas, silakan pindahkan dulu kode program kemampuan baru yang telah kita pelajari ke dalam unit `webUtils`. Sedemikian hingga program `test7.pas` di atas hanya berisi baris program, `uses`, prosedur `writeContent()`, serta bagian program utama. Setelah pemindahan, pastikan program `test7.pas` tetap bisa dikompilasi dan bekerja dengan benar.

3. Memilih dan memilah data masukan

Membaca data masukan web tidaklah sulit karena semuanya bisa diambil dengan mudah melalui jalur masukan baku dan peubah sistem. Namun jika kita perhatikan keluaran program `test7.cgi`, data masukan yang diterima server web telah mengalami perubahan penulisan. Contohnya pada masukan *memo*, spasi berubah menjadi tanda + (tambah).

Tak hanya itu saja, jika kita jalankan lagi program `test7.cgi` dan coba masukkan tulisan `~!@#$$%^&*()_+ (semua huruf di atas angka di papan ketik, ketik dengan tombol shift)`, lalu kita lihat hasilnya. Ya, hasilnya adalah `%7E%21%40%23%24%25%5E%26*%28%29_%2B` yang seolah sama sekali berbeda dengan masukan yang kita ketikkan di peramban. Ini disebut dengan *percent encoding* (penulisan dengan tanda persen) untuk menuliskan data-data selain alfanumerik (huruf dan angka normal).

Selain itu, perhatikan lagi gambar di atas, tampak bahwa data masukan berupa sebuah teks (*string*) panjang dengan pemisah tanda & (*ampersand*). Artinya, kita harus memilih dan memilah data yang ingin kita baca dari rangkaian teks panjang tersebut. Penulisan ini adalah baku penulisan data HTTP (*HTTP encoding*), termasuk *percent encoding* di atas. Dengan mengetahui cara penulisannya maka kita bisa membaca data HTTP dengan benar.

Selain *HTTP encoding* ada juga cara baku penulisan data HTML (*HTML encoding*). *HTTP encoding* berbeda dengan *HTML encoding*. Jika *HTTP encoding* melibatkan tanda persen dan tambah, *HTML encoding* melibatkan setidaknya 4 karakter, yaitu & (*ampersand*), " (petik ganda), < (lebih kecil), dan > (lebih besar). Keempat karakter tersebut tak boleh dituliskan apa adanya dalam HTML, tetapi harus diubah menjadi karakter khusus HTML, yaitu `&`, `"`, `<`, dan `>` (termasuk tanda ; di akhir) sesuai urutan sebelumnya.

Baiklah... mari kita langsung saja membuat program baru tentang bagaimana membaca data HTTP dan menuliskannya ke berkas HTML dengan benar. Seperti biasa, mari ikuti langkah-langkah berikut:

1. Buat berkas program baru bernama `test8.pas` dan salin program berikut ini (173 baris):

```
program webTest;

uses
  SysUtils, StrUtils, webUtils;

// menerjemahkan teks normal ke teks html (berkode &)
function htmlEncode(const aText: string): string;
begin
  result := aText;
  // penggantian tanda & harus dilakukan paling awal
  result := stringReplace(result, '&', '&amp;', [rfReplaceAll]);
```

```

    result := stringReplace(result, '"', '&quot;', [rfReplaceAll]);
    result := stringReplace(result, '<', '&lt;', [rfReplaceAll]);
    result := stringReplace(result, '>', '&gt;', [rfReplaceAll]);
    log.debug('#'+{$I %LINE%}+: @htmlEncode '+aText+' ⇒ '+result');
end;

// menerjemahkan teks http (berkode %) ke teks normal
function httpDecode(const encodedText: string): string;
var
    b: byte;
    c: word;
    l, p: integer;
    s, r, res: string;
begin
    // ganti seluruh tanda + menjadi spasi
    res := stringReplace(encodedText, '+', ' ', [rfReplaceAll]);
    l := length(res);
    p := 1;
    // cari tanda % satu per satu
    repeat
        p := posEx('%', res, p);
        if p > 0 then
            begin
                s := copy(res, p+1, 2);
                val('$'+s, b, c); // % dlm hexa
                // ganti tanda % menjadi huruf yg sesuai
                if c = 0 then
                    begin
                        r := chr(b); // hexa ke huruf
                        res := stringReplace(res, '%'+s, r, [rfIgnoreCase]);
                    end;
                p += 1;
            end;
        until (p = 0) or (p > l);
        result := res;
        log.debug('#'+{$I %LINE%}+: @httpDecode '+encodedText+' ⇒ '+res');
    end;

// mengambil nilai dari pasangan key=value dalam data masukan web
function getValue(const aKey, ofString: string; const delimiter: string = '&'): string;
var
    pStart, pStop: integer;
    s: string;
begin
    s := lowerCase(ofString);
    pStart := pos(lowerCase(aKey)+'=', s);
    if pStart > 0 then
        begin
            pStop := posEx(delimiter, s, pStart);
            pStart := pStart + length(aKey) + 1;
            if pStop = 0 then pStop := length(s) + 1;
            result := copy(ofString, pStart, pStop-pStart);
        end
    else result := '';
    log.debug('#'+{$I %LINE%}+: @getValue '+aKey+' = '+result);
end;

```

```

end;

// gabungkan antarmuka masukan dan pembacaan data masukan [string]
procedure webRead(var str: string; const newLine: boolean = false);
var
  s: string;
begin
  // baca data masukan
  s := getValue('input_'+intToStr(inputIndex+1),allWebInput);
  if s <> '' then str := s;
  // tampilan antarmuka masukan
  write(inputText(htmlEncode(httpDecode(str))));
  if newLine then writeln(BR) else writeln;
end;

// gabungkan antarmuka masukan dan pembacaan data masukan [integer]
procedure webRead(var int: integer; const newLine: boolean = false);
var
  s: string;
begin
  // baca data masukan
  s := getValue('input_'+intToStr(inputIndex+1),allWebInput);
  if s <> '' then int := strToInt(s);
  // tampilan antarmuka masukan
  write(inputNumber(int));
  if newLine then writeln(BR) else writeln;
end;

// gabungkan antarmuka masukan dan pembacaan data masukan [boolean]
procedure webRead(var bool: boolean; const aCaption: string; const newLine: boolean = false);
var
  s: string;
begin
  // baca data masukan
  s := getValue('input_'+intToStr(inputIndex+1),allWebInput);
  if webHasInput then bool := (s = 'true');
  // tampilan antarmuka masukan
  write(inputBool(aCaption,bool));
  if newLine then writeln(BR) else writeln;
end;

// akhiran ln untuk membuat baris baru setelah antarmuka masukan
procedure webReadLn(var str: string);
begin
  webRead(str,true);
end;

procedure webReadLn(var int: integer);
begin
  webRead(int,true);
end;

procedure webReadLn(var bool: boolean; const aCaption: string);
begin
  webRead(bool,aCaption,true);
end;

