۱) مقدمه

برای سالیان متوالی سیستم پست الکترونیکی عمومی ترین ابزار کاربردی بر روی شبکه اینترنت به شمار می رفت تا آنکه در اوائل دهه نود میلادی تور جهان گستر[۱] متولد شد. گزافه نیست اگر تور جهان گستر (یا اختصاراً وب) را وسیعترین و پررونق‌ترین ابزار روی شبکه اینترنت تلقّی کنیم؛ بگونه‌ای که در نظر کاربران معمولی ، وب و اینترنت مفهومی معادل دارد. امروزه صحبت از تور جهان گستر (وب) ، صفحهٔ وب[۲] ، ابرمتن[۳] ، آدرسهای حوزه و پروتکل انتقال ابرمتن[٤] بر سر هر زبانی شنیده می‌شود و بقدری تب آن همه جا را فراگرفته که گاه مهندسین شبکه نیز مفاهیم این اصطلاحات را رها کرده و به کاربردهای آن دل بسته‌اند.

در این فصل پس از معرفی ساده و مفهومی هر یک از اصطلاحات فوق به سمت تعریف پروتکلهای حاکم بر تور جهان‌گستر پیش خواهیم رفت.

۱-۱) مفهوم تور جهان گستر

تور جهان‌گستر یا وب یک روش معماری یا بعبارتی یک نظام برای ذخیره‌سازی و دسترسی به مستندات به هم پیوندخورده‌ای است که روی هزاران هزار ماشین در کلّ جهان پراکنده و توزیع شده اند. هر یک از این مستندات پیوندخورده که شامل متن ، صدا و تصاویر گرافیکی و تصاویر متحرک می‌شود ، می‌تواند به یک سند دیگر در محلی متفاوت در جهان اشاره نماید. این روش برای دسترسی به اطلاعات توزیع شده در عرض چند سال چنان جاذبه‌ای ایجاد کرد که کل دنیا تحت تاثیر آن قرار گرفت ، بگونه‌ای که مردم عادی را نیز واداشت که به این پدیده نوظهور بعنوان ابزاری برای زندگی راحتتر نگاه کنند.

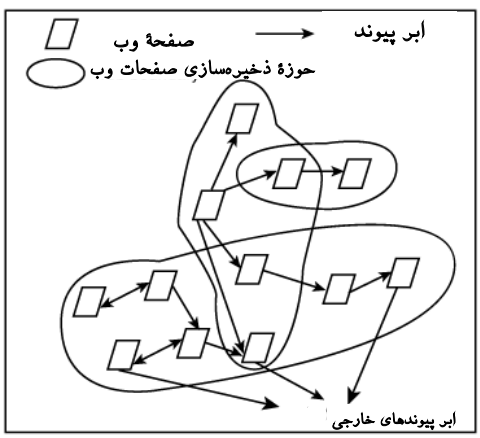
بزرگترین حسن وب، سادگی استفاده از آن است؛ کاربر با وارد کردن آدرس یک سایت که یک صفحه وب در آنجا ذخیره شده ، آنرا بر روی کامپیوترِ خود منتقل می‌کند ، آنرا نگاه و مطالعه کرده و اگر تمایل به دریافت اطلاعات بیشتری در مورد آیتمهائی که پررنگ[۵] هستند ، داشته باشد با ماوس خود روی آن کلیک می‌کند.

---

۱ Web
۲ Web page
۳ Hypertext
٤ Hype Text Transfer Protocol/URL : HTTP
۵ Highlight

(آیتمهای پر رنگ نقطهٔ پیوندِ[۱] صفحهٔ فعلی با یک صفحهٔ وبِ دیگر به حساب می‌آیند) صفحهٔ جدید هم مثل صفحهٔ قبلی روی کامپیوتر او بارَ شده و همین روند ادامه می‌یابد. ممکن است کاربر صفحه‌ای را از یک سایت در آمریکا بار کرده باشد. پس از مطالعهٔ آن به موضوعی علاقمند می‌شود که بصورت پر رنگ علامتگذاری شده است؛ روی آن کلیک می‌کند ، بدون آنکه بداند صفحه وب مورد نظر او روی سایتی مثلاً در اروپا قرار دارد. در حقیقت در صفحهٔ اوّلی که او نگاه می‌کند در ذیل عنوان پر رنگ یک آدرس دیگر (که از این بعد آنرا URL می‌نامیم) ، وجود داشته که ارتباط بین دو صفحه را برقرار کرده است. در شکل (۱-۱۰) شمای کلی ارتباطات بین صفحات وب به تصویر کشیده شده است.

حال شما تصور کنید اکنون که (ابتدای سال ۱۳۸۰) طبق آمار غیر رسمی حدود یک و نیم میلیارد صفحه وب در کلّ دنیا ایجاد و ذخیره شده و بین هر یک از آنها دهها پیوند برقرار است ، آیا این صفحات و ارتباطات آنها بصورت یک تار عنکبوت که روی کلّ جهان کشیده شده است مجسّم نمی‌شود؟ اصطلاح وِب (تار عنکبوت) از همین تصور نشأت گرفته است. به هر حال ما بدنبال وجه تسمیه و جنبه‌های ادبی قضیه نیستیم و باید مفاهیم فنی آنرا بررسی نمائیم.



شکل (۱-۱۰) شمای کلی ارتباطات بین صفحات وب

[۱] Link

شاید وب جالبترین جنبه استفاده از مفهوم برنامه‌های سرویس‌دهنده / مشتری در شبکه اینترنت باشد؛ بدینگونه که برنامهٔ سمت مشتری تقاضایی را برای دریافت یک صفحه وب یا یک فایل ، به سمت سرویس‌دهنده ارسال می‌کند. برنامهٔ سرویس‌دهنده در صورت امکان این تقاضا را اجابت کرده و داده‌های لازم را ارسال می‌نماید.

بد نیست قبل از بررسی برنامهٔ سرویس دهندهٔ وب[1] و برنامهٔ مشتری (مرورگر) تعریف URL را ارائه نمائیم.

۲-۱) مفهوم URL[2]

اشاره کردیم که یک صفحه وب می‌تواند شامل یک پیوند به صفحه وب دیگر درهر نقطه از دنیا باشد. هر پیوند در حقیقت آدرس دقیق یک فایل یا صفحهٔ وب محسوب می‌شود.

با توجه به ناهمگون بودن سیستمهای عامل و کامپیوترها در دنیا ، بعنوان یک نیاز بنیادی باید بتوان فایلها وپروسه‌ها را از لحاظ سبک نامگذاری و محل ذخیرهٔ آنها بر روی یک ماشین، هماهنگ و استاندارد کرد. یعنی باید یک روش آدرس دهی برای فایلها انتخاب شود به گونه ای که بتواند به سه سوال زیر برای هر فایلی در دنیا پاسخ بدهد:

- نام فایل چیست؟
- محل دقیق ذخیره شده فایل کجاست؟ (یعنی روی چه ماشینی و چه زیر شاخه‌ای قرار دارد؟)
- به چه روشی باید به فایل دسترسی داشت و طبق چه قاعده‌ای می‌توان آن فایل را انتقال داد؟

یک روش آدرس‌دهیِ استاندارد باید بتواند هر فایلی را بطور منحصر به فرد و یکتا در دنیا مشخص کند بنحوی که هیچ ابهامی در آن وجود نداشته باشد. URL روشی برای آدرس دهی منابع و فایلها در دنیاست که به هر سه سوال فوق بدون هیچ ابهامی پاسخ می‌دهد. بنابراین در ذهن خود URL را بعنوان یک آدرس استاندارد یا یک سَبکِ آدرس‌دهی تجسّم کنید.

---

[1] Web Server
[2] Uniform Resource Locator

آدرس URL که امروزه شما حتی روی پوسته یک بیسکوئیت آنرا می‌بینید شامل سه قسمت اساسی است :

- قسمت پروتکل که به آن قاعده انتقال[۱] هم گفته می‌شود.
- نام ماشینی که فایل روی آن قرار دارد. (نام حوزهٔ ماشین یا آدرس IP آن)
- شاخه و نام فایل

بعنوان مثال به آدرس URL زیر دقت کنید:

http://www.ibm.com/researches/piii/paper.htm

آدرس فوق از سه قسمت تشکیل شده است:

- فیلد پروتکل که در این مثال **http** است. این فیلد به برنامهٔ سمت مشتری که "مرورگر[۲]" نام دارد تفهیم می‌کند که برای بارکردن و انتقال فایل باید از کدام پروتکل و مجموعهٔ فرامین استفاده نماید. درحقیقت این فیلد زبان صحبت کردن مرورگر با سرویس‌دهنده را تعیین می‌کند. فیلد پروتکل با علامت //: از بقیه آدرس جدا می‌شود.
- نام حوزهٔ ماشینی که فایل مربوطه روی آن ذخیره شده است. در مثال فوق نام حوزهٔ ماشین www.ibm.com است. نام حوزه بین **//:** تا اوّلین **/** بعدی قرار می‌گیرد.
- نام شاخه و نام فایل: در این قسمت که دقیقاً بعد از نام حوزه شروع می‌شود و تا انتها ادامه می‌یابد ، نامِ شاخه ای که فایل درون آن قرار گرفته و همچنین نام فایل درج می‌شود. در مثال فوق نام فایل paper.htm است که بر روی شاخه /researches/piii/ قرار دارد.

دقت کنید که بر خلاف سیستم عامل DOS و MS-Windows شاخه‌ها با علامت \ از هم جدا نمی‌شوند بلکه مشابه سیستم عامل یونیکس با علامت / از یکدیگر تفکیک می‌شوند. بنابراین بصورت عمومی آدرس URL بصورت زیر تجزیه می‌شود:

نام فایل/.../.../نام شاخه/ نام حوزهٔ ماشین // : نام پروتکل

دو نکته بسیار مهم در مورد آدرس URL وجود دارد:

---

[۱] Transfer Protocol / Schema
[۲] Browser

اوّل آنکه اگر هنگام وارد کردن آدرس نام فایل ذکر نشود ، سرویس‌دهنده بصورت پیش‌فرض نام فایل مورد نظر را index.htm در نظر می‌گیرد. یعنی دو آدرس زیر دقیقاً معادلند:

http://www.sony.com/

http://www.sony.com/index.htm

با چنین فرضی آدرسها کوتاه می‌شوند. معمولاً به اولین صفحه ای که کاربر از یک سایت می‌بیند و با نام index.htm ذخیره شده است ، **"صفحه خانگی"**[1] گفته می‌شود. طراح صفحهٔ وب ، صفحه اصلی یک سایت را بگونه‌ای طراحی می‌کند که برای کاربر نقش یک کاتالوگِ فهرست‌دار را بازی کند؛ نام این فایل را index.htm می‌گذارد تا کاربر مجبور نباشد نام دقیق آنرا در آدرس URL وارد نماید و آدرس کوتاه باشد. پس از انجام مراحل بار شدن صفحهٔ خانگی با نام index.htm ، کاربر این صفحه را مطالعه کرده روی عناوین دلخواهش کلیک می‌کند. در بطن هر عنوان پررنگ می‌تواند یک URL مفصل و بزرگ باشد که اگر چه کاربر آنرا نمی‌بیند ولی وقتی روی آن کلیک می‌کند یک صفحهٔ دیگر بار می‌شود؛ بنابراین کاربر مجبور نیست تمام آدرسهای URL را بداند و فقط دانستن آدرس صفحهٔ خانگی برای دسترسی به تمام مستندات آن سایت کفایت می‌کند. این آدرسها توسط طراح صفحهٔ وب در ذیل هر عنوان ، تنظیم شده و یک پیوند به صفحهٔ دیگر محسوب می‌شود.

نکته دوم آنست که نام حوزه در آدرس URL می‌تواند با آدرس IP جایگزین شود؛ یعنی اگر آدرس 203.141.207.14 معادل با نام حوزهٔ www.sony.com باشد ، دو آدرس URL زیر معادلند:

http://www.sony.com/products.htm

http://203.141.207.14/products.htm

استفاده از آدرسهای IP بجای نام حوزه مرسوم نیست زیرا به راحتی به خاطر سپرده نمی‌شود ولی در مجموع امکان‌پذیر است و چون نیازی به ترجمهٔ نام حوزه وجود ندارد ، طبیعتاً اندکی سریعتر است.

توصیه مؤکّد آن است که سعی کنید تمامی حروف آدرس URL را حروف کوچک وارد کنید مگرآنکه صریحاً بصورت حروف بزرگ معرفی شده باشند.

---

[1] Home Page

دقت کنید که در آدرس‌دهی به سبک URL ، به غیر از نام و محل دقیق فایل ، روش دریافت فایل ذخیره شده نیز به مرورگر تفهیم می‌شود. بعنوان مثال اگر آدرس زیر را داشته باشید فایلی را بر روی کامپیوتر خودتان مشخص می‌کند:

file://utils/av/readme.htm

برخی از پروتکلها که در آدرس URL استفاده می‌شود در جدول (۲-۱۰) معرفی شده است که در آینده مهمترین آنها را تشریح خواهیم کرد.

نام پروتکل	مورد استفاده	مثال
http	انتقال صفحات ابرمتن	http://www.ibm.com/
ftp	انتقال فایل	ftp://ftp.cs.vu.nl.minx/readm
file	فایلهای محلی	file://user/prog.c
news	گروههای خبری	news://comp.os.minix
gopher	گوفر	gopher://gopher.tc.mn.edu
telnet	تِل نت	telenet://www.w3.org:80

جدول (۲-۱۰) نام پروتکلهای استاندارد در آدرس URL

۲) مقدمه ای بر سیستم وب

کلاً از دیدگاه مسائل فنی ، سیستمِ وِب در دو بخش سازماندهی می‌شود:

- برنامهٔ سمت سرویس‌دهندهٔ وب و برنامهٔ سمت مشتری وب[۱]
- پایگاه اطلاعاتیِ توزیع شده از صفحات ابرمتن و فایلهای داده مثل صدا ، تصویر و ...