```

```

// gabungkan antarmuka masukan dan pembacaan data masukan [memo]
procedure webReadMemo(var str: string; const newLine: boolean = true);
var
  s: string;
begin
  // baca data masukan
  s := getValue('input_'+intToStr(inputIndex+1),allWebInput);
  if s <> '' then str := s;
  // tampilan antarmuka masukan
  write(inputMemo(htmlEncode(httpDecode(str))));
  if newLine then writeln(BR) else writeln;
end;

// menulis isi laman
procedure writeContent;
var
  m: string = '';
  s: string = '';
  i: integer = -1;
  b: boolean = true;
begin
  // tampilkan antarmuka masukan
  writeln(Po,formOpen(ExtractFilename(ParamStr(0)),true));
  write(putLabel('String: ')); webReadln(s);
  write(putLabel('Number: ')); webReadln(i);
  write(putLabel('Boolean: ', '')); webReadln(b,'Yes, I agree');
  write(putLabel('Memo: ')); webReadMemo(m);
  writeln(separator);
  writeln(putSpace,inputButton(' SUBMIT '),BR);
  writeln(FORMc,Pc,HR);

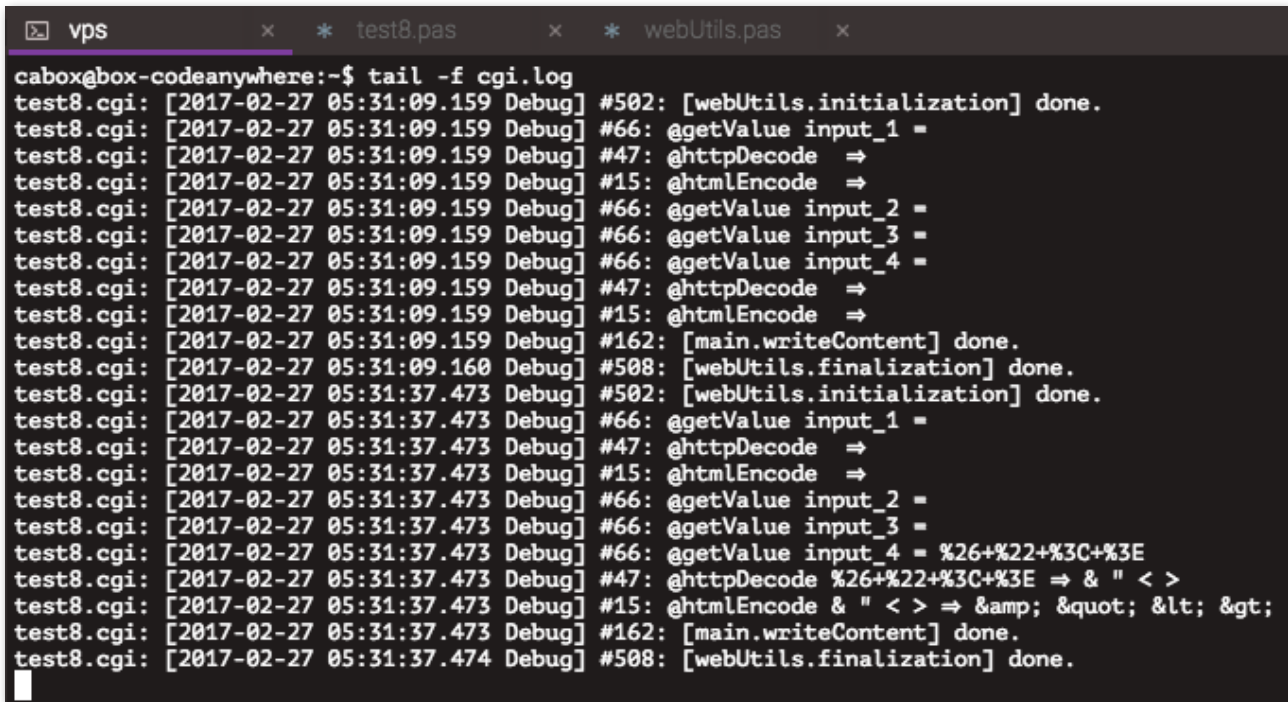
  // tampilkan masukan, jika ada
  if webHasInput then
  begin
    writeln(Po,span('Accepted input:', 'bold'),BR);
    writeln('Input: ',switch(webInputs = '', '[empty]',webInputs),BR);
    writeln('Query: ',switch(webQueries='', '[empty]',webQueries),BR);
    writeln(Pc,HR);
  end;
  log.debug('#'+{$I %LINE%}+' : [main.writeContent] done. ');
end;

(*** program utama ***)
begin
  writeHeader('Read Input','test.css');
  writeTitle('READ INPUT',3);
  allWebInput := readAllWebInput;
  writeContent;
  writeFooter;
end.

```

2. Simpan berkas, kompilasi program, lalu buka [test8.cgi](#) di peramban. Kemudian isikan teks & “ < > ke masukan *memo*, lalu klik tombol **SUBMIT**. Teks masukan tersebut untuk menguji apakah fungsi `htmlEncode()`, `httpDecode()`, dan `getValue()` bekerja dengan benar.

3. Jika kita lihat catatan CGI ([cgi.log](#)) setelah klik tombol **SUBMIT**, kita dapatkan hasil seperti pada gambar berikut ini. Perhatikan teks keluaran pada baris dengan penanda @httpDecode, @htmlEncode, dan @getValue pada data input_4.



```
vps x * test8.pas x * webUtils.pas x
cabox@box-codeanywhere:~$ tail -f cgi.log
test8.cgi: [2017-02-27 05:31:09.159 Debug] #502: [webUtils.initialization] done.
test8.cgi: [2017-02-27 05:31:09.159 Debug] #66: @getValue input_1 =
test8.cgi: [2017-02-27 05:31:09.159 Debug] #47: @httpDecode =>
test8.cgi: [2017-02-27 05:31:09.159 Debug] #15: @htmlEncode =>
test8.cgi: [2017-02-27 05:31:09.159 Debug] #66: @getValue input_2 =
test8.cgi: [2017-02-27 05:31:09.159 Debug] #66: @getValue input_3 =
test8.cgi: [2017-02-27 05:31:09.159 Debug] #66: @getValue input_4 =
test8.cgi: [2017-02-27 05:31:09.159 Debug] #47: @httpDecode =>
test8.cgi: [2017-02-27 05:31:09.159 Debug] #15: @htmlEncode =>
test8.cgi: [2017-02-27 05:31:09.159 Debug] #162: [main.writeContent] done.
test8.cgi: [2017-02-27 05:31:09.160 Debug] #508: [webUtils.finalization] done.
test8.cgi: [2017-02-27 05:31:37.473 Debug] #502: [webUtils.initialization] done.
test8.cgi: [2017-02-27 05:31:37.473 Debug] #66: @getValue input_1 =
test8.cgi: [2017-02-27 05:31:37.473 Debug] #47: @httpDecode =>
test8.cgi: [2017-02-27 05:31:37.473 Debug] #15: @htmlEncode =>
test8.cgi: [2017-02-27 05:31:37.473 Debug] #66: @getValue input_2 =
test8.cgi: [2017-02-27 05:31:37.473 Debug] #66: @getValue input_3 =
test8.cgi: [2017-02-27 05:31:37.473 Debug] #66: @getValue input_4 = %26+%22+%3C+%3E
test8.cgi: [2017-02-27 05:31:37.473 Debug] #47: @httpDecode %26+%22+%3C+%3E => & " < >
test8.cgi: [2017-02-27 05:31:37.473 Debug] #15: @htmlEncode & " < > => &amp; &quot; &lt; &gt;
test8.cgi: [2017-02-27 05:31:37.473 Debug] #162: [main.writeContent] done.
test8.cgi: [2017-02-27 05:31:37.474 Debug] #508: [webUtils.finalization] done.
```