صفحه وب چیزی نیست مگر یک فایل متنی بسیار ساده که با زبان علامت‌گذاری HTML تدوین می‌شوند و در بخشهای آتی روش ایجاد آنرا بوسیله یک ویرایشگر متنی (مثل Notepad) توضیح خواهیم داد.

[۱] Web Server/Web Browser

کاری که "مرورگر" بعنوان یک سرویس گیرندهٔ وب انجام می‌دهد آنست که تقاضای دریافت یکی از این صفحات یا فایلها را در قالب قراردادی استاندارد (به نام پروتکل HTTP) به سمت سرویس‌دهنده ارسال می‌کند. در سمت مقابل سرویس‌دهندهٔ وب این تقاضا را پردازش کرده و در صورت امکان ، فایل مورد نظر را برای مرورگر ارسال می‌کند. مرورگر پس از دریافتِ فایل ابرمتنی ، آنرا تفسیر کرده و بصورت صفحه آرایی شده روی خروجی نشان می‌دهد.

اگر فایل ابرمتنی در جائی به فایل صدا یا تصویر پیوند خورده باشد ، آنها نیز توسط مرورگر تقاضا شده و پس از دریافت در جای خود قرار می‌گیرند.

بنابراین سرویس‌دهندهٔ وب را باید یک برنامهٔ سوکت در نظر گرفت که فرامین مشتری‌ها را دریافت ، پردازش و در صورت امکان اجرا می‌کند. برنامه سمت مشتری نیز برنامه سوکتی است که تقاضاها را در قالب فرامین استاندارد ، برای سرویس‌دهندهٔ وب ارسال می‌کند؛ در ضمن وظیفه تفسیر و نمایش داده‌های دریافتی را نیز بر عهده دارد.

در ذهن خود دو مفهوم کاملاً مجزای زیر را از هم تفکیک کنید:

- **"پروتکل انتقال صفحات ابرمتن[۱]"** : این پروتکل زبان یا قراردادی برای صحبت کردن مشتری با سرویس‌دهنده وب (HTTP) است.
- **"زبان نشانه گذاری صفحات ابرمتن[۲]"** زبانی برای قالب‌بندی و صفحه‌آرائی اطلاعات متنی (HTML) است.

ممکن است خواننده نکته گیر به این قضیه اعتراض کند که سرویس دهندهٔ وب را با سرویس‌دهنده HTTP همسان گرفتیم در صورتیکه سرویس‌دهنده وب فراتر از سرویس دهندهٔ HTTP است. برای توجیه قضیه خاطر نشان می‌کنیم که فعلاً کاربردی‌ترین بخش از سرویس دهندهٔ وب همان سرویس دهندهٔ HTTP است.

معرفیِ زبان نشانه گذاری صفحات ابرمتنی (HTML) را به بخشهای بعد موکول کرده و فعلاً برنامه‌های سمت سرویس‌دهنده و سمت مشتری وب (مرورگر) را مورد بررسی قرار می‌دهیم تا پیوستگی بحث ما با موضوع برنامه نویسی سوکت از دست نرود.

---

[۱] Hyper Text Transfer Protocol
[۲] Hyper Text Markup Language

۱-۲) برنامه سمت سرویس‌دهنده وب

در سمت سرویس دهندهٔ وب ، پروسه‌ای وجود دارد که دائماً به پورت شماره 80 گوش می‌دهد و منتظر تقاضای برقراری ارتباط توسط مشتریان (برنامهٔ مرورگر) می‌باشد. دقت کنید که این برنامه از سوکتهای نوع استریم استفاده می‌کند و ارتباط از نوع TCP است؛ فرامین و داده‌هایی که بین سرویس‌دهنده و مرورگر مبادله می‌شوند تماماً متنی هستند.

بعد از آنکه ارتباط TCP بین برنامهٔ سرویس‌دهنده و برنامه مشتری برقرار شد برنامه مشتری حق دارد یک تقاضا بفرستد و این تقاضا باید در قالب استاندارد HTTP باشد. سرویس‌دهنده این تقاضا را دریافت و پردازش می‌کند و در صورت امکان این آن را اجرا می‌کند. مراحل بار شدن یک صفحه وب در قالب یک مثال ارائه می‌شود. به روند زیر توجه نمایید:

فرض کنید کاربر، URL زیر را در خط آدرس مرورگر خود (مثلاً در برنامه IE5.0 یا Netscape) وارد کرده و کلید ↵ را فشار داده باشد:

http://www.w3.org/hypertext/www/theproject.html

مرورگر با تحلیل آدرس URL متوجه می‌شود که باید تقاضای فایلی را طبق پروتکل HTTP به سمت سرویس‌دهنده بفرستد. مراحلی که اتفاق می‌افتد به شرح زیر خواهد بود :

الف) مرورگر آدرس URL را تحلیل کرده و قسمتهای پروتکل، آدرس نام حوزه ، شاخه و نام فایل را از آن استخراج می‌کند.

ب) مرورگر تقاضای ترجمهٔ آدرس نام حوزه را به DNS ارسال می‌نماید تا آدرس IP ماشین سرویس‌دهنده به دست آید. دراین مثال مرورگر تقاضای ترجمه نام www.w3.org را به DNS ارسال می‌کند. (معادل فراخوانی تابع gethostbyname() در برنامه نویسی سوکت)

ج) DNS در پاسخ ، آدرس IP معادل با نام حوزه را بر می‌گرداند. فرض کنید در این مثال DNS آدرس IP را 18.23.0.23 برگردانده است.

د) مرورگر یک ارتباط TCP با آدرس 18.23.0.3 و پورت 80 برقرار می‌کند. (معادل فراخوانی تابع connect() در برنامه نویسی سوکت)

ه) پس از برقراری ارتباط، رشته کاراکتری زیر که مجموعاً ۳۵ بایت است به سمت سرویس‌دهنده ارسال می‌شود:

"**GET /hypertex/www/theproject.html**"

GET یکی از فرامین استاندارد HTTP است که در ادامه توضیح خواهیم داد.

و) سرویس‌دهنده این رشته را دریافت و پس از پردازش آن ، فایل theproject.html را از شاخهٔ /hypertext/www/ استخراج کرده و برای مرورگر ارسال می‌نماید.

ز) مرورگر فایل را دریافت کرده و پس از خاتمهٔ دریافت ارتباط TCP را قطع می‌کند.

ح) مرورگر فایل ابرمتنی را تفسیر کرده و آنرا روی خروجی نمایش می‌دهد.

ط) اگر فایل ابرمتنی در جائی دارای تصویر یا صدا باشد به ازای تک‌تک آنها مراحل الف تا ح را تکرار نموده و آنها را به ترتیب دریافت می‌کند. دقّت کنید که درون فایلهای ابرمتنی داده‌های فایلهای صدا یا تصویر وجود ندارد بلکه فقط نام و محل قرار گرفتن فایل تصویر یا صدا درون آن درج شده است. (این موضوع بر خلاف سیستم پست الکترونیکی است که داده‌های صدا و تصویر در ادامهٔ متن نامه قرار می‌گیرد.)

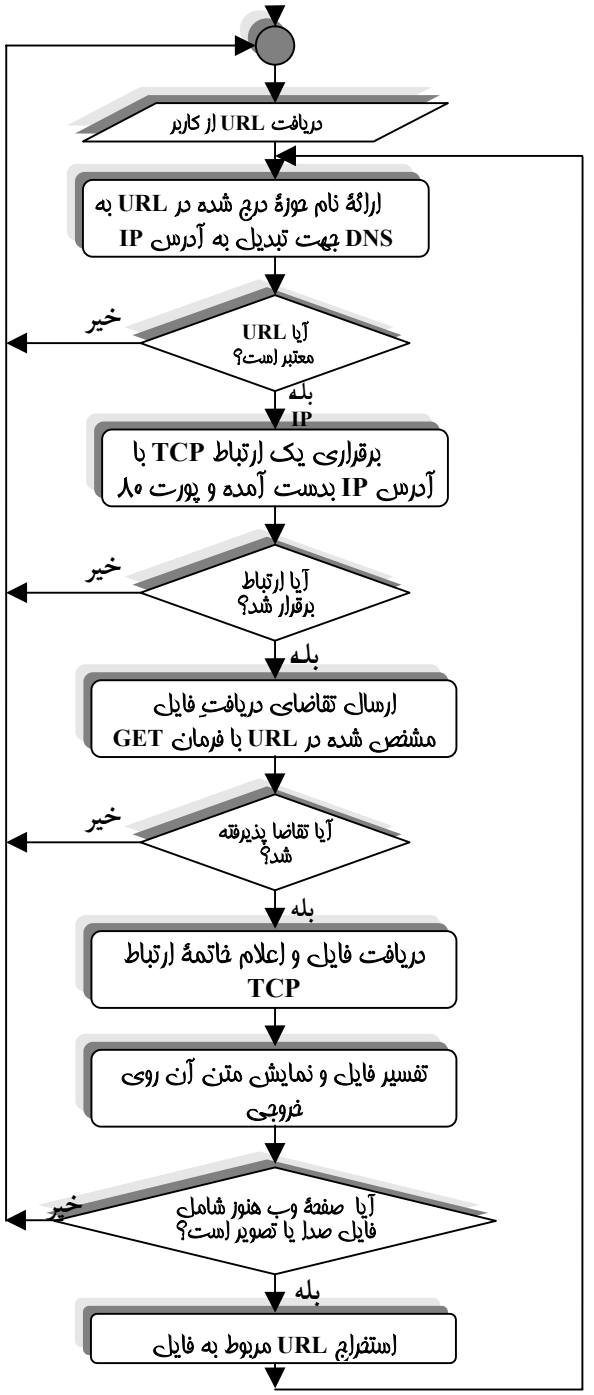
در شکل (۳-۱۰) فلوچارتِ عملیاتی که مرورگر برای دریافت یک صفحهٔ وب و ملحقات آن انجام می‌دهد ، به تصویر کشیده شده است. با تحلیل الگوریتم فوق ، روند عملیات برنامهٔ سمت سرویس‌دهندهٔ HTTP مشخص خواهد شد.

سرویس‌دهنده بایستی فرامین دریافتی که بصورت متنی و بر طبق پروتکل HTTP هستند را دریافت ، پردازش ، احراز هویت و اجرا نماید. در ادامه ، پروتکل انتقال صفحات ابرمتن را بررسی می‌نمائیم.

۳) پروتکل انتقال ابرمتن : HTTP

پروتکل انتقال ابرمتن مجموعه ای از فرامین استانداردی است که از سمت مشتری به سمت سرویس‌دهنده ارسال می‌شود. در حقیقت این پروتکل طریقهٔ صحبت کردن سرویس‌دهنده و مشتری را تبیین کرده است.

بدون مقدمه سراغ مجموعهٔ فرامین در جدول (٤-۱۰) می‌رویم؛ این فرامین در RFC-1945 **"متود"** نامیده شده است.

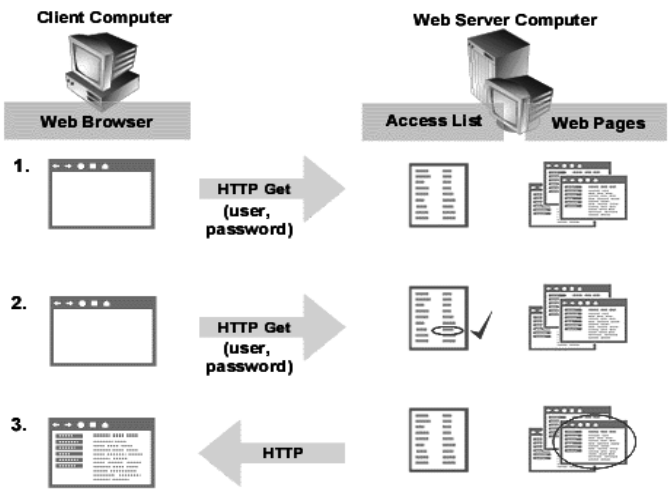


شکل (۳-۱۰) فلوچارت عملیات مرورگر برای دریافت یک صفحهٔ وب

نام فرمان	توضیـــح
GET	تقاضا برای دریافت یک صفحه وب از سرویس‌دهنده
HEAD	تقاضا برای دریافت سرآیند[1] یک صفحه وب
PUT	تقاضا برای ذخیره کردن یک صفحه وب روی یک سرویس‌دهنده
POST	تقاضا برای ضمیمه کردن اطلاعاتی به یک منبع (مثل فایل یا صفحه وب)
DELETE	تقاضا برای حذف یک صفحه وب
LINK	تقاضای برقراری پیوند بین دو منبع موجود
UNLINK	تقاضای خاتمه پیوند دو منبع موجود

جدول (٤-١٠) فرامین تعریف شده در پروتکل HTTP

- **متود GET** : مرورگر با ارسال این متود از سرویس‌دهنده تقاضا می‌کند که یک صفحه وب یا یک فایل دودویی مثل فایل تصویر یا صدا برایش ارسال شود. دقت کنید که فایلهای صدا یا تصویر درقالب استاندارد MIME ارسال خواهد شد؛ یعنی حتی فایلهای دودویی نیز بایستی طبق یکی از روشهای تبدیل در استاندارد MIME به حالت متنی تغییر شکل داده شود و سپس ارسال گردد. در جلوی متود GET نام فایل درخواستی درج می‌شود. در شکل (۵-۱۰) محاورهٔ مرورگر و سرویس‌دهندهٔ HTTP برای دریافت یک صفحهٔ وب را نشان می‌دهد.



شکل (۵-۱۰) محاورهٔ مرورگر و سرویس‌دهندهٔ HTTP برای دریافت یک صفحه

[1] Header

اگر بعد از نام فایل درخواستی گزینهٔ "If-Modified-Since:" که در ادامهٔ آن تاریخ و زمان درج شده ، اضافه شود سرویس‌دهنده را وادار می‌کند که فایل درخواستی را به شرطی ارسال نماید که آن فایل بعد از تاریخ و زمان مشخص شده ، تغییر کرده باشد. با این گزینه مرورگر می‌تواند درصورتی که قبلاً فایل را دریافت کرده و از آن تاریخ به بعد تغییری نداشته ، آنرا از روی ماشین محلی بار کند تا سرعت نمایش صفحات بیشتر شده و اطلاعات بیهوده مبادله نشود. مثال:

GET /hypertext/www/theproject.html HTTP/1.0 If-Modified- Since: Sat 29 Oct 1999

- **متود HEAD** : این متود از سرویس‌دهنده تقاضا می‌کند که فقط سرآیند صفحه وبی را که نام آن در جلوی متود درج شده ، ارسال نماید. این متود چند کاربرد دارد :

اوّل آنکه مشخصات صفحه وب ، شامل تاریخ آخرین تغییر ، عنوان صفحه ، نام تدوین کننده و صاحب اصلی آن و برخی از مشخصات اختیاری که در سرآیند صفحهٔ وب درج شده ، ارسال می‌شود و این اطلاعات میتواند برای مقاصدی همانند تهیه بانک اطلاعاتی از صفحات وب و طراحی "جستجوگرهای وب[1]" مفید واقع شود.

دوم آنکه می‌توان با این متود صحیح بودن یک URL و وجود یک صفحه وب را ارزیابی کرد.

- **متود PUT** : این متود عکس عمل GET است یعنی مرورگر تقاضا می‌کند که یک صفحه وب را بر روی ماشین سرویس‌دهنده ذخیره نماید. این متود را سرویس‌دهنده‌هائی حمایت می‌کنند که بخواهند صفحات برخی از کاربران را دریافت کرده ضمن ذخیره ، آنها را در اختیار دیگران قرار بدهند. فایلهایی که با این متود ارسال می‌شوند باید طبق استاندارد MIME سازماندهی‌شده باشند.

- **متود POST** : این متود از سرویس‌دهنده تقاضا می‌کند که داده‌هائی را به یک منبع موجود (مثل یک صفحه وب یا یک فایل) اضافه کند. این متود برای ایجاد "صفحات

---

[1] Search Engine

آزاد خبری" ، "تابلو اعلانات[1]" ، محیطهای نظر خواهی یا ارسال اطلاعات برای یک پروسهٔ تحت وِب همانند برنامه‌های CGI مورد استفاده قرار می‌گیرد.

- **متود DELETE** : این متود از سرویس‌دهنده تقاضا می‌کند که یک صفحهٔ وب با نام مشخص را از روی ماشین (ماشین سرویس‌دهنده) حذف نماید.

دقت شود که بسیاری از سرویس‌دهنده‌ها به دلایل امنیتی از متودهای PUT ، POST و DELETE پشتیبانی نمی‌کنند.

- **متودهای LINK و UNLINK** : این دو متود اجازه می‌دهند که بین دو صفحه وب (یا دو منبع) ارتباط و پیوند برقرار شده یا پیوند قبلی خاتمه داده شود؛ برای توضیح بیشتر در مورد این دو متود به RFC-1945 مراجعه نمائید.

وقتی تقاضا به سمت سرویس‌دهنده ارسال می‌شود چه پذیرفته شود و چه پذیرفته نشود ، پاسخی متنی دریافت می‌دارد که معمولاً بصورت زیر است :

رشتهٔ متنی	شماره وضعیت	شمارهٔ نسخه‌پروتکل

مثال :

HTTP/1.0 200 OK

یا HTTP/1.0 304 Not Modified

- **HTTP/1.0** : نسخه پروتکل را مشخص می‌کند.
- **شمارهٔ وضعیت** : شماره ای است سه رقمی که وضعیت اجرای فرمان ارسالی را مشخص می‌نماید. این شمارهٔ سه رقمی بر اساس رقم صدگان به پنج دسته تقسیم می‌شود:
  - 1xx : اطلاعاتی (پاسخی جهت آگاهی بیشتر مشتری)
  - 2xx : عمل درخواستی موفقیت آمیز اجرا شده است.
  - 3xx : URL مورد تقاضا تغییر آدرس داشته است.
  - 4xx : در تقاضای ارسال شده از طرف مشتری خطایی وجود دارد.

[1] Bulletin Board

• 5xx : در سرویس‌دهنده خطایی داخلی رخ داده است.

در صورتی که که رقم صدگان ۳ ، ٤ یا ٥ باشد وضعیت فرمان ارسالی ناموفق بوده است.

♦ **رشتهٔ متنی** : متن کوتاهی که وضعیت اجرای فرمان را به زبان طبیعی توصیف می‌کند.

برخی از شماره‌های وضعیت در جدول جدول (٦-١٠) فهرست شده اند.

پس از ارسال اوّلین خطّ پاسخ توسط سرویس‌دهنده در خطوط بعدی نیز یک یا چند سطر اطلاعات اضافی برای مرورگر ارسال می‌شود که می‌تواند برای تفسیر داده‌ها بسیار مهم باشد. قالب برخی از این اطلاعات در جدول (٧-١٠) مشخص شده است. پس از این خطوط که **سرآیند** نامیده می‌شوند ، یک سطر خالی مرز داده‌های فایل را از سرآیند مشخص کرده و در ادامه داده‌های فایل ارسال می‌شوند.

شماره	توصیف متنی	توضیح
200	OK	فرمان پذیرفته شد.
204	No content	چیزی وجود ندارد که به مشتری برگشت داده شود.
301	Moved permanently	URL درخواست شده برای همیشه تغییر کرده است.
302	Moved temporarily	URL درخواست شده موقتاً تغییر کرده است.
304	Not modified	عدم تغییر سند درخواستی پس از تاریخ اعلام شده
400	Bad request	فرمان صادره شناخته نشد.
401	Unauthorized	فرمان ارسالی به احراز هویت کاربر نیاز دارد.
403	Not permitted	فرمان صادره غیر مجاز است و اجرا نشد.
404	Not found	صفحه وب یا فایل درخواستی پیدا نشد.
500	Server error	بروز اشکال داخلی در سرویس‌دهنده
502	Bad gateway	برنامهٔ CGI پاسخ نمیدهد یا وجود ندارد.
503	Service Unavailable	سرویس‌دهنده به دلیل خاصی فعلاً سرویس نمی‌دهد.

جدول (٦-١٠) معرفی مجموعه‌ای از پاسخهای سرویس دهندهٔ HTTP

نوع اطلاعات	قالب اطلاعات
محل دقیق و مطلق منبع یا فایل درخواستی	**Location:** *absoluteURL*
نوع و نام سرویس دهندهٔ وب	**Server:** *product*
تاریخ و زمان آخرین تغییر منبع یا فایل	**Last-Modified:** *HTTP-date*
تاریخ و زمان منقضی شدن منبع یا فایل	**Expires:** *HTTP-date*
روش کد گذاری بدنهٔ پیام (استاندارد MIME)	**Content-Encoding:** *encoding*
طول داده‌ها بر حسب بایت (استاندارد MIME)	**Content-Length:** *length*
نوع سند ارسالی (استاندارد MIME)	**Content-Type:** *type*

جدول (۷-۱۰) معرفی مجموعه ای از پاسخهای سرویس دهندهٔ HTTP

بگونه ای که اشاره شد ، در پروتکل HTTP همانند پروتکل SMTP ، تمام اطلاعات در قالب کاراکترهای اَسکی مبادله می‌شوند و هر گاه فایل دودویی (مثل صدا و تصویر) مبادله شود ، باید طبق استاندارد MIME (از طریق یکی از دو روش ASCII Armor یا qouted-printable–encoding) به حالت متنی تبدیل شود.

برای آشنائی با چگونگی ارسال داده‌ها از طرف سرویس‌دهنده ، در شکل (۸-۱۰) پاسخی را که سرویس‌دهندهٔ HTTP به یک تقاضای دریافت صفحه وب برگردانده ، می‌بینیم. این تقاضا بصورت زیر فرستاده شده است :

GET /example/hello.htm HTTP/1.0

```
HTTP/1.0 200 OK
Server: Microsoft-IIS/4.0
Date: Friday, 9-OCT-1998 10:15:00 GMT
Last-Modified: Monday, 5-OCT-1998 21:35:30 GMT
Content-Type: text/html
Content-Length: 98

<html>
<head>
<title>Example of HTTP Response</title>
</head>
<body>
HELLO WORLD!
</body>
</html>
```

شکل (۸-۱۰) ساختار اطلاعات ارسالی از سرویس‌دهنده در پاسخ به تقاضای یک صفحهٔ وب

در شکل (۹-۱۰) مثالی دیگر از چگونگی مبادلهٔ فرمان و پاسخ بین برنامهٔ سرویس دهندهٔ HTTP و برنامهٔ مشتری برای دریافت یک صفحهٔ وب نشان داده شده است. بگونه ای که در این شکل مشاهده می‌کنید ابتدا سعی شده با ماشینی با نام حوزه www.csc.com و پورت ۸۰ ارتباط TCP برقرار شود؛ برای اینکار از برنامهٔ Telnet استفاده شده است. برای برقراری ارتباط ، برنامهٔ Telnet ابتدا سعی کرده که نام حوزه را به آدرس IP ترجمه کند؛ سپس اقدام به برقراری ارتباط کرده است. در ادامه کاربر توسط متود GET تقاضای یک صفحهٔ وب با نام toc.html را صادر کرده است. دقت شود که پس از ارسال رشته فوق در قالب یک خط مجزا ، باید یک خط خالی دیگر نیز ارسال شود. (یعنی دو عدد کاراکتر Carriage Return متوالی)

سرویس‌دهنده پس از دریافت فرمان فوق ابتدا پنج سطر حاوی اطلاعات متنی را بعنوان سرآیند آورده است و سپس با یک خط خالی قسمت سرآیند را خاتمه داده و در ادامه داده‌های فایل درخواستی ارسال را ارسال کرده است. معنای خطوط سرآیند به شرح ذیل می‌باشد:

- **HTTP/1.0 200 Document follows :** تقاضای مرورگر پذیرفته شده و صفحهٔ درخواستی در ادامه ضمیمه می‌باشد.
- **MIME-Version: 1.0 :** روش سازماندهی اطلاعات مبتنی بر استاندارد MIME نسخهٔ ۱ می‌باشد.
- **Server: CERN/3.0 :** نام برنامهٔ سرویس‌دهنده CERN و نسخهٔ آن ۱ است.
- **Content-Type: text/html :** محتوای داده‌ها متنی و از نوع HTML می‌باشد. این گزینه در استاندارد MIME تشریح شد.
- **Content-Length: 8247 :** داده‌های ارسال شده مجموعاً ۸۲٤۷ بایت می‌باشد.
- یک سطر خالی انتهای بخش سرآیند را مشخص می‌کند.

پس از معرفی **"زبان نشانه گذاری صفحات ابرمتن"** بازهم به سراغ پروتکل HTTP خواهیم آمد.