Tampilan catatan program test8.pas.

Mari kita bahas apa saja yang dilakukan oleh program `test8.pas` di atas. Pertama, fungsi `htmlEncode()` (mulai baris ke-7) adalah untuk menerjemahkan teks normal menjadi teks terkodekan HTML. Caranya mudah yaitu dengan memanggil fungsi `stringReplace()` yang tersedia melalui unit `SysUtils` untuk setiap huruf yang akan diterjemahkan.

Kedua, fungsi `httpDecode()` (mulai baris ke-19) adalah untuk menerjemahkan teks terkode HTTP menjadi teks normal. Caranya sedikit panjang, yaitu mencari pasangan `%xx` dimana `xx` adalah dua angka dalam heksadesimal yang merupakan kode huruf ASCII. Selanjutnya, tinggal mengganti pasangan `%xx` tersebut dengan huruf ASCII yang sebenarnya. Daftar padanan angka dan huruf ASCII dan HTML bisa dibaca di [sini](#)¹⁰.

Ketiga, fungsi `getValue()` (mulai baris ke-50) adalah untuk membaca sebuah data berupa pasangan `key=value` berdasarkan kata kuncinya, dalam sebuah teks yang panjang dimana setiap data dipisahkan oleh pemisah (*delimiter*). Caranya mudah saja, yaitu mencari teks kata kunci diikuti tanda `=` (sama dengan). Jika ditemukan, baca teks selanjutnya hingga bertemu dengan tanda pemisah. Maka data yang dimaksud adalah teks yang berada antara tanda `=` (sama dengan) dan tanda pemisah (yaitu huruf `&` dalam data HTTP).

Pada contoh di atas, perhatikan keluaran catatan program di data `input_4`. Data yang diterima dari peramban untuk kunci `input_4` adalah `%26+%22+%3C+%3E` kemudian fungsi `httpDecode()` mengubahnya menjadi `& " < >` untuk diolah program, kemudian fungsi `htmlEncode()` mengubahnya menjadi `& " < >` untuk tampil di peramban.

¹⁰ <http://www.ascii.cl/htmlcodes.htm>

Keempat, prosedur `webRead/ln()` adalah untuk membaca data, baik dari masukan web atau dari program, kemudian menampilkan data tersebut ke antarmuka masukan HTML terkait. Prosedur yang berakhiran `ln` memberikan garis baru setelah tampilan antarmuka, seperti prosedur klasik Pascal `readln()`.

Untuk lebih memahami alur dan bagaimana program di atas bekerja, silakan Anda ubah-ubah data masukannya. Bisa data awal dari kode program (mulai baris ke-139 di prosedur `writeContent()`) atau data masukan dari pengguna di peramban. Coba eksperimen dengan berbagai variasi masukan, lalu perhatikan keluaran catatan `cgi.log` di terminal.

E. Mengolah data

Kita telah bisa menerima masukan data dan menampilkan keluaran data. Selanjutnya kita akan belajar mengolah data. Untuk mengolah data, umumnya kita menggunakan *database* tetapi *database* saya rasa terlalu rumit dan di luar lingkup bahasan kita. Sebagai gantinya, kita akan mengolah data dengan cara yang lebih sederhana dan mudah. Kita akan gunakan fasilitas memori bersama (*shared memory*) di Linux.

1. Mengolah data dengan memori bersama

Pada setiap sistem operasi biasanya tersedia fitur berbagi memori atau memori bersama (*shared memory*). Memori bersama adalah sebagian pingingat komputer yang dialokasikan khusus untuk dipakai bersama oleh beberapa aplikasi secara bersamaan. Ini salah satu fitur dasar dalam sistem operasi.

Free Pascal telah menyediakan fungsi-fungsi untuk bekerja dengan memori bersama di Linux. Hak akses ke memori mengikuti pengaturan hak akses pengguna dan kelompok yang berlaku di Linux. Untuk lebih memudahkan penggunaannya, mari kita buat *unit* sendiri. Kita namakan *unit* `sharedMem.pas` yang isinya sebagai berikut (76 baris):

```
unit sharedMem;

{$MODE OBJFPC} {$H+} {$J-}
{$WARN 4056 OFF} // omit warning for pointer portability
{$WARN 4082 OFF} // omit warning for pointer casting

interface

type
  TSharedMem = record
    ID: longint;      // data identifier
    Data: pChar;      // saved data (as string)
    Size: integer;    // memory size
    Length: integer;  // data length
  end;

function memCreate(var sharedMem:TSharedMem;const memSize:integer=255;const filePath:string=''):boolean;
function memWrite(var sharedMem: TSharedMem; const str: string): boolean;
function memRead(const sharedMem: TSharedMem; var str: string): boolean;
function memDestroy(const sharedMem: TSharedMem): boolean;
```

implementation

uses

IPC, Strings, BaseUnix, SysUtils;

// create a new and/or attach to existing shared memory

function memCreate(var sharedMem:TSharedMem;const memSize:integer=255;const filePath:string=''):boolean;

var

key: TKey;

path: string;

begin

// setup shared memory identifier

if filePath = '' then path := ExtractFilename(ParamStr(0))+#0

else path := filePath+#0;

key := ftok(pChar(@path[1]),ord(path[1]));

// try to create a new shared memory

sharedMem.Size := memSize;

sharedMem.ID := shmGet(key, sharedMem.Size, IPC_CREAT or IPC_EXCL or &0660);