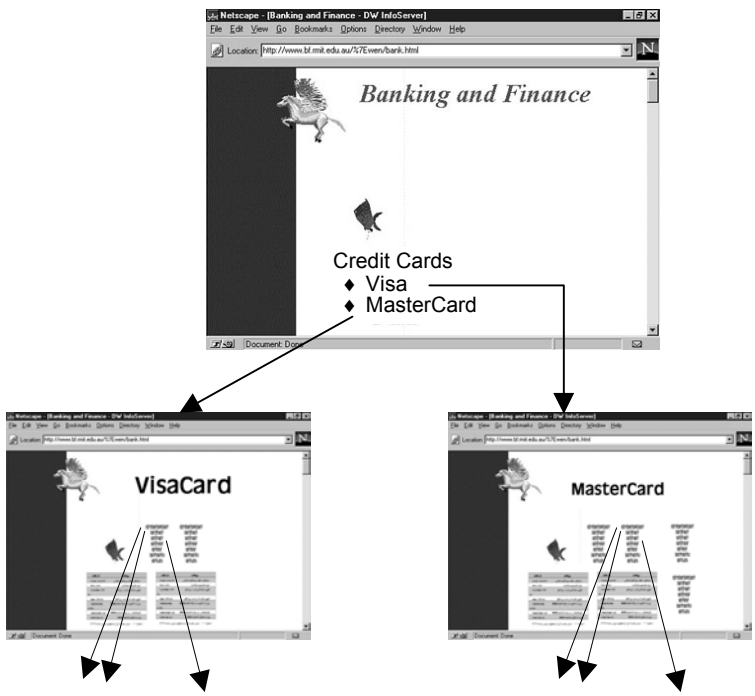
```
C: telnet www.csc.com 80
T: Trying 18.23.0.23 …
T: Connected to www.csc.com.
T: Escape character is '^]'.
C: GET /client-server-computing/toc.html HTTP/1.0
C:
S: HTTP/1.0 200 Document follows
S: MIME-Version: 1.0
S: Server: CERN/3.0
S: Content-Type: text/html
S: Content-Length: 8247
S:
S: <html>
S: <head>
S: <meta http-equiv="Content-Type"
S: content="text/html; charset=windows-1256">
S: <meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
S: <title>Client/Server Computing Table of Contents</title>
S: </head>

S: <body bgcolor="#FFFFFF" text="#000000" link="#0000FF" Vlink="#FF0000">
S:
S: <h1 align="center"><font color="#FF0000">Client/Server Computing </font>
S: <font color="#FF0000" size="4">Second Edition</font></h1>
S:
S: <ul>
S:    <li><ul>
S:          <li><a href="cscfm.htm#I8"><font size="3">Foreword</font></a></li>
S:          <li><a href="cscfm.htm#I9"><font size="3">Preface</font></a></li>
S:          <li><a href="cscfm.htm#I10"><font size="3">Acknowledgments</font></a></li>
S:          <li><a href="cscfm.htm#I12"><font size="3">Introduction</font></a></li>
S:    </ul> </li>
S:  <li><a href="cs01.htm#I1"><font size="3">— 1 —The  Business Opportunity</font></a></li>
S:  <li><a href="cs02.htm"><font size="3">— 2 —Advantages of  Client/Server Computing</font></a></li>
S:  <li><a href="cs03.htm"><font size="3">— 3 —Components of   Client/Server Applications—The Client</font></a></li>
S:  <li><a href="cs04.htm"><font size="3">— 4 —Components of  Client/Server Applications—The Server</font></a></li>
S:  <li><a href="cs05.htm"><font size="3">— 5 —Components of   Client/Server Applications—Connectivity</font></a></li>
S:  <li><a href="cs06.htm"><font size="3">— 6 —Client/Server  Systems Development—Software</font></a></li>
S:  <li><a href="cs07.htm"><font size="3">— 7 —Client/Server  Systems Development—Hardware</font></a></li>
S:  <li><a href="cs08.htm"><font size="3">— 8 —Client/Server  Systems Development—Service and Support</font></a></li>
S:  <li><a href="csc09.htm"><font size="3">— 9 —Client/Server  Systems Development—Training</font></a></li>
S:  <li><a href="csc10.htm"><font size="3">— 10 —The Future  of Client/Server Computing</font></a></li>
S:  <li><a href="cscxa.htm"><font size="3">— Appendix A —Case   Studies</font></a></li>
S:  <li><a href="cscxb.htm"><font size="3">— Appendix B —Apple/IBM  Joint Venture</font></a></li>
S:  <li><a href="cscxc.htm"><font size="3">— Appendix C —Electronic  Document Management standards</font></a></li>
S: </ul>
S: </body>
S: </html>
```

شکل (۹-۱۰) محاورهٔ سرویس‌دهنده و مشتری برای دریافت یک صفحهٔ وب

۴) زبان نشانه گذاری ابر متن : HTML

صفحات HTML ، متون غنی شده‌ای هستند که اطلاعات موجود در یک سند را بصورت صفحه‌آرایی شده و سازمان‌یافته ، در اختیار کاربر قرار می‌دهند. بزرگترین حسن این صفحات آنست که به کاربر این امکان را می‌دهند که به سادگی به صفحهٔ دیگری دسترسی پیدا کند؛ به گونه‌ای که می‌توان مجموعه‌ای از اطلاعات خام را بصورت سلسله‌مراتبی و سطح‌بندی شده در اختیار علاقمندان قرار داد. در شکل (۱۰-۱۰) یک صفحهٔ خانگی بسیار ساده از یک سایت نشان داده شده است. این صفحه دارای دو آیتم است که کاربر با کلیک کردن روی آن ، صفحهٔ دیگری را که مرتبط با موضوع دلخواهش می‌باشد ، دریافت می‌کند. آن صفحه نیز به صفحات مرتبط دیگری پیوند خورده است؛ این روند ممکن است تا چندین سطح ادامه داشته باشد.



شکل (۱۰-۱۰) یک صفحهٔ خانگی بسیار ساده از یک سایت در اینترنت

HTML (زبان نشانه‌گذاری ابرمتن) ، زبانی مانند پاسکال ، بیسیک یا C نیست بلکه روشی است که بواسطه آن می‌توان متون خالص و معمولی[۱] را صفحه‌آرایی کرده و عواملی مثل صدا ، تصویر ، پنجره ، منو و فهرستهای انتخاب را به متن اضافه کرد.

HTML حاوی فرامین صفحه‌آرایی است. این فرامین که در بطن متن اصلی قرار می‌گیرد ، "برچسب"[۲] نام دارد. برچسبهای درون متن توسط مرورگر تشخیص داده شده و پس از تفسیر ، نمایش متون را تحت تاثیر قرار می‌دهند. برچسبهای HTML با علامتهای < > از متن اصلی متمایز می‌شود و عبارتی که در بین آن قرار می‌گیرد ، توسط مرورگر از متن اصلی تشخیص داده خواهد شد. تاثیر عملی که با برچسب درون < ...> مشخص می‌شود ، با علامت < .../> لغو می‌گردد. به عنوان مثال متن سادهٔ زیر را در نظر بگیرید:

Internet Engineering

با استفاده از برچسبهای زیر می‌توان مرورگر را وادار کرد که متن را بصورت پررنگ و کج[۳] نشان بدهد:

<I><B>Internet Engineering</B></I>

مرورگر با تفسیر برچسبها ، متن را بصورت زیر نمایش می‌دهد:

Internet Engineering

کسی که با برچسبهای HTML آشنا باشد براحتی می‌تواند با یک ویرایشگر ساده ، متن خود را سازماندهی کند. ساختار کلی یک صفحهٔ HTML بصورت زیر است:

```
<HTML>
   <HEAD>
      <TITLE>
      عنوان سند در این ناحیه درج می‌شود.
      </TITLE>
   </HEAD>
   <BODY>
      تمام اطلاعات صفحهٔ وب در این ناحیه درج می‌شود.
   </BODY>
</HTML>
```

---

۱ Plain Text
۲ Tag
۳ Bold Italic

در ادامه با برخی از برچسبهای HTML آشنا می‌شویم. هدف از این بخش فقط آشنایی با کلیات HTML است و برای یادگیری اصولی آن باید به مراجع اصلی و کتب تفصیلی مراجعه کرد.

♦ یکی از اصلیترین برچسبها در سازماندهی متون ، برچسب پاراگراف است. مرورگر پاراگراف را بر اساس عرض پنجرهٔ نمایش ، تنظیم می‌کند و در صورت لزوم جملات را شکسته و به خطوط بعد می‌برد؛ به این کار پوشش خودکار گفته می‌شود. یک پارگراف با برچسب <P> آغاز می‌شود و با </P> خاتمه می‌یابد. به مثال زیر دقت کنید:

در اینجا خط بدلیل بزرگ بودن عرض صفحه ، شکسته نشده است.

در اینجا خط بدلیل کوچک بودن عرض صفحه شکسته شده است.

♦ برای ارائهٔ یک مطلب مهم که نظر مخاطب را به خود جلب کند از برچسب <EM> استفاده می‌شود؛ در این حالت متنی که تحت تاثیر این برچسب قرار می‌گیرد بصورت کج نشان داده می‌شود. برای تاکید بیشتر می‌توان از برچسب <STRONG> استفاده کرد تا متن را بصورت پررنگ نشان بدهد.

♦ برای آنکه یک سطر را قطع کرده و سطر بعدی از سر خط جدید آغاز شود از برچسب
 استفاده می‌شود. این برچسب می‌تواند عملیات نمایش متن را در سطر جدید بصورت زیر کنترل کند:

<BR CLEAR=LEFT> : خاتمهٔ سطر فعلی و شروع سطر بعدی از سمت چپ به راست

<BR CLEAR=RIGHT> : خاتمهٔ سطر فعلی و شروع سطر بعدی از سمت راست به چپ

<BR CLEAR=ALL> : خاتمهٔ سطر فعلی و شروع سطر بعدی در کل عرض پنجرهٔ نمایش

♦ **ابرپیوند**[1] : قبلاً اشاره شد که در یک صفحهٔ وب هر آیتم از متن می‌تواند به صفحهٔ دیگر اشاره کند. برای آنکه قطعه‌ای از متن بصورت "ابرپیوند" عمل کرده و کاربر با کلیک کردن روی آن صفحهٔ دیگری را دریافت نماید ، از برچسب <A HREF=.....> استفاده می‌شود. به مثال زیر دقت کنید :

<A HREF="introduc.html"> Introduction

[1] Hyperlink / Hyper Reference

این عبارت باعث می‌شود که کلمهٔ Introduction در متن بصورت پررنگ نشان داده شده و کاربر با کلیک روی آن ، مرورگر را وادار کند تا فایل introduc.html را بارگذاری[1] نموده و نمایش بدهد. در زیر یک ابرپیوند کامل را می‌بینید.

```
<A HREF="http://www.w3.org/hypertext/DataSource/www/Geo.html "> Geographical.html
```

♦ برای نمایش تصاویر گرافیکی در متن از برچسب <... IMG> استفاده می‌شود. تصاویر فشرده شدهٔ نوع GIF و JPEG قابل بارگذاری در یک صفحهٔ وب می‌باشد:

```
<IMG ALIGN = TOP SRC="filename.gif">
```

با این عبارت مرورگر تصویر مشخص شده را بر بالای خط فعلی متن قرار می‌دهد.

```
<IMG ALIGN = MIDDLE SRC="filename.gif">
<IMG ALIGN = BOTTOM SRC="filename.gif">
```

♦ **نمایش فهرستها** : در HTML سه نوع فهرست تعریف می‌شود:

• **فهرست بی‌شماره** : این نوع فهرست با برچسب <UL> و </UL> مشخص می‌شود و عناوین فهرست ، با برچسب <LI> و </LI> تفکیک می‌شود. بهترین توضیح مثال زیر است:

```
<UL>
<LI> عنوان ۱ </LI>
<LI> عنوان ۲ </LI>
<LI> عنوان ۳ </LI>
</UL>
```

نمایش این قسمت از متن بصورت زیر است :

- **عنوان ۱**
- **عنوان ۲**
- **عنوان ۳**

فهرست شماره‌دار : این نوع فهرست مشابه با فهرست قبلی است ، با این تفاوت که عناوین فهرست به ترتیب شماره‌گذاری می‌شود. ابتدای فهرست شماره‌دار با <OL> مشخص می‌شود و برچسب <LI> برای جداسازی عناوین از یکدیگر می‌باشد؛ پایان فهرست نیز با برچسب </OL> مشخص می‌گردد. به مثال زیر دقت کنید:

[1] Download

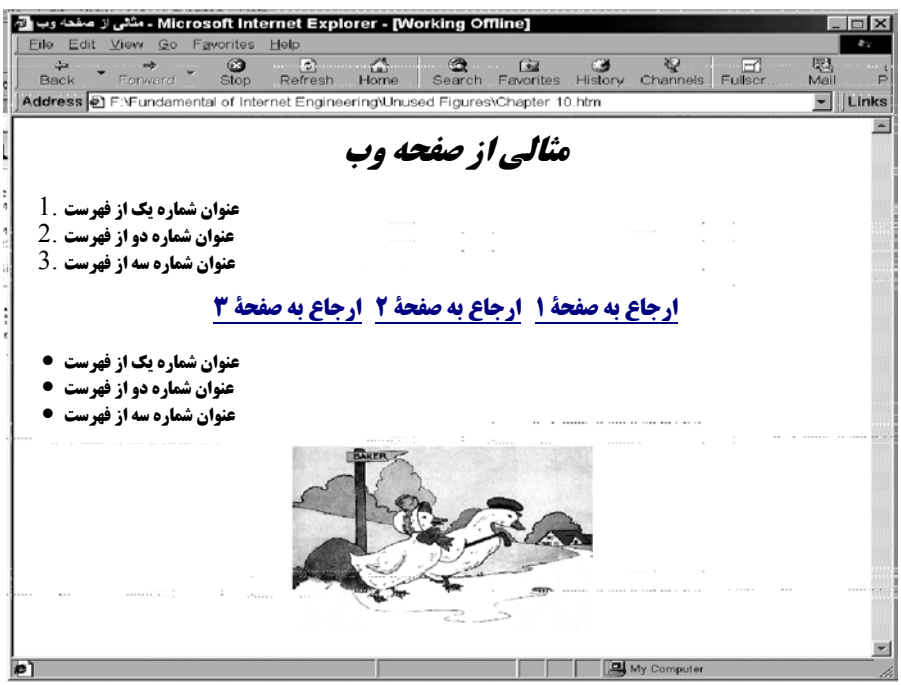
```
<OL>
<LI> عنوان ۱ </LI>
<LI> عنوان ۲ </LI>
<LI> عنوان ۳ </LI>
</OL>
```

نمایش این قسمت از متن بصورت زیر است :

1. عنوان ۱
2. عنوان ۲
3. عنوان ۳

♦ برای تعیین نوع و اندازهٔ قلم (فونت) از برچسب <font size="xxx" face="Font Name"> استفاده می‌شود که xxx اندازهٔ قلم و Font Name نام قلم مورد نظر را تعیین می‌کند.

برای آشنایی با برچسبهای فوق به شکل (۱۱-۱۰) دقت کنید. این شکل یک صفحهٔ HTML را در محیط مرورگر نشان می‌دهد؛ اصل فایل HTML آن در جدول (۱۲-۱۰) نشان داده شده است.



شکل (۱۱-۱۰) یک صفحهٔ HTML در محیط مرورگر

```
<html>

<head>
<title>اصول مهندسی اینترنت - مثالی از صفحهٔ وب</title>
</head>

<p align="center"><font size="6" face="Nazanin">
<em><strong>مثالی از صفحهٔ وب</strong></em></font></p>

<ol>
    <li>عنوان شماره یک از فهرست</li>
    <li>عنوان شماره دو از فهرست</li>
    <li>عنوان شماره سه از فهرست</li>
</ol>
</font></a><font size="5" face="Zar">
<p align="center">
<a href="http://www.malekian.com/page1.html"><strong>ارجاع به صفحهٔ ۱
</strong>
<a href="http://www.malekian.com/page1.html"><strong>ارجاع به صفحهٔ ۲
</strong>
<a href="http://www.malekian.com/page1.html"><strong>ارجاع به صفحهٔ ۳
</strong>

<ul>
    <li>عنوان شماره یک از فهرست</li>
    <li>عنوان شماره دو از فهرست</li>
    <li>عنوان شماره سه از فهرست</li>
</ul>

<p align="center">
<img src="file:///F:/ANIMALS/Bk.jpg" width="319" height="245">
</font></p>
</body>
</html>
```

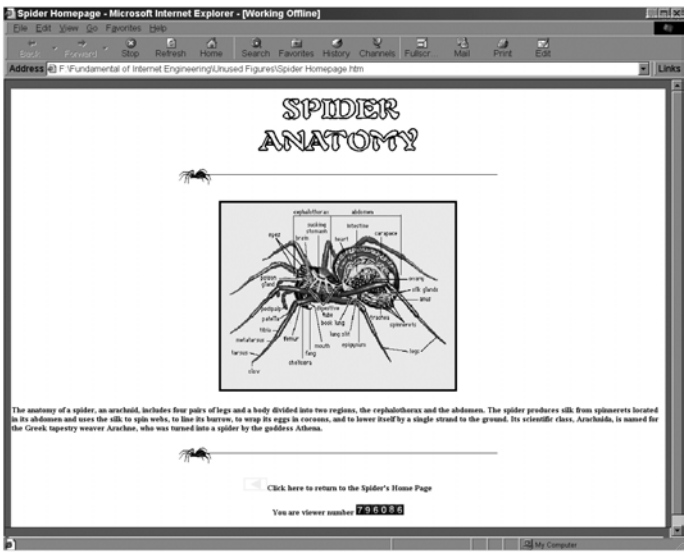
شکل (۱۲-۱۰) فایل HTML مربوط به صفحهٔ وب در شکل (۱۱-۱۰)

۱۴-۱) فرمهای ورود اطلاعات در HTML

آن دسته از صفحات وب که فقط ارائه کنندهٔ اطلاعات به کاربر هستند و هیچ کنش و واکنشی با کاربر ندارند ، اصطلاحاً "صفحات ایستا"[1] نامیده می‌شوند. کاربر با دریافت چنین صفحاتی ، در زمینهٔ موضوع دلخواه خود اطلاعاتی را از سرویس‌دهنده دریافت کرده و مطالعه

[1] Static Pages

می‌کند. در شکل (۱۳-۱۰) یک "صفحهٔ وب ایستا" دیده می‌شود. تمام اجزای صفحه به منظور مطالعهٔ کاربر ، آراسته شده است.



شکل (۱۳-۱۰) یک "صفحهٔ وب ایستا"

یکی از بزرگترین مشخصه‌های وب آنست که کاربر می‌تواند با یک صفحهٔ وب در تراکنش باشد؛ بدین معنا که کاربر قادر است در صفحهٔ وب اطلاعاتی را وارد نموده و آنرا به سمت سرویس‌دهنده ارسال کند. سرویس‌دهنده پس از دریافت اطلاعات و پردازش آن ، مجدداً در پاسخ ، اطلاعاتی را برمی‌گرداند. برای چنین کاربردهایی صفحهٔ وب باید شامل اجزایی جهت ورود اطلاعات باشد. به عنوان مثال فرض کنید یک شرکت مایل است در سایت وب خود ، ضمن معرفی محصولاتِ شرکت ، از مشتریان سفارش بگیرد. بنابراین طراح صفحهٔ وب باید نواحی ورود اطلاعات مشتری را در صفحهٔ وب تعریف کند. پس از آنکه کاربر اطلاعات خود را در این نواحی وارد کرد ، مرورگر را وادار می‌کند تا آنها را برای سرویس‌دهنده بفرستد. در شکل (۱۴-۱۰) یک صفحهٔ وب با قابلیت ورود و ارسال اطلاعات نشان داده شده است. در جدول (۱۵-۱۰) اصل فایل HTML آن ارائه شده است.

(۱۴-۱۰) یک صفحهٔ وب با قابلیت ورود و ارسال اطلاعات

```
 <html>

<head>
<title>نمونه</title>
</head>

<body background="../../SampleWebSite/Nabkgnd.jpg" bgcolor="#FFFFFF">

<p align="center"><font size="6" face="Tabassom"><strong>نمونه‌ای
از صفحهٔ وب با قابلیت ارسال اطلاعات برای سرویس‌دهنده</strong></font></p>

<form method="POST" ACTION="http://www.malekian.com/cgi-bin/proc1.pl">
<p align="right"><font face="Nazanin">
<input type="text"  size="108" name="Name"><strong> : نام</strong></font
<img src="../Figures/shock.gif"  width="52" height="56"></p>
<p dir="rtl">
<img  src="../Figures/shock.gif" width="52" height="56">
<strong>  : نام خانوادگی</strong>
<input type="text" size="140"   name="Family"></p>
<p dir="rtl"><strong>
```

ادامه صفحهٔ بعد

```
<img  src="Telephone.gif" width="50" height="51">شماره تلفن تماس:</strong>
<input type="text" size="79"  name="TelNo">
</p>
<p dir="rtl"><strong>آدرس دقیق پستی:</strong>
<input type="text" size="251" name="Addr1">
</p>
<input type="text"  size="294" name="Addr2">
</p>
<img  src="../Figures/dan.gif" align="left" hspace="0"  width="127" height="180"><strong>
<img  src="../ Figures/fl.gif" width="52" height="56">نام محصول مورد نظر: </strong>
<input type="text"  size="165" name="ProductNName">
</p>
<strong>
<img src="../ Figures/fl.gif" width="52" height="56">کد محصول مورد نظر:</strong>
<input type="text"  size="170" name=" ProductNo">
</p>
<p align="center">
<input type="submit" name="B1"  value="Submit/ارسال">
</p> <p>
<input type="reset" name="B2"   value="Reset/پاک کردن همهٔ فیلدها">
</p>
</form>
</body>
</html>
```

(۱۵-۱۰) اصل فایل HTML مربوط به صفحهٔ وبِ شکل (۱۴-۱۰)

در جدول (۱۵-۱۰) تمام برچسبهایی که در زمینهٔ خاکستری نشان داده شده‌اند ، مربوط به دریافت اطلاعات از کاربر هستند. این برچسبها را با در نظرگرفتن مثال فوق بررسی می‌کنیم:

♦ برای تعیین ناحیهٔ ورود اطلاعات در یک صفحهٔ وب ، از برچسب <... FORM> استفاده می‌شود. ناحیهٔ ورود اطلاعات با برچسب </FORM> خاتمه می‌یابد. هر صفحهٔ وب می‌تواند چند ناحیهٔ ورود اطلاعات داشته باشد؛ اطلاعاتِ هر ناحیه بطور یکجا برای سرویس‌دهنده ارسال خواهد شد. در هر ناحیه حداقل یک گزینهٔ **"تسلیم/ارسال"**[1] وجود دارد که کاربر پس از آنکه اطلاعات این ناحیه را تکمیل کرد ، توسط آن به مرورگر فرمان می‌دهد تا آنها را برای سرویس‌دهنده ارسال کند. اطلاعات ارسالی در قالب مشخص و استاندارد تحویل یک پروسهٔ خاص بر روی ماشین سرویس‌دهنده خواهد شد. نام و آدرس این پروسه درون برچسب <... FORM> مشخص شده است. این پروسه که اصطلاحاً **CGI** نامیده می‌شود ، توسط سرویس‌دهنده فراخوانی شده و اطلاعات ارسالی را دریافت می‌دارد. در ضمن این پروسه

[1] Submit

وظیفه دارد اطلاعات دریافتی را پردازش و در صورت لزوم ذخیره کند و پاسخهای مناسب را به کاربر برگرداند. (در ادامه CGI را بررسی خواهیم کرد.)

برچسب <... FORM> دارای ویژگیهای زیر است:

- ACTION : آدرس URL مربوط به محل قرار گرفتن برنامهٔ CGI که اطلاعات ارسالی را دریافت و پردازش خواهد کرد.
- METHOD : یکی از متودهای HTTP که توسط آن اطلاعات به سمت سرویس‌دهنده ارسال می‌شود. این متودها می‌تواند GET یا POST باشد. [به جدول (٤-١٠) مراجعه کنید؛ تفاوت این متودها در بخشهای آتی تشریح خواهد شد.]
- ENCTYPE : نوع کدگذاری داده‌های ارسالی را مشخص می‌کند. (چون ارسال اطلاعات از مرورگر به سرویس‌دهنده از استاندارد MIME تبعیت می‌کند لذا در این قسمت نوع کدگذاری داده‌ها مشخص می‌شود-مثل base64-. اگر این خصوصیت معرفی نشود ، اطلاعات ارسالی ، متن معمولی فرض خواهد شد.)

♦ **برچسب <... INPUT>**

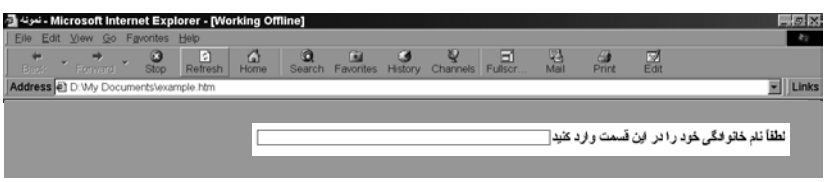
با استفاده از این برچسب می‌توان یکی از عوامل دریافت اطلاعات را در صفحهٔ وب تعریف کرد. ویژگیهای این برچسب عبارتند از:

- **TYPE** : در صفحهٔ وب می‌توان انواع اشیاء و عوامل ورود اطلاعات را تعریف کرد. با ویژگی TYPE ، می‌توان نوع عامل ورودی را تعیین کرد. انواع عوامل ورودی عبارتند از:

TYPE=TEXT : یک فضا برای ورود اطلاعات متنی فراهم می‌آورد. مثال:

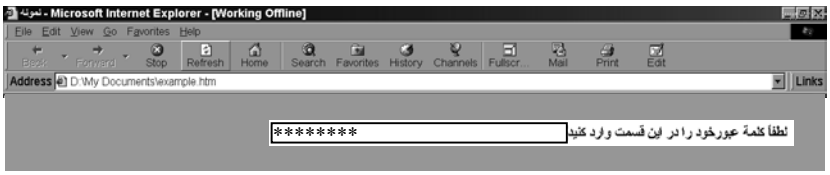
```
<p>
<input type="text" size="152" name="T1">
<strong>لطفاً نام خانوادگی خود را در این قسمت وارد کنید
</p >
```

نمایش این عامل ورودی در محیط مرورگر به صورت زیر است:



TYPE=PASSWORD : یک فضا برای ورود کلمهٔ عبور فراهم می‌آورد؛ با این تفاوت که در هنگام ورود اطلاعات ، محتوای آن دیده نخواهد شد. مثال:

```
<p>
<input type="password" size="152" name="T1">
لطفاً کلمهٔ عبور خود را در این قسمت وارد کنید<strong></p>
```



TYPE=CHECKBOX : یک لیست از گزینه‌ها را تعریف می‌کند تا کاربر بتواند به دلخواه ، هر کدام را انتخاب نماید. مثال:

```
<font face="Sina">
<p><input type="checkbox" checked name="C1" value="ON"><strong>گزینهٔ ۱</strong></p>
<p><input type="checkbox" name="C2" ><strong>گزینهٔ ۲</strong></p>
<p><input type="checkbox" name="C3" ><strong>گزینهٔ ۳</strong></p>
<p><input type="checkbox" name="C4" ><strong>گزینهٔ ۴</strong></p>
```

TYPE=RADIO : یک لیست از گزینه‌ها را تعریف می‌کند تا کاربر بتواند به دلخواه ، فقط یکی از آنها را انتخاب نماید. مثال:

```
<font face="Sina">
<p><input type="radio" checked name="R1" value="V1"><strong>گزینهٔ ۱</strong></p>
<p><input type="radio" name="R1" value="V2"><strong>گزینهٔ ۲</strong></p>
<p><input type="radio" name="R1" value="V3"><strong>گزینهٔ ۳</strong></p>
<p><input type="radio" name="R1" value="V4"><strong>گزینهٔ ۴</strong></p>
```

TYPE=SUBMIT : یک "کلید فشاری"[1] تعریف می‌کند تا کاربر به مرورگر فرمان بدهد که اطلاعات را به سمت سرویس‌دهنده ارسال نماید. مثال:

```
<p align="center">
<input type="submit" name="B1"  value="Submit/ارسال">
</p>
```

TYPE=RESET : یک "کلید فشاری" تعریف می‌کند تا کاربر به مرورگر فرمان بدهد که اطلاعات درون تمام عوامل ورودی اطلاعات را پاک نماید. مثال:

```
<p>
<input type="reset" name="B2"   value="Reset/پاک کردن همهٔ فیلدها">
</p>
```

در مثال زیر نقش کلید RESET مشخص شده است.

انتخاب نام برای تمام ورودیها بجز کلیدهای SUBMIT و RESET الزامی است زیرا وقتی مرورگر اطلاعات را برای سرویس‌دهنده ارسال می‌نماید ، نام فیلد مربوطه را هم به همراه

[1] Push Botton

مقدار آن ، ارسال خواهد کرد. برنامهٔ CGI ، وظیفه دارد نام فیلدها و مقادیر آنها را از هم تفکیک کند. در تمام اشیاء و عوامل ورودی ، گزینهٔ VALUE مقدار پیش‌فرض آنرا تعیین می‌کند. برای کلیدهای فشاری SUBMIT و RESET ، مقدار گزینهٔ VALUE عنوانی است که بر روی کلیدها ظاهر خواهد شد.