// if failed then link it to existing shared memory

if sharedMem.ID = -1 then sharedMem.ID := shmGet(key, sharedMem.Size, 0);

// attach to created or existing shared memory

sharedMem.Data := shmAt(sharedMem.ID, nil, 0);

result := (longint(sharedMem.Data) <> -1);

end;

// write data to shared memory

function memWrite(var sharedMem: TSharedMem; const str: string): boolean;

var

s: string;

begin

result := (longint(sharedMem.Data) <> -1);

if result then

begin

s := str;

// cut text if it's too long

if length(s) > sharedMem.Size then s := copy(str,1,sharedMem.Size);

strpCopy(sharedMem.Data, s);

sharedMem.Length := length(s);

end;

end;

// read data from shared memory

function memRead(const sharedMem: TSharedMem; var str: string): boolean;

begin

result := (longint(sharedMem.Data) <> -1);

if result then str := string(sharedMem.Data);

end;

// destroy a shared memory

function memDestroy(const sharedMem: TSharedMem): boolean;

begin

result := (shmCtl(sharedMem.ID, IPC_RMID, nil) <> -1);

end;

end.

Unit `sharedMem.pas` dibuat untuk mendukung kita belajar mengolah data sederhana. Cara kerja memori bersama secara rinci tidak akan kita bahas di sini karena di luar lingkup bahasan kita.¹¹ *Unit* ini berisi empat fungsi utama, yaitu:

- `memCreate()` untuk membuat memori bersama baru **atau** membuka akses ke memori bersama yang telah ada. Sebelum mengolah data di memori bersama, program harus memanggil fungsi ini terlebih dahulu.
- `memWrite()` untuk menyimpan data bertipe teks ke memori bersama.
- `memRead()` untuk membaca data sebagai teks dari memori bersama.
- `memDestroy()` untuk menghapus memori bersama.

Linux menyediakan perintah-perintah untuk melihat status dan informasi memori bersama di sistem. Caranya, ketik perintah berikut di terminal:

```
$ sudo ipcs -m
```

maka akan ditampilkan status memori bersama di sistem. Jika program web macet yang berakibat memori bersama yang dibuatnya tidak bisa dihapus dengan *unit* `sharedMem.pas` maka ketik perintah berikut di terminal:

```
$ sudo ipcrm -i shmid
```

dengan `shmid` adalah nomor identitas memori bersama yang akan dihapus. Informasi `shmid` bisa dilihat dari keluaran perintah `ipcs` di atas.

2. Mengolah data masukan

Pada program `test8.pas` sebelumnya, kita telah bisa membaca dan menampilkan kembali data masukan dari peramban. Namun data belum tersimpan sehingga setiap kali kita membuka program `test8.cgi`, tidak ada data yang bisa ditampilkan. Sekarang kita telah mempunyai perangkat untuk mengolah data dengan *unit* `sharedMem`, mari kita mulai menggunakan *unit* tersebut untuk menyimpan data di memori bersama.

Langsung saja, mari kita buat program baru untuk menyimpan data dari peramban ke memori bersama. Silakan ikuti langkah-langkah berikut:

1. Buat berkas program baru bernama `test9.pas` dan salin program berikut ini (180 baris):

```
program webTest;

uses
  SysUtils, sharedMem, webUtils;

var
  webData: TSharedMem;

// menyisipkan javascript ke dokumen html
procedure writeScript(const aScript: string);
begin
  writeln('<script>',LE,aScript,'</script>',LE);
end;
```

¹¹ Sebagai awalan, bisa dimulai dari sini: <http://www.makelinux.net/alp/035>

```

// mengubah isian masukan dgn kode javascript [string]
function jsFillValue(const aID: string; const aValue: string): string;
begin
    result := 'document.getElementById("' + aID + '").value = "' + aValue + '";' + LE;
end;

// mengubah isian masukan dgn kode javascript [integer]
function jsFillValue(const aID: string; const aValue: integer): string;
begin
    result := 'document.getElementById("' + aID + '").value = ' + intToStr(aValue) + ';' + LE;
end;

// mengubah isian masukan dgn kode javascript [boolean]
function jsFillValue(const aID: string; const aValue: boolean): string;
begin
    result := 'document.getElementById("' + aID + '").checked = ' +
        lowerCase(boolToStr(aValue, true)) + ';' + LE;
end;

// mengubah isian masukan dgn kode javascript [memo]
function jsFillMemo(const aID: string; const aValue: string): string;
begin
    result := 'document.getElementById("' + aID + '").innerHTML = "' + aValue + '";' + LE;
end;

// membuat garis pemisah vertical
function spacer(const aClass: string = 'spacer'; const aID: string = ''): string;
begin
    result := '<span' + switch(aClass = '', '', ' class="' + aClass + '"') +
        switch(aID = '', '', ' id="' + aID + '"') + '> | </span>';
end;

// membuat masukan tombol
function inputButton(const aCaption: string; const action: string = '';
    const isReset: boolean = false): string;
begin
    inputIndex := inputIndex + 1;
    result := '<button' +
        ' type="' + switch(isReset, 'reset', 'submit') + '" +
        ' id="input_' + intToStr(inputIndex) + '" +
        ' name="input_' + intToStr(inputIndex) + '" +
        ' switch(action = '', '', ' formaction="' + action + '"') +
        ' value="clicked">' + aCaption +
        '</button>';
end;

// gabungkan antarmuka masukan dan pembacaan data masukan [button]
procedure webReadButton(var clicked: boolean; const aCaption: string; const newLine: boolean = false;
    const action: string = ''; const isReset: boolean = false);
var
    s: string;
begin
    s := getValue('input_' + intToStr(inputIndex + 1), allWebInput);
    clicked := (s = 'clicked'); // baca klik dari nilai

```

```

    write(inputButton(aCaption,action,isReset));
    if newLine then writeln(BR) else writeln;
end;

// cek ketersediaan data tersimpan
function hasWebData: boolean;
var
    s: string;
begin
    memRead(webData,s);
    result := (s <> '');
end;

// simpan data ke memori
procedure saveWebData;
begin
    memWrite(webData,allWebInput);
end;

// baca data dari memori
function readWebData: string;
begin
    memRead(webData,result);
end;

// mengisi form dengan data tersimpan
procedure showWebData(const isReset: boolean = false);
var
    i: integer = 0;
    d,s,web: string;

    function getInput: string;
    begin
        i += 1;
        result := getValue('input_'+intToStr(i),web);
    end;

begin
    web := readWebData;
    log.debug('#'+{$I %LINE%}+' : @showWebData saved = '+web);
    // baca dan tampilkan data masukan sesuai urutannya
    d := switch(isReset,'',htmlEncode(httpDecode(getInput)));
    s := jsFillValue('input_'+intToStr(i),d);
    d := switch(isReset,'',getInput);
    s += jsFillValue('input_'+intToStr(i),d);
    d := switch(isReset,'',getInput);
    s += jsFillValue('input_'+intToStr(i),d='true');
    d := switch(isReset,'',htmlEncode(httpDecode(getInput)));
    s += jsFillMemo('input_'+intToStr(i),d);
    writeScript(s);
end;

// hapus data tersimpan
procedure clearWebData;
begin
    memDestroy(webData);