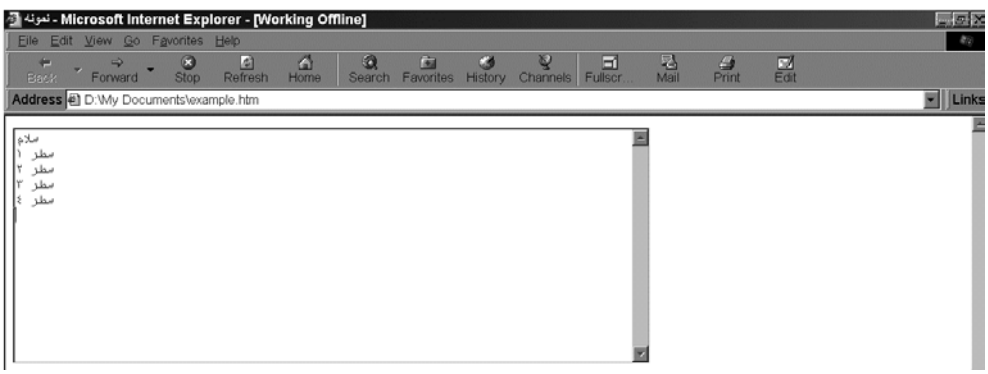
- **برچسب <... TEXTAREA>**

با استفاده از این برچسب می‌توان یک فضای چندخطی ورودی در صفحهٔ وب تعریف کرد. این فضا به کاربر اجازه خواهد داد تا متون طولانی خود را وارد کرده ، ارسال نماید. چنین فیلدی می‌تواند برای کاربردهایی نظیر ورود متن نامه یا نظرسنجی استفاده شود. برخی از ویژگیهای این برچسب عبارتند از:

- NAME : نام فیلد
- ROWS : تعداد سطرهای ورودی
- COLS : تعداد ستونهای ورودی
- DISABLED : ورودی را غیر فعال می‌کند.

مثال :

```
<p><textarea name="text1" rows="15" cols="80">
</textarea> </p>
```



۴-۲) قابلیتهای دیگر HTML

استاندارد HTML بطور مداوم رو به توسعه و پیشرفت است. پس از معرفی نسخهٔ سوم HTML در ۱۹۹۵ تقریباً هر سال ایده‌های جدیدی در این زمینه ارائه شده است. در نسخهٔ HTML 3.0 قابلیتهای زیر عرضه شده‌اند:

♦ **امکان رسم جدول** : جداول یکی از نیازهای اصلی صفحات وب تلقی می‌شوند در حالی که نسخه‌های قدیمی HTML ، این امکان را فراهم نکرده بودند. در نسخهٔ ۳ ، رسم جدول با برچسب <TABLE ...> امکان‌پذیر شد. قالب این برچسب بصورت زیر است:

<TABLE WIDTH="?" COLS="?" BORDER="?" FRAME="?" CELLSPACING="?" CELLPADDING="?">

WIDTH : عرض جدول COLS : تعداد ستونها

BORDER : ضخامت حاشیهٔ اطراف جدول بر حسب پیکسل

FRAME : حاشیهٔ قابل رویت اطراف جدول

CELLSPACING : فضای خالی میان خانه‌های جدول

CELLPADDING : فضای خالی بین داده‌های درون جدول

پس از تعریف ساختار جدول سطر و ستونهای آن با برچسبهای زیر تعریف می‌شوند:

برچسبهای <TR> و </TR> : یک سطر از جدول را تعیین می‌کند.

برچسبهای <TD> و </TD> : محتوی هر یک از خانه‌های جدول توسط یک برچسب <TD> مشخص می‌شود.

به مثال زیر دقت کنید. در این مثال یک جدول ۳×۳ تعریف و به خانه‌های آن مقدار داده شده است.

```
<font size="6" face="Nazanin">
<table border="3" cellpadding="3" cellspacing="3">
  <tr>
    <td><p align="center"><strong>ستون ۱ سطر ۱ </strong></td>
    <td><p align="center"><strong>ستون ۲ سطر ۱ </strong></td>
    <td><p align="center"><strong>ستون ۳ سطر ۱ </strong></td>
  </tr>
<tr>
    <td><p align="center"><strong>ستون ۱ سطر ۲ </strong></td>
    <td><p align="center"><strong>ستون ۲ سطر ۲ </strong></td>
    <td><p align="center"><strong>ستون ۳ سطر ۲ </strong></td>
 </tr>
```

```
<tr>
    <td><p align="center"><strong>ستون ۱ سطر ۳ </strong></td>
    <td><p align="center"><strong>ستون ۲ سطر ۳ </strong></td>
    <td><p align="center"><strong>ستون ۳ سطر ۳ </strong></td>
  </tr>
</table>
```

♦ **جاسازی یک APPLET در صفحهٔ وب** : در یک صفحـهٔ وب بـا اسـتفاده از بـرچسـبهای <... APPLET> می‌توان اپلتهای جاوا را جاسازی کرد. اپلتها ، قابلیتهای حرفه‌ای شبیه مدیریـت نقشه‌های تصویری ، امکانات چندرسـانه‌ای و بازیـهای کـامپیوتری را بـه صفحـات وب ارائـه می‌کنند. مثال :

```
<APPLET CODE="FirstApplet" WIDTH=300 HEIGHT=200> </APPLET>
```

CODE : نام فایل اپلت
WIDTH : عرض پنجره‌ای که اپلت در آن اجرا می‌شود.
HEIGHT : ارتفاع پنجره‌ای که اپلت در آن اجرا می‌شود.

♦ **امکانات فرمول‌نویسی** : در نسخه‌های جدید HTML ارائهٔ فرمولهای ریاضی در یک صفحهٔ وب امکان‌پذیر شده است. این قابلیت معمولاً یکی از نیازهای اولیهٔ سایتهای علمی و تحقیقاتی تلقی می‌شود.

در خاتمهٔ این بخش بار دیگر تاکید می‌کنیم که هدف از معرفی HTML ، فقط آشنایی با اصـول آن بوده است و برای آشنایی تفصیلی با آن باید به مراجع آخر فصل مراجعه کنید.

۴-۳) مزیتهای HTML

HTML اولین زبان نشانه‌گذاری متن نبود و در دههٔ نود (دورهٔ تولد و رشد فراگیر HTML) حتی ایدهٔ جدیدی هم محسوب نمی‌شد. رشد HTML ناشی از تواناییهای آن در برآورده کردن نیازهای شبکهٔ اینترنت ، در موارد زیر بوده است :

- **استقلال**[1] : HTML یک استاندارد مبتنی بر کدهای ASCII است و و هیچ وابستگی اجرایی به سخت‌افزار و نرم‌افزار ندارد. یک فایل HTML می‌تواند بین دو ماشین کاملاً متفاوت مبادله شود ، بدون آنکه هیچ نگرانی در مورد عدم سازگاری ماشینها وجود داشته باشد. هر ماشین با سخت‌افزار و نرم‌افزار خاص خود ، با استفاده از مرورگر سازگار با محیط خود ، یک فایل HTML را تفسیر کرده و نمایش خواهد داد. با توجه به تنوع ماشینها و سیستمهای عامل در شبکهٔ اینترنت ، این خصوصیت یکی از نیازهای بنیادی محسوب می‌شود.
- **سرعت و سادگی** : فایلهای HTML از نظر اندازه و حجم فایل ، کوچک هستند و برای پایین آوردن ترافیک شبکه ، ابزاری مناسب محسوب می‌شوند؛ در عین حال پیچیدگی خاصی ندارند و به راحتی می‌توان یک فایل HTML ایجاد و استفاده کرد. سادگی این استاندارد باعث شد که در زمان بسیار کوتاهی ابزارهای قدرتمند طراحی صفحات وب پدید آید و تولید صفحات وب در حد یک کار تجربی و بدون نیاز به اطلاعات فنی ، برای عموم آسان شود.
- **امکان دریافت اطلاعات از صفحهٔ وب** : صفحات وب ابزار بسیار ساده و مناسبی برای دریافت اطلاعات از کاربر و ارسال آن به سرویس‌دهنده هستند؛ کاری که اگر نیاز باشد مستقل از وب انجام شود ، به تخصص و زمان بسیار زیادی احتیاج دارد. این امکان باعث شده که محیط وب از حالت اطلاع‌رسانی صِرفْ درآمده و به یک ابزار مناسب و سریع جهت امور اقتصادی ، اداری ، تجاری تبدیل شود.
- **سازمان‌دهی سلسله‌مراتبی** : صفحات وب با استفاده از مفهوم ابرپیوند ، مستندات را بصورت سلسله‌مراتبی و دسته‌بندی‌شده به متقاضی عرضه می‌کنند. در این روش دسترسی به اطلاعات ، بسیار سریع و ساده خواهد شد.
- **پشتیبانی همگانی** : سادگی و جذابیتِ وب باعث شد که تمام توسعه‌دهندگان نرم‌افزار ، در سیستمهای خود از آن پشتیبانی کنند. امروزه سیستمی را نمی‌توان یافت که از وب پشتیبانی نکند یا مرورگر نداشته باشد. امروزه اکثر بانکهای اطلاعاتی قادرند در محیط وب نیز بکارگرفته شوند. یعنی کاربر از راه دور و با استفاده از مرورگر و استاندارد HTML با آن تراکنش داشته باشد.

---

[1] Platform Independence

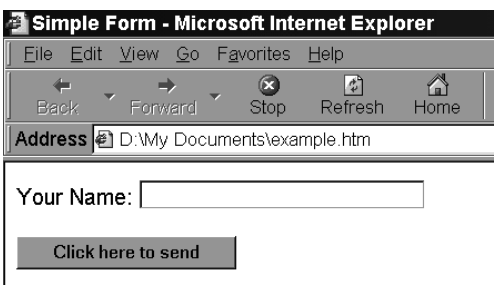
۵) برنامه‌های CGI[1]

فرض کنید که یک صفحهٔ وب ، اطلاعاتی را از کاربر دریافت کند. کاربر می‌تواند با فشار دادن کلید SUBMIT آنها را برای سرویس‌دهنده ارسال نماید. سرویس‌دهندهٔ HTTP فقط وظیفهٔ دریافت یا ارسال داده‌ها را بر عهده دارد و کاری در مورد پردازش آنها انجام نمی‌دهد. حال دو سوال زیر مطرح می‌شود :

- کدام برنامه بر روی ماشین سرویس‌دهنده داده‌های ارسالی از مرورگر را دریافت و پردازش می‌نماید؟
- مرورگر بر اساس چه الگویی داده‌ها را ارسال می‌کند و برنامهٔ پردازش‌کنندهٔ داده‌ها چگونه آنها را از سرویس‌دهندهٔ HTTP تحویل می‌گیرد؟

وقتی طراح صفحهٔ وب ، یک ناحیه را برای ورود اطلاعات با برچسب <FORM ...> تعریف می‌کند ، موظف است در درون این برچسب آدرس برنامهٔ تحویل‌گیرنده و پردازش‌کنندهٔ داده‌ها را دقیقاً مشخص نماید. به عنوان مثال به قطعه کد HTML زیر و نمایش آن در محیط مرورگر دقت کنید.

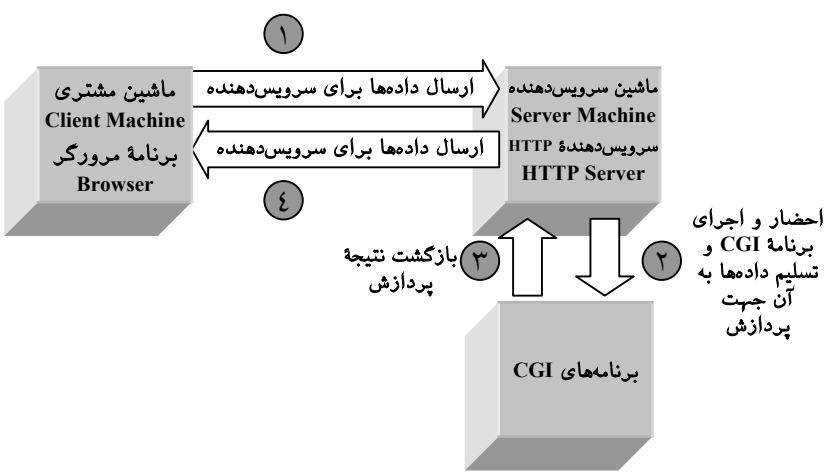
```
<HTML>
<HEAD><TITLE>Simple Form</TITLE></HEAD>
<BODY>
<FORM Method="POST" Action="http://www.abcdef.com/cgi-bin/prog.exe">
 Your Name:
<INPUT Name="user" SIZE="30"><P>
<INPUT Type=submit Value="Click here to send">
</FORM>
</BODY>
</HTML>
```



[1] Common Gateway Interface

در این قطعه کد ، نام و محل برنامهٔ تحویل گیرندهٔ داده‌ها با آدرس URL http://www.abcdef.com/cgi-bin/prog.exe مشخص شده است. وقتی مرورگر داده‌ها را برای سرویس‌دهندهٔ HTTP ارسال می‌کند ، سرویس‌دهنده برنامهٔ prog.exe را بارگذاری و اجرا کرده و داده‌ها را تحویل آن می‌دهد. این برنامه قادر است ضمن پردازش داده‌ها ، پاسخهای مناسب را در قالب استاندارد HTML تولید کرده و برای کاربر باز پس بفرستد. این برنامه اصطلاحاً CGI نامیده می‌شود. برنامه‌های CGI با نامهای "اسکریپتِ CGI"[1] یا "برنامهٔ کاربردی CGI"[2] هم معرفی شده‌اند.

CGI استانداردی است که چگونگی ارتباط برنامه‌های جانبی با سرویس‌دهندهٔ HTTP را تبیین می‌کند. پروتکل HTTP به تنهایی فقط قادر به ارسال و دریافت صفحات وب است و برنامه‌های CGI در کنار این پروتکل می‌توانند یک ارتباط دوطرفه با کاربر ایجاد نمایند؛ به گونه‌ای که کاربر می‌تواند از راه دور با این برنامه‌ها تراکنش داشته باشد. در حقیقت HTTP به عنوان یک پروتکل واسطِ انتقال داده ، بین کاربر و این برنامه‌ها انجام وظیفه می‌کند. شکل (١٦-١٠) این مفهوم را نشان می‌دهد.



شکل (١٦-١٠) تراکنش مرورگر و برنامهٔ CGI از طریق پروتکل HTTP

[1] CGI Script
[2] CGI Application

برنامه‌های CGI را می‌توان به زبانهای مختلفی نوشت و نیاز به ابزار خاصی ندارد. زبانهایی که امکان نوشتن برنامه‌های CGI را فراهم آورده‌اند ، عبارتند از:

- C
- C++
- Perl
- Tcl
- Visual Basic (Micosoft Windows)
- Shell Scripts (UNIX)
- Apple Scripts
- Delphi

برنامه‌های CGI ، برنامه‌های کاملاً معمولی هستند؛ تنها تفاوت آنها در دریافت داده‌ها از ورودی است. برنامه‌های معمولی داده‌های خود را از طریق صفحه‌کلید یا موْس دریافت می‌کنند ، درحالی که برنامه‌های CGI ورودیهای خود را از سرویس‌دهندهٔ HTTP می‌گیرند. خروجیهای این برنامه‌ها نیز به سرویس‌دهندهٔ HTTP ارسال می‌شود تا برای مشتری فرستاده شده و در محیط مرورگر نشان داده شود. برنامه‌های CGI معمولاً هیچگونه خروجی یا پنجره بر روی ماشین سرویس‌دهنده ندارند. (مگر در موارد خاص)

بطور معمول طراح یک صفحهٔ وب ، خودش برنامهٔ CGI متناظر با آن را برنامه‌نویسی می‌کند؛ زیرا نام فیلدها و اشیاء ورودی در یک صفحهٔ وب ، باید در برنامهٔ CGI متناظر با آن شناخته شده و تطابق داشته باشد و مرورگر محتوای هر یک از این فیلدها را همراه نام فیلد ، ارسال می‌نماید.

در ادامه باید الگوی ارسال داده‌ها را به یک برنامهٔ CGI بررسی کنیم.

۱-۵) الگوهای ارسال اطلاعات برای یک برنامهٔ CGI

به دو روش سرویس‌دهندهٔ HTTP یک برنامهٔ CGI را راه‌اندازی کرده و داده‌های ارسالی از مرورگر را تحویل آن می‌دهد. طراح صفحهٔ وب ، در برچسب <... FORM> ، ضمن معرفی کردن محل و نام برنامهٔ CGI ، روش تسلیم داده‌ها را نیز تعریف می‌نماید. این دو روش عبارتند از:

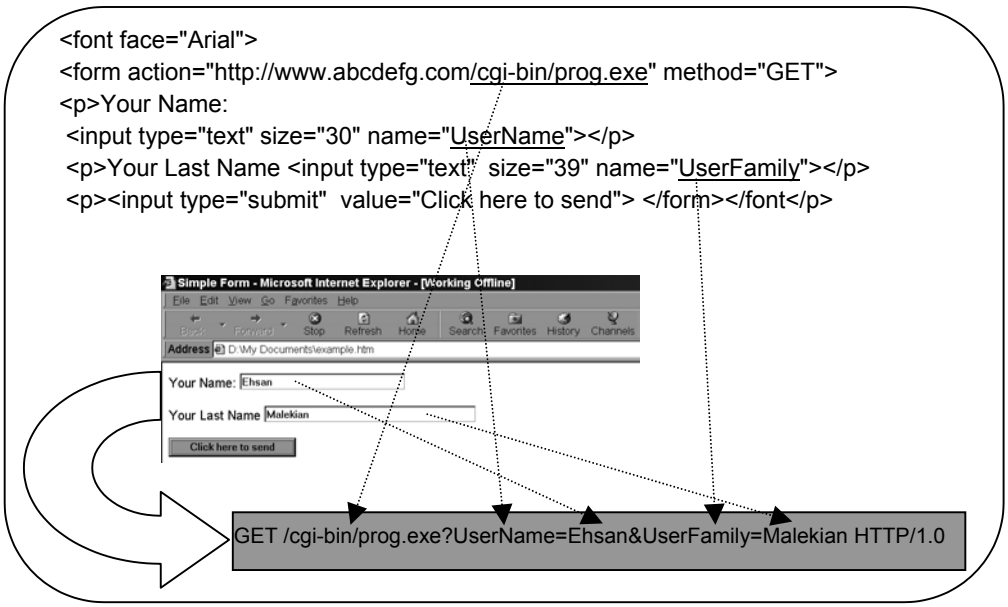
- استفاده از الگوی GET :

```
<FORM Method="GET" Action="http://www.abcdef.com/cgi-bin/prog.exe">
```

- استفاده از الگوی POST :

```
<FORM Method="POST" Action="http://www.abcdef.com/cgi-bin/prog.exe">
```

اگر از الگوی GET برای ارسال داده‌ها استفاده شود ، داده‌های جمع‌آوری شده از صفحهٔ وب به آدرس URL آن ضمیمه شده و به سمت سرویس‌دهنده ارسال خواهد شد. مثال زیر بسیار گویاست:



به گونه‌ای که نشان داده شده اگر از الگوی GET استفاده شود ، مرورگر پس از برقراری ارتباط TCP با سرویس‌دهندهٔ HTTP ، از فرمان معمولی GET استفاده می‌کند ، با این تفاوت که در ادامهٔ آدرس URL ، نام و مقادیر هر فیلد ضمیمه و ارسال می‌شود. وقتی سرویس‌دهندهٔ HTTP این رشته را دریافت می‌کند ، برنامهٔ مشخص شده را اجرا کرده و ادامهٔ رشته را (بعد از کاراکتر ؟) در یک "متغیر محیطی"[1] به نام QUERY_STRING قرار می‌گیرد.[2] برنامهٔ CGI

[1] Environment Variable

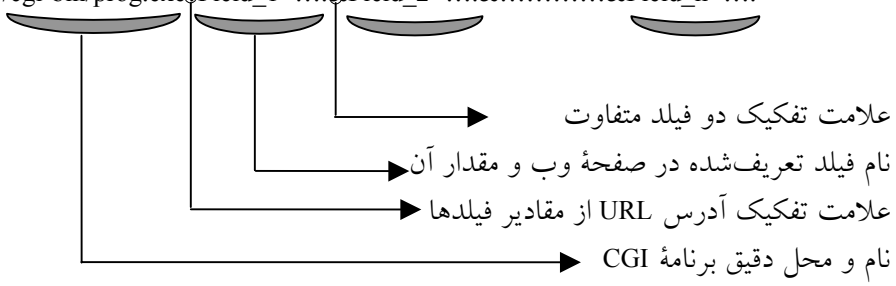
[2] **متغیر QUERY_STRING ، در برنامه نیاز به تعریف ندارد بلکه با استفاده از تابع getenv(...) ، (در زبان C یا توابع معادل در دیگر زبانها) می‌توان به آن دسترسی داشت.**

می‌تواند داده‌ها را از این متغیر استخراج ، تجزیه و تحلیل[1] و پردازش نماید. اگر بخواهیم این روش فراخوانی برنامهٔ CGI را با برنامه‌ها معمولی مقایسه کنیم ، همانند فراخوانی برنامه در خط فرمان بهمراه پارامترهای ورودی است؛ در این برنامه‌ها ، متغیرهای argc و argv[] پارامترهای خط فرمان را در اختیار برنامه‌نویس قرار می‌دهند.

الگوی ارسال داده‌های یک صفحهٔ وب بصورت زیر است:

فرض کنید عوامل و اشیاء یک صفحهٔ وب به نامهای فرضی Field_1 تا Field_n نامگذاری شده باشد. هرگاه کاربر کلید SUBMIT را فشار داد ، رشتهٔ زیر تولید شده و ارسال خواهد شد:

GET /cgi-bin/prog.exe?Field_1=….&Field_2=….&……&Field_n=….



رعایت قواعد زیر در تدوین رشتهٔ ارسالی ، لازم است:

- محل و نام برنامهٔ CGI با علامت **?** از بقیهٔ رشته جدا می‌شود.
- نام هر فیلد و مقدار فیلد با علامت = از هم تفکیک می‌شود.
- اگر قرار باشد چندین فیلد و مقادیر آن ارسال شود ، نام و مقدار هر فیلد با علامت **&** از هم تفکیک می‌شود.
- اگر در بین داده‌ها فاصلهٔ خالی (Blank) وجود داشته باشد ، باید با علامت + جایگزین شود.
- اگر در بین داده‌ها کاراکترهای ASCII با کد زیر ۳۳ یا یکی از علامتهای (+ = & ٪ ؟) وجود داشته باشد ، ابتدا علامت % و سپس کد هگزادسیمال آن بجای آن کاراکتر قرار می‌گیرد. مثلاً کاراکتر با کد ۳۲ بصورت 20% یا کاراکتر با کد ۳۰ بصورت 1E% جایگزین می‌شود.

مثال :

GET /cgi-bin/prog.exe?Name=Ali+Reza&Family=Ahmadi+Tehrani

[1] Parse

استفاده از الگوی GET زمانی مناسب خواهد بود که مجموع کل رشته‌ای که به سمت سرویس‌دهنده ارسال می‌شود زیر هزار کاراکتر باشد ، زیرا سرویس‌دهنده‌های HTTP رشته‌های با طول بیش از هزار کاراکتر را نخواهند پذیرفت.[1] اکثر برنامه‌های CGI از الگوی POST استفاده می‌کنند.

۲-۵) متغیرهای محیطی قابل استفاده در یک برنامهٔ CGI

به غیر از متغیر محیطی QUERY_STRING ، متغیرهای دیگری نیز هستند که برنامهٔ CGI می‌تواند در صورت لزوم از آنها استفاده کند. این متغیرهای محیطی در جدول (۱۷-۱۰) معرفی شده‌اند. برای دسترسی به این متغیرهای در یک زبان برنامه‌نویسی همانند C باید آدرس آنرا با استفاده از تابع سیستمی (**"نام متغیر محیطی"**)getenv بدست آورد. مثال :

```
p = getenv("CONTENT_LENGTH");
 if(p != NULL) {…}
else { …. }
```

نام متغیر محیطی	محتوی
REQUEST_METHOD	الگوی ارسال داده‌ها (یکی از دو مقدار GET یا POST)
QUERY_STRING	رشتهٔ بعد از علامت ؟ ضمیمه شده به URL (رشتهٔ حاوی نام و مقادیر فیلدها)
CONTENT_LENGTH	طول داده‌های ارسالی به برنامهٔ CGI (بر حسب بایت) (اگر الگوی ارسال POST باشد.)
CONTENT_TYPE	نوع داده‌های ارسالی به برنامهٔ CGI (مشخصهٔ MIME) (اگر الگوی ارسال POST باشد.)
PATH_INFO	مسیر دقیق و محل قرار گرفتن برنامهٔ CGI (مشخص شده در URL)
HTTP_AccEPT	نوعی از قالب داده‌ها که برنامهٔ مرورگر قادر است ، بپذیرد. (مشخصهٔ MIME)
GATEWAY_INTERFACE	نام و نسخهٔ CGI (مثلاً CGI/1.1)
SERVER_PORT	شمارهٔ پورت سوریس‌دهندهٔ HTTP که تقاضا به آن ارسال شده است. (بطور معمول ۸۰)
SERVER_PROTOCOL	نام و نسخهٔ سرویس‌دهندهٔ وب (مثلاً HTTP/1.0)
SERVER_SOFTWARE	نام و نسخهٔ نرم‌افزار نصب شده بعنوان سرویس‌دهندهٔ وب (مثلاً IIS/4.0)
SERVER_NAME	نام ماشینی که سرویس‌دهندهٔ وب بر روی آن نصب شده (مثلاً www.ibm.com)
REMOTE_ADDR	آدرس IP ماشین سرویس‌دهندهٔ وب

جدول (۱۷-۱۰) متغیرهای محیطی قابل استفاده در یک برنامهٔ CGI

[1] در برخی از سرویس‌دهنده‌ها ، حداکثر طول یک خط ۲۵۵ کاراکتر و پیش‌فرض آن ۸۰ است.

۳-۵) الگوی POST

برای آن دسته از صفحات وب که حجم نامشخصی از داده‌ها را برای سرویس‌دهنده ارسال می‌کند ، برنامهٔ CGI باید از الگوی POST استفاده کند. اگر برای فراخوانی برنامهٔ CGI از الگوی POST استفاده شود ، سرویس‌دهندهٔ HTTP ، داده‌ها را از طریق متغیر محیطی QUERY_STRING به برنامه نمی‌فرستد بلکه از طریق "ورودی استاندارد" (یعنی همان مفهوم stdin در زبان C) به برنامه هدایت می‌شود. در این حالت برنامهٔ CGI می‌تواند از دستورات معمولی خواندن از کنسول ورودی (مثل تابع fscanf() یا fgetc() در زبان C یا cin در C++) برای دریافت داده‌ها از سرویس‌دهنده اقدام کند.