```

```

    showWebData(true);
end;

// menulis isi laman
procedure writeContent;
var
    d: string = '';
    s: string = '';
    i: integer = -1;
    b: boolean = true;
    m: string = '';
    btnSubmit, btnClear: boolean;
begin
    // buat penyimpanan data dgn kapasitas 1 kb
    memCreate(webData,1024);

    // tampilkan antarmuka masukan
    writeln(Po,formOpen(ExtractFilename(ParamStr(0)),true));
    write(putLabel('String: '));    webReadln(s);
    write(putLabel('Number: '));    webReadln(i);
    write(putLabel('Boolean: ', '')); webReadln(b,'Yes, I agree');
    write(putLabel('Memo: '));      webReadMemo(m);
    writeln(separator);
    write(putSpace); webReadButton(btnSubmit,' SUBMIT ');
    write(spacer);   webReadButton(btnClear , ' CLEAR ');
    writeln(FORMc,Pc,HR);

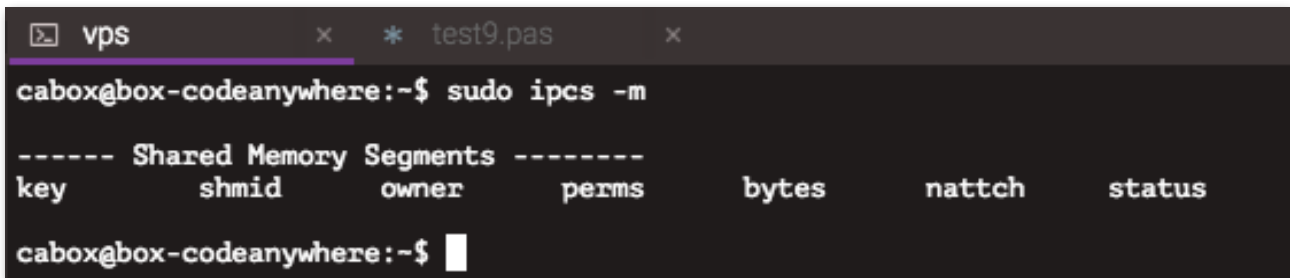
    // tampilkan masukan, jika ada
    if webHasInput then
    begin
        if btnClear then clearWebData;
        if btnSubmit then saveWebData;
        if hasWebData then d := readWebData;
        // tampilkan data tersedia
        writeln(Po,span('Available data:', 'bold'),BR);
        writeln('Input: ' ,switch(webInputs = '', '[empty]',webInputs),BR);
        writeln('Query: ' ,switch(webQueries='', '[empty]',webQueries),BR);
        writeln('Saved: ' ,switch(d='', '[empty]',d),BR);
        writeln(Pc,HR);
    end
    // tampilkan data tersimpan
    else begin
        if hasWebData then showWebData;
    end;

    log.debug('#'+{$I %LINE%}+' : [main.writeContent] done. ');
end;

(** program utama **)
begin
    writeHeader('Read Input','test.css');
    writeTitle('SAVE INPUT',3);
    allWebInput := readAllWebInput;
    writeContent;
    writeFooter;
end.

```

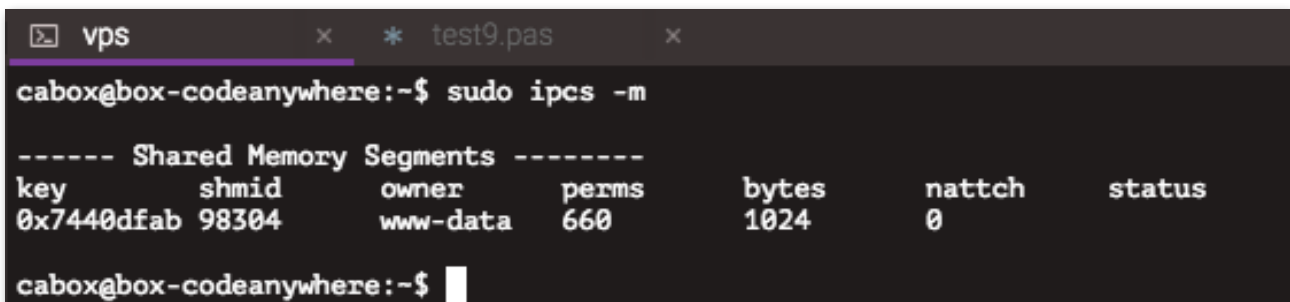

2. Simpan berkas, kompilasi program, tapi jangan dulu buka `test9.cgi` di peramban. Di terminal, ketikkan perintah `sudo ipcs -m` terlebih dahulu. Seharusnya perintah tersebut akan menunjukkan bahwa tidak ada memori bersama yang sedang digunakan, seperti pada gambar di bawah ini.



```
vps x * test9.pas x
cabox@box-codeanywhere:~$ sudo ipcs -m
----- Shared Memory Segments -----
key          shmid      owner      perms      bytes      nattch     status
cabox@box-codeanywhere:~$
```

Keluaran perintah `ipcs` saat **tak ada** memori bersama yang digunakan.

3. Sekarang baru buka `test9.cgi` di peramban. Sesuai dengan isi pada baris nomor 139, program langsung membuat atau menyambungkan ke memori bersama. Kita kembali ke terminal dan jalankan lagi perintah `ipcs` di atas untuk melihat status memori bersama. Jika semua berjalan benar, berikut tampilan keluarannya.



```
vps x * test9.pas x
cabox@box-codeanywhere:~$ sudo ipcs -m
----- Shared Memory Segments -----
key          shmid      owner      perms      bytes      nattch     status
0x7440dfab   98304      www-data   660        1024        0
cabox@box-codeanywhere:~$
```

Keluaran perintah `ipcs` saat **ada** memori bersama yang digunakan.