هر برنامهٔ CGI ممکن است بخواهد برای اطلاع کاربر نتیجه‌ای را به مرورگر برگرداند. واضح است که خروجی یک برنامهٔ CGI باید روی مرورگر کاربر نشان داده شود و اینگونه برنامه‌ها بطور معمول خروجی خاصی بر روی سرویس‌دهنده ندارند. برای برگرداندن اطلاعات به مرورگری که برای یک برنامهٔ CGI داده ارسال کرده است ، از "خروجی استاندارد" (یعنی مفهوم stdout در زبان C) استفاده می‌شود. یعنی برای نمایش خروجی‌ها در برنامهٔ CGI می‌توان از دستورات معمولی نوشتن روی کنسول خروجی (مثل تابع printf() در زبان C یا cout در C++) استفاده کرد.

در حقیقت سرویس‌دهندهٔ HTTP هنگامی که یک برنامهٔ CGI را فراخوانی می‌کند ، مسیر کنسول ورودی و خروجی استاندارد (stdin و stdout) را به سمت خودش برمی‌گرداند.[۱]

در هنگام نوشتن بر روی کنسول خروجی (یعنی زمانی که پیغامی جهت نمایش برای مرورگر ارسال می‌شود) باید به دو نکتهٔ اساسی زیر دقت شود:

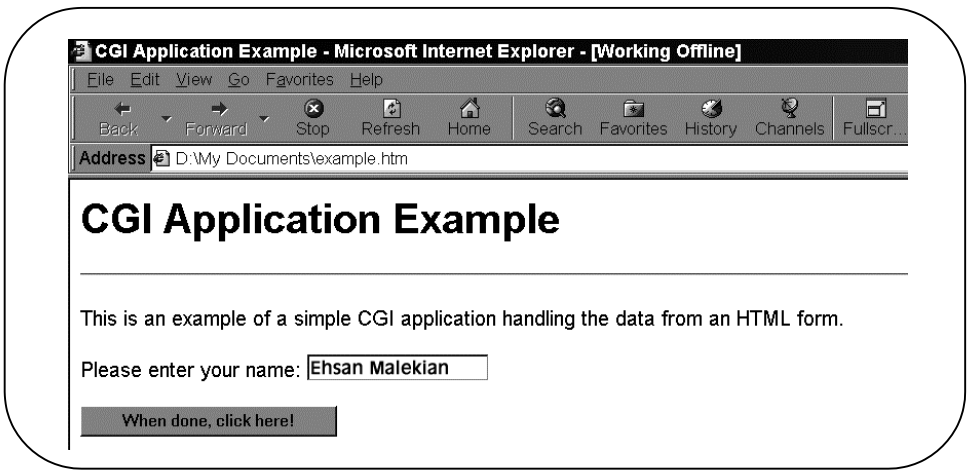
- با توجه به آنکه هر چیزی که با دستورات معمولی مثل printf() بر روی خروجی نوشته می‌شود ، بر روی مرورگر نشان داده خواهد شد ، لذا پیغامها باید در قالب یک فایل HTML نوشته شوند.
- هر خط ارسالی برای مرورگر باید با دو کاراکتر \n\n خاتمه یابد.

برای آشنایی با روش برنامه‌نویسی CGI ارائهٔ یک مثال بسیار راهگشا خواهد بود. به صفحهٔ وب شکل (۱۸-۱۰) و فایل HTML آن دقت کنید.

---

[۱] **برای درک این مفهوم تکنیک Piping در یونیکس و مفهوم کنسول (Consol I/O) را به یاد بیاورید.**

```
<HTML>
<HEAD>
<TITLE>CGI Application Example</TITLE>
</HEAD>
<BODY>
<H1>CGI Application Example</H1>
<hr>
This is an example of a simple CGI application
handling the data from an HTML form.
<BR>
<FORM ACTION="http://www.hqz.com/scripts/cgisamp.exe" METHOD="Post">
Please enter your name: <INPUT NAME="name" TYPE="text"><p>
<input type=submit value="When done, click here!">
</FORM>
</BODY>
</HTML>
```



شکل (۱۸-۱۰) یک صفحهٔ وب فرضی برای فراخوانی برنامهٔ CGI

وقتی کاربر "کلید ارسال" را فشار می‌دهد ، سرویس‌دهنده برنامه‌ای به نام cgisamp.exe را فراخوانی کرده و ورودی را به کنسول stdin هدایت می‌نماید. در ضمن متغیرهای محیطی جدول (۱۷-۱۰) را نیز تنظیم می‌کند. (به غیر از QUERY_STRING) برنامهٔ cgisamp.exe را که به زبان C نوشته شده ، بررسی می‌نماییم. در این برنامه پنج تابع تعریف شده است:

- تابع void strcvrt(char *cStr, char cOld, char cNew) : رشتهٔ cStr را جستجو کرده و هرگاه کاراکتر cOld را در آن بیابد به کاراکتر cNew تبدیل می‌کند.
- تابع static int TwoHex2Int(char *pC) : اگر در یک رشتهٔ ورودی ، کدهایی باشند که با علامت % و معادل هگزادسیمال آن مشخص شده باشند آنرا به کد اصلی بر می‌گرداند.

- تابع void urlDecode(char *p) : این تابع تمام رشتهٔ دادهٔ ارسالی از مرورگر را بررسی و به حالت اصلی تبدیل می‌کند.
- تابع void StoreField(char *f, char *Item) : این تابع زوج "نام فیلد/مقدار" را از رشتهٔ ورودی استخراج می‌کند.
- تابع اصلی برنامه main() : این برنامه داده‌های ارسالی توسط مرورگر را از stdin می‌خواند و بر اساس ورودی (نام کاربر) ، یک خروجی ساده طبق شکل (۱۹-۱۰) برای آن تولید و و از طریق stdout برای مرورگر ارسال می‌کند.

```
<HEAD><TITLE>Submitted OK</TITLE></HEAD>
<BODY><h2>The information you supplied has been accepted.
<br>سپاسگزار از لطف شما<br>Ehsan Malekian</h2>
<h3><A href="http://www.hqz.com/cgisamp.htm">
[Return]</a></h3></BODY>
```

شکل (۱۹-۱۰) خروجی تولید شده توسط برنامهٔ CGI

در ادامه اصل برنامهٔ CGI که به زبان C نوشته شده ، ارائه شده است. این برنامه با صفحهٔ وب شکل (۱۸-۱۰) تراکنش داشته و یک خروجی مطابق با شکل (۱۹-۱۰) تولید و ارسال می‌نماید. این برنامه با نام cgisamp.exe بر روی سرویس‌دهنده نصب می‌شود. این برنامه را در جدول (۲۰-۱۰) ملاحظه می‌کنید. اگر به دستورات printf() دقت کنید هیچ تفاوتی با برنامه‌نویسی معمولی زبان C ندارند؛ با این تفاوت که پیغامهای ارسالی روی خروجی ، متن معمولی نیستند بلکه با برچسبهای HTML غنی شده‌اند تا بر روی مرورگر نشان داده شوند.

```
/**************************************************************************
* File: cgisamp.c
*
* Use: CGI Example Script.
*
* Notes: Assumes it is invoked from a form and that REQUEST_METHOD is POST.
* Ensure that you compile this script as a console mode app.
*
* This script is a modified version of the script that comes with EMWAC
* HTTPS.
*
* Date: 8/21/95
* Christopher L. T. Brown clbrown@netcom.com
*
**************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <io.h>

char InputBuffer[4096];
static char * field;
static char * name;

/* Convert all cOld characters   */
/* in cStr into cNew characters. */
void strcvrt(char *cStr, char cOld, char cNew)
{
    int i = 0;

    while(cStr[i])
    {
        if(cStr[i] == cOld)
            cStr[i] = cNew;
        i++;
    }
}

/* The string starts with two hex */
/* characters.  Return an integer */
/* formed from them.              */
static int TwoHex2Int(char *pC)
{
    int Hi, Lo, Result;

    Hi = pC[0];
    if('0' <= Hi && Hi <= '9')
        Hi -= '0';
    else if('a' <= Hi && Hi <= 'f')
        Hi -= ('a' – 10);
    else if('A' <= Hi && Hi <= 'F')
        Hi -= ('A' – 10);

    Lo = pC[1];
    if('0' <= Lo && Lo <= '9')
        Lo -= '0';
    else if('a' <= Lo && Lo <= 'f')
```

```
            Lo -= ('a' – 10);
        else if('A' <= Lo && Lo <= 'F')
            Lo -= ('A' – 10);

        Result = Lo + 16 * Hi;
        return(Result);
    }

    /* Decode the given string in-place */
    /* by expanding %XX escapes.       */
    void urlDecode(char *p)
    {
        char *pD = p;

        while(*p)
        {
            if (*p == '%')      /* Escape: next 2 chars are hex          */
            {                   /* representation of the actual character.*/
                p++;
                if(isxdigit(p[0]) && isxdigit(p[1]))
                {
                    *pD++ = (char)TwoHex2Int(p);
                    p += 2;
                }
            }
            else
                *pD++ = *p++;
        }
        *pD = '\0';
    }

    /* Parse out and store field=value items. */
    /* Don't use strtok!                      */
    void StoreField(char *f, char *Item)
    {
        char *p;

        p = strchr(Item, '=');
        *p++ = '\0';
        urlDecode(Item);
        urlDecode(p);
        strcvrt(p, '\n', ' ');
        strcvrt(p, '+', ' ');       /* Get rid of those nasty +'s */
        field = f;                  /* Hold on to the field just in case. */
        name = p;                   /* Hold on to the name to print*/
    }

    int main(void)
    {
        int ContentLength, x, i;
        char *p,
            *pRequestMethod,
            *URL,
            *f;

        /* Turn buffering off for stdin.*/
        setvbuf(stdin, NULL, _IONBF, 0);

        /* Tell the client what we're going to send */
```

```
        printf("Content-type: text/html\n\n");

        /* What method were we invoked through? */
        pRequestMethod = getenv("REQUEST_METHOD");

        /* Get the data from the client     */
        if(strcmp(pRequestMethod,"POST") == 0)
        {
            /* according to the requested method.*/
            /* Read in the data from the client. */
            p = getenv("CONTENT_LENGTH");
            if(p != NULL)
                ContentLength = atoi(p);
            else
                ContentLength = 0;
            if(ContentLength > sizeof(InputBuffer) –1)
                ContentLength = sizeof(InputBuffer) –1;

            i = 0;
            while(i < ContentLength)
            {
                x = fgetc(stdin);
                if(x == EOF)
                    break;
                InputBuffer[i++] = x;
            }
            InputBuffer[i] = '\0';
            ContentLength = i;

            p = getenv("CONTENT_TYPE");
            if(p == NULL)
                return(0);

            if(strcmp(p, "application/x-www-form-urlencoded") == 0)
             {
                p = strtok(InputBuffer, "&");      /* Parse the data */
                while(p != NULL)
                {
                    StoreField(f, p);
                    p = strtok(NULL, "&");
                }
             }
         }

        URL = getenv("HTTP_REFERER");           /* What url called me.*/
        printf("<HEAD><TITLE>Submitted OK</TITLE></HEAD>\n");
        printf("<BODY><h2>The information you supplied has been accepted.");
        printf("<br> سپاسگزار از لطف شما<br>%s</h2>\n", name);
        printf("<h3><A href=\"%s\">[Return]</a></h3></BODY>\n", URL);

        return(0);
    }
```

جدول (۲۰–۱۰) اصل برنامهٔ CGI

۷) مراجع این فصل

مجموعهٔ مراجع زیر می‌توانند برای دست آوردن جزییات دقیق و تحقیق جامع در مورد مفاهیم معرفی شده در این فصل مفید واقع شوند.

"Computer Networks" , Andrew S.Tanenbaum, Third Edition, Prentice-Hall, 1996.	
"Internet and Intranet Engineering", Daniel Minoli , McGraw-Hill, NewYork 1997.	
"Internetworking with TCP/IP", Commer D.E. Prentice-Hall, 1996.	
"Special Edition Using VRML",Atephen Matsuba , Bernie Roehl, 1998.	
"CGI Programming Unleashed , Eugene Eric Kim , 1996 by Sams.net Publishing.	
"HTML 3.2 and CGI", Professional Reference Edition,UNLEASHED , John December and Mark Ginsburg, Sams.net Publishing.	
"CGI Manual of Style" , Robert McDaniel, ISBN: 1-56276-397-0.	
"Teach yourself VRML 2.0 in 21 days", Chris Arrin Bruse Campbell, 1997.	
RFC1945	Hypertext Transfer Protocol -- HTTP/1.0. T. Berners-Lee, R. Fielding & H. Frystyk. May 1996
RFC2068	Hypertext Transfer Protocol -- HTTP/1.1. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. January 1997
Microsoft Internet URL : http://www.microsoft.com/internet	
The VRML Repository URL : http://www.sdsc.edu/vrml/	
The HTML Guru URL : http://members.aol.com/htmlguru/index.html	
Tim Berners-Lee's Style Guide URL : http://www.w3.org/hypertext/WWW/Provider/Style/Overview.html	
The Home of the WWW Consortium URL : http://www.w3.org/	
The World Wide Web FAQ URL : http://www.boutell.com/faq/	
The 24-Hour HTML Café URL : http://www.mcp.com/sams/books/235-8/	
The Developer's JumpStation URL : http://oneworld.wa.com/htmldev/devpage/dev-page.html	