4. Gambar terakhir di atas menunjukkan bahwa sedang ada satu segmen memori bersama yang sedang digunakan, dengan `shmid` adalah 98304. Nilai `shmid` ini bisa berbeda-beda di tiap mesin, tergantung yang diberikan oleh sistem operasi saat program meminta alokasi memori bersama.
5. Kemudian silakan masukkan nilai bebas namun sesuai ke setiap masukan yang tampil di program `test9.cgi` tersebut, lalu klik tombol **SUBMIT**. Sesuai dengan isi baris nomor 156 yang jika tombol **SUBMIT** diklik maka program akan menyimpan data masukan dari peramban ke memori bersama yang sebelumnya dibuka.
6. Lalu tutup *tab* peramban yang sedang membuka program `test9.cgi` (lebih baik salin dulu alamat tautan program) lalu buka tab baru di peramban yang sama. Lalu buka lagi program `test9.cgi` (atau tempel alamat tautan yang sudah disalin tadi).
7. Sesuai dengan perintah program di baris nomor 167, maka program `test9.cgi` langsung menampilkan data yang telah tersimpan di memori bersama. Artinya, data terakhir yang kita masukkan sebelumnya (di langkah ke-5 di atas) akan langsung muncul kembali.
8. Silakan buka kembali program `test9.cgi` di peramban lain (yang berbeda) di komputer yang sama, atau di komputer yang berbeda, atau di ponsel pintar, dan sebagainya. Nilai yang tersimpan tetap akan muncul di mana pun program dijalankan.

9. Kemudian klik tombol **CLEAR**. Sesuai dengan isi baris nomor 155 yang jika tombol **CLEAR** diklik maka program akan menghapus memori bersama yang digunakan. Menghapus di sini tak hanya mengosongkan data di memori bersama, tapi menghapus alokasi memori bersama yang digunakan dan mengembalikannya ke sistem operasi.
10. Jika kita melihat status memori bersama setelah tombol **CLEAR** diklik maka kita akan mendapati bahwa memori bersama yang muncul tadi (dengan shmid = 98304) telah tidak ada. Tampilan perintah `ipcs` akan kembali seperti pada langkah nomor 2 di atas.

Ada dua hal baru yang diperhatikan dalam program `test9.cgi`, yaitu:

1. Pengolahan data dengan memori bersama melalui fungsi `hasWebData()`, `saveWebData()`, `writeWebData()`, `clearWebData()`, dan `showWebData()` di mana fungsi-fungsi tersebut untuk mengolah data masukan dari peramban di memori bersama.

Bagi yang telah biasa menggunakan basis data, fungsi-fungsi olah data dengan memori bersama bisa digantikan dengan fungsi-fungsi basis data yang sesuai. Silakan berlatih sendiri. Karena bersifat khusus di setiap program, fungsi-fungsi ini sebaiknya tidak dipindahkan ke *unit* `webUtils.pas`.

2. Fungsi `writeScript()` dan `jsFillValue()` yang digunakan untuk mengubah isian/nilai di masukan HTML melalui kode JavaScript. Fungsi-fungsi tersebut digunakan dalam prosedur `showWebData()` yang mengubah isian HTML sesuai data yang tersimpan (di memori bersama). Serta tambahan fungsi `inputButton()` dan `webReadButton()` untuk membaca masukan berupa tombol.

Fungsi-fungsi ini yang perlu dipindahkan ke *unit* `webUtils.pas` untuk melengkapi fitur HTML yang telah ada sehingga bisa kembali digunakan dalam program-program web selanjutnya. Tepatnya, kode program dari baris ke-9 hingga 71 di program `test9.pas`.

3. Menggunakan sesi web

Pada bab awal telah dijelaskan tentang kue HTTP dan sesi web, yaitu pemasangan kode identitas peramban berupa kue HTTP yang disesuaikan dengan data yang disimpan di sisi program web sehingga memungkinkan komunikasi HTTP yang tidak nirkeadaan. Kita baru saja belajar menyimpan data ke memori bersama, artinya kita bisa menerapkan sesi web.

Kita belajar sesi web yang sederhana dulu, yaitu menandai setiap peramban (klien web). Setiap peramban yang mengakses program web akan diberi tanda unik dengan datanya masing-masing. Misal, jika ada 5 peramban yang berbeda (atau dari mesin berbeda) maka masing-masing peramban bisa menyimpan datanya sendiri.

Untuk itu mari kita buat program baru untuk belajar penggunaan sesi web. Seperti biasa, silakan ikuti langkah-langkah berikut:

1. Buat berkas program baru bernama `test10.pas` dan salin program berikut ini (194 baris):

```
program webTest;
```

```
uses
```

```
  SysUtils, DateUtils, sharedMem, webUtils;
```

```

const
    SPACE = '&nbsp;';

var
    webData: TSharedMem;
    browserID: string = '';

// membuat guid
function generateGUID: string;
var
    guid: TGUID;
begin
    repeat sleep(1) until createGUID(guid) = 0;
    result := guidToString(guid);
end;

// menyusun kue http; harus dipanggil sebelum writeHeader()
function createCookie(const aName, aValue: string; const aDuration: integer = 0):
string;
var
    utc: TDateTime;
begin
    result := 'set-cookie: '+aName+'='+aValue+'; SameSite=Strict; HTTPOnly';
    if aDuration > 0 then
        begin
            utc := incMinute(now, GetLocalTimeOffset + aDuration);
            result += formatDateTime('; "Expires="ddd, dd mmm yyyy hh:nn:ss "GMT"', utc);
        end;
end;

// cek ketersediaan data tersimpan
function hasWebData: boolean;
var
    m, d: string;
begin
    memRead(webData, m);
    d := getValue(browserID, m, '|', ':');
    result := (d <> '');
end;

// simpan data ke memori
procedure saveWebData;
var
    m, d, t: string;
begin
    memRead(webData, m);
    // format sesi <id>:<data> dgn pemisah |
    d := getValue(browserID, m, '|', ':');
    // hapus data sesi yg sudah tersimpan
    m := stringReplace(m, browserID+':'+d+'|', '', []);
    // catat waktu simpan
    t := '&last='+floatToStr(now);
    // ganti data sesi dgn yg baru
    m += browserID+':'+allWebInput+t+'|';
    memWrite(webData, m);
    log.debug('#'+{$I %LINE%}+' : @saveWebData saved = '+m);

```

```

end;

// baca data dari memori
function readWebData: string;
var
  m: string;
begin
  memRead(webData,m);
  result := getValue(browserID,m,'|',':');
end;

// mengisi form dengan data tersimpan
procedure showWebData(const isReset: boolean = false);
var
  i: integer = 0;
  d,s,web: string;

  function getInput: string;
  begin
    i += 1;
    result := getValue('input_'+intToStr(i),web);
  end;

begin
  web := readWebData;
  // baca dan tampilkan data masukan sesuai urutannya
  d := switch(isReset,'',htmlEncode(httpDecode(getInput)));
  s := jsFillValue('input_'+intToStr(i),d);
  d := switch(isReset,'',getInput);
  s += jsFillValue('input_'+intToStr(i),d);
  d := switch(isReset,'',getInput);
  s += jsFillValue('input_'+intToStr(i),d='true');
  d := switch(isReset,'',htmlEncode(httpDecode(getInput)));
  s += jsFillMemo('input_'+intToStr(i),d);
  writeScript(s);
end;

// hapus data tersimpan
procedure deleteWebData;
var
  m,d: string;
begin
  memRead(webData,m);
  d := getValue(browserID,m,'|',':');
  m := stringReplace(m,browserID+':'+d+'|','',[ ]);
  memWrite(webData,m);
  showWebData(true);
end;

// kosongkan memori bersama
procedure clearWebData;
begin
  memDestroy(webData);
  showWebData(true);
end;

```

```

// baca kue http
procedure setupCookies;
begin
    webCookies := readWebCookie;
    if webCookies = '' then
    begin
        // simpan kue baru
        browserID := generateGUID;
        writeln(createCookie('browser_id',browserID));
    end
    else
        // baca kue tersimpan
        browserID := getValue('browser_id',webCookies);
end;

// menulis isi laman
procedure writeContent;
var
    d: string = '';
    s: string = '';
    i: integer = -1;
    b: boolean = true;
    m: string = '';
    btnSubmit, btnDelete, btnClear: boolean;
begin
    // buat penyimpanan data dgn kapasitas 1 kb
    memCreate(webData,1024);

    // tampilkan antarmuka masukan
    writeln(Po,formOpen(ExtractFilename(ParamStr(0)),true));
    write(putLabel('String: '));      webReadln(s);
    write(putLabel('Number: '));      webReadln(i);
    write(putLabel('Boolean: ', '')); webReadln(b,'Yes, I agree');
    write(putLabel('Memo: '));        webReadMemo(m);
    writeln(separator);
    write(putSpace); webReadButton(btnSubmit,' SUBMIT ');
    write(spacer);   webReadButton(btnDelete,' DELETE ');
    write(SPACE);    webReadButton(btnClear , ' CLEAR ');
    writeln(FORMc,Pc,HR);

    // tampilkan masukan, jika ada
    if webHasInput then
    begin
        if btnSubmit then saveWebData;
        if btnDelete then deleteWebData;
        if btnClear then clearWebData;
        memRead(webData,d);

        // tampilkan data tersedia
        writeln(Po,span('Available data:', 'bold'),BR);
        writeln('Input: ',switch(webInputs='', '[empty]',webInputs),BR);
        writeln('Query: ',switch(webQueries='', '[empty]',webQueries),BR);
        writeln('Cookie: ',switch(webCookies='', '[empty]',webCookies),BR);
        writeln('Saved: ',BR,switch(d='', '[empty]'+BR, stringReplace(d, '|',BR,[rfReplaceAll])));

        // info waktu perbaruan

```

```

    d := getValue('last',readWebData);
    if d <> '' then d := formatDateTime('dd-mm-yyyy hh:nn:ss',strToFloat(d));
    writeln('Last updated: ',switch(d='', '- ',d));
    writeln(Pc,HR);
end
// tampilkan data tersimpan
else begin
    if hasWebData then showWebData;
end;

log.debug('#'+{$I %LINE%}+' : [main.writeContent] done. ');
end;

(***) program utama (***)
begin
    setupCookies;
    writeHeader('Read Input','test.css');
    writeTitle('DATA & COOKIE',3);
    allWebInput := readAllWebInput;
    writeContent;
    writeFooter;
end.

```

2. Simpan berkas, kompilasi program, tapi jangan dulu buka `test10.cgi` di peramban.
3. Kemudian silakan masukkan nilai bebas namun sesuai ke setiap masukan yang tampil di program `test10.cgi` tersebut, lalu klik tombol **SUBMIT**. Seperti program yang lalu, ketika tombol **SUBMIT** diklik maka program web akan menyimpan data yang diberikan. Namun program web kali ini juga menampilkan informasi sesi web yang pernah dibuat yang unik berdasarkan peramban yang mengakses. Jika ada, misalnya 3, peramban yang berbeda atau dari mesin yang berbeda maka akan tampil 3 informasi sesi web. Contohnya seperti pada gambar berikut.

Available data:

Input: input_1=abc&input_2=123&input_3=true&input_4=test+lgi&input_5=clicked

Query: [empty]

Cookie: browser_id={4F709763-B446-4D8C-8B88-244780D7AFBF}

Saved:

{6F5C0FC1-E773-4B43-AAB2-7592096CF094}:input_1=uiop&input_2=890&input_3=true&input_4=another+test&input_5=clicked&last=42848.026179213

{83DCECA2-61DC-4DA2-BE89-A608C58823E6}:input_1=string&input_2=12345&input_3=true&input_4=this+is+another+test&input_5=clicked&last=42848.0285434144

{4F709763-B446-4D8C-8B88-244780D7AFBF}:input_1=abc&input_2=123&input_3=true&input_4=test+lgi&input_5=clicked&last=42848.0966385069

Last updated: 23-04-2017 02:19:09

This page is served in 1 ms.

Contoh tampilan program saat ada tiga peramban berbeda yang mengakses. Nilai bisa berbeda tergantung kode GUID yang acak dan masukan pengguna.

4. Untuk membuktikannya, coba buka program `test10.cgi` dari peramban berbeda, atau dari komputer yang berbeda, atau dari peramban di ponsel pintar juga boleh. Pada setiap peramban, isikan data masukan yang berbeda (jangan disamakan dengan data masukan di peramban lain). Ini untuk menguji sesi web.

5. Kemudian buka *tab* baru di setiap peramban, lalu buka lagi program `test10.cgi` di *tab* baru tersebut. *Tab* yang sebelumnya boleh ditutup dulu, asal jangan menutup jendela utama peramban, karena program ini menggunakan jenis kue HTTP sementara.
6. Jika program `test10.cgi` ini bekerja benar maka data yang ditampilkan di *tab* baru akan sama dengan data yang diberikan sebelumnya pada masing-masing peramban. Artinya sesi web bekerja dengan baik.
7. Selanjutnya, klik tombol **DELETE** di salah satu peramban. Sesuai dengan kode program di baris nomor 161, maka program akan menghapus data sesi web dari peramban yang bersangkutan dan mengosongkan isian data.

Perlu diperhatikan bahwa yang dihapus adalah data sesi web di program web, bukan kue HTTP di peramban. Artinya, data kue HTTP peramban masih ada dan dikirim oleh peramban, namun padanan sesi web untuk kue HTTP tersebut di program web telah dihapus. Ini yang disebut dengan sesi web yang rusak, dengan kerusakan berada di sisi program web (sisi belakang).
8. Kemudian mari beralih ke peramban yang lain atau peramban di mesin yang berbeda. Tutup aplikasi peramban tersebut. Tutup aplikasinya, bukan sekadar menutup *tab*. Karena program kali ini menggunakan kue HTTP sementara maka menutup aplikasi peramban akan menghapus kue HTTP sementara yang sebelumnya digunakan.
9. Lalu buka kembali peramban lain atau peramban di komputer yang berbeda. Kemudian buka program web `test10.cgi` di peramban tersebut. Karena peramban sudah pernah ditutup maka kue HTTP sementara tadi sudah tidak berlaku lagi sehingga sesi web tak bisa terjalin. Ini yang disebut sesi web yang rusak, dengan kerusakan di sisi peramban (sisi depan).
10. Selanjutnya, klik tombol **CLEAR** di salah satu peramban. Sebagaimana pada program sebelumnya, tombol ini berfungsi untuk menghapus memori bersama yang digunakan, yang perintahnya bisa dilihat pada baris ke-162 dan prosedur `clearWebData()`. Karena yang dihapus adalah memori bersama maka **seluruh** data sesi web yang tersimpan akan hilang. Seperti yang telah dijelaskan di program sebelumnya, untuk mengetahui status memori bersama bisa menggunakan perintah `sudo ipcs -m` di terminal.

Program `test10.cgi` ini tak banyak berbeda dengan program sebelumnya. Beberapa hal baru yang penting antara lain; pertama adalah penggunaan UUID (*universally unique identifier*) atau disebut juga GUID (*globally unique identifier*) melalui *class* `TGUID` yang disediakan Free Pascal. GUID adalah suatu metode pembuatan identitas unik dengan nomor acak. Konon GUID menjamin keunikan hingga $2,7 \times 10^{18}$ sehingga kemungkinan muncul nomor kembar amat sangat kecil. Penggunaannya mudah sekali, silakan pelajari isi fungsi `generateGUID()` pada baris nomor 14.

Yang kedua adalah pengolahan kue HTTP. Pembuatan kue HTTP di program ini dilakukan oleh fungsi `createCookie()` yang dimulai dari baris program nomor 23. Secara bawaan, fungsi ini akan membuat kue HTTP sementara. Jika ingin membuat kue HTTP permanen, isi parameter `aDuration` dengan angka lebih dari 0 yang menyatakan lama masa berlaku kue HTTP dalam menit.

Selanjutnya adalah proses pengenalan identitas peramban untuk sesi web. Hal ini dilakukan oleh prosedur `setupCookie()` yang dimulai dari baris program nomor 118. Nilai kue HTTP dibaca dari peubah sistem seperti yang sudah kita bahas di pembacaan data masukan web. Jika peramban tidak membawa kue HTTP maka program akan memberikan kue HTTP sebagai penanda keunikan peramban. Nilai unik berupa GUID dari fungsi `generateGUID()` disimpan dalam peubah `browserID`. Jika peramban membawa kue HTTP maka program akan mengambil nilainya ke peubah `browserID`. Nilai peubah `browserID` ini yang menjadi acuan pembacaan data masukan yang tersimpan di memori bersama. Dengan pencocokan peubah `browserID` dari kue HTTP yang dikirim peramban dengan data yang tersimpan di memori bersama, terjalin yang disebut dengan sesi web.

Untuk lebih memahami cara kerja program, silakan bereksperimen lebih lanjut dengan program menggunakan peramban yang berbeda. Untuk simulasi peramban yang banyak, kita bisa menggunakan fitur mode *private* atau *incognito* yang disediakan peramban. Mode tersebut memungkinkan peramban untuk memiliki berkas penyimpanan data yang berbeda (termasuk data kue HTTP) dengan peramban yang dibuka normal atau dengan peramban mode *private* lainnya.

• • •

IV. Tanya Jawab & Diskusi

Secara umum tidak ada tanya jawab atau diskusi penting setelah kulgram karena peserta sibuk bereksperimen dengan kode-kode program yang disertakan.

Satu catatan penting adalah adanya beberapa kesalahan karakter petik ganda dan petik tunggal di artikel akibat proses koreksi-otomatis (*auto-correction*) aplikasi penyunting artikel ini. Karakter petik ganda yang dalam kode program seharusnya `"..."` diubah menjadi `"..."` oleh penyunting. Demikian juga pada tanda petik tunggal yang dalam kode program seharusnya `'...'` diubah menjadi `'...'` oleh penyunting. Karena itu, jika saat Anda mengkompilasi program muncul pesan kesalahan karakter tak dikenali, kemungkinan besar ada kesalahan karakter petik ganda atau tunggal tersebut. Untuk mengatasinya, cukup ketik ulang karakter petik yang salah pada posisi yang ditunjukkan oleh pesan kesalahan.

Selamat bereksperimen!

...

V. Penutup

Saat ini, aplikasi web dan aplikasi perangkat bergerak (*mobile applications*) sedang menjadi tren dunia perangkat lunak. Sampai-sampai ada ujaran, “*if you don’t exist on the internet, you don’t exist at all*” yang terjemah bebasnya kira-kira “jika kamu tak ada di internet, kamu tidak dianggap ada”. Semuanya sekarang saling terhubung di internet, tak hanya mesin dengan mesin, bahkan sudah pada manusia dengan manusia, berkat adanya media sosial di internet. Kemampuan pemrograman web bagi pemrogram masa kini sudah jadi kemampuan yang wajib dimiliki.

Apa yang kita pelajari di kulgram atau buku-el ini adalah hal-hal yang sangat mendasar dalam pemrograman web dengan bahasa Pascal menggunakan Free Pascal. Bisa jadi apa yang telah kita pelajari ini tak tampak jika kita menggunakan berbagai kerangka-kerja (*framework*) atau pustaka (*library*) web yang melakukan semua hal mendasar ini secara otomatis. Namun pengetahuan dasar seperti ini tetap penting untuk kita pahami agar kita sebagai pemrogram paham betul bagaimana teknologi web bekerja.

Saya berharap materi yang mendasar dan singkat ini bisa menjadi pembuka bagi para pemrogram yang baru saja/akan belajar pemrograman web. Juga semoga bisa menjadi pemancing rasa ingin tahu bagi yang sudah bi(a)sa membuat program web agar lebih mempelajari rincian teknologi pemrograman web. Untuk kemudian tertarik membuat kerangka-kerja dan pustaka web sendiri, atau setidaknya paham bagaimana kerja kerangka-kerja dan pustaka web buatan orang lain. Dengan begitu, pemrogram lokal tak hanya mampu jadi pengguna karya orang lain, tetapi juga bisa menjadi pembuat karya yang akan digunakan orang lain.

...

Berkas-berkas kulgram ini tersedia di:

- kode sumber program: <https://github.com/pakLebah/dasarweb/>
- buku elektronik PDF: https://pak.lebah.web.id/paklebah_kulgram2.pdf