From HTML to PostGIS

Michał Okulewicz

Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska

Lecture plan

HTTP protocol

Definition

Text messages protocol, operating on the basis of request-response scheme. Utilizes TCP/IP as the lower level transport protocol.

Address parts

$$\underbrace{\text{http}}_{\textit{protocol}}://\underbrace{\text{www.mini.pw.edu.pl}}_{\textit{domain}}:\underbrace{80}_{\textit{port}}/\underbrace{\sim\text{okulewiczm/www/}}_{\textit{path}}?\underbrace{\text{Dydaktyka:IO}}_{\textit{query}}$$

HTML forms

Idea

HTML forms have been designed as the basic mechanism for data exchange between UI and web server.

Security

HTML forms offer only an illusion of password security. HTTP by itself is a non-encrypted open text protocol.

HTML forms

Idea

HTML forms have been designed as the basic mechanism for data exchange between UI and web server.

Security

HTML forms offer only an illusion of password security. HTTP by itself is a non-encrypted open text protocol.

Data transfer

Attribute	**Purpose**
`method`	Type of request (GET, POST, (PUT, DELETE))
`action`	Address of the component ready to process data
`enctype`	Way in which data is encoded (`multipart/form-data`, `application/x-www-form-urlencoded`)

How does the browser know what to do with data?

Selected MIME types

Attribute	**Purpose**
`text/html`	HTML document
`text/plain`	Plain text document
`image/jpeg`	JPEG encoded image
`application/octet-stream`	Binary data

Response

- Static document served from remote machine (MIME type defined by mapping the extension)
- Dynamically generated document (MIME type explicitly defined in `Content-Type` header)
- The first idea of web application architecture were CGI containers enclosing console applications and redirecting input/output streams in form of HTTP requests and responses
- The second idea were template pages consisting partially of HTML code and partially of generated content (PHP, ASP, JSP, .NET WebForms etc.)
- The current idea are Single Page Applications consisting of HTML/CSS layout utilized by JavaScript application with content provided from REST services in a form of JSON objects

Response

- Static document served from remote machine (MIME type defined by mapping the extension)
- Dynamically generated document (MIME type explicitly defined in `Content-Type` header)
- The first idea of web application architecture were CGI containers enclosing console applications and redirecting input/output streams in form of HTTP requests and responses
- The second idea were template pages consisting partially of HTML code and partially of generated content (PHP, ASP, JSP, .NET WebForms etc.)
- The current idea are Single Page Applications consisting of HTML/CSS layout utilized by JavaScript application with content provided from REST services in a form of JSON objects

Response

- Static document served from remote machine (MIME type defined by mapping the extension)
- Dynamically generated document (MIME type explicitly defined in `Content-Type` header)
- The first idea of web application architecture were CGI containers enclosing console applications and redirecting input/output streams in form of HTTP requests and responses
- The second idea were template pages consisting partially of HTML code and partially of generated content (PHP, ASP, JSP, .NET WebForms etc.)
- The current idea are Single Page Applications consisting of HTML/CSS layout utilized by JavaScript application with content provided from REST services in a form of JSON objects

Response

- Static document served from remote machine (MIME type defined by mapping the extension)
- Dynamically generated document (MIME type explicitly defined in `Content-Type` header)
- The first idea of web application architecture were CGI containers enclosing console applications and redirecting input/output streams in form of HTTP requests and responses
- The second idea were template pages consisting partially of HTML code and partially of generated content (PHP, ASP, JSP, .NET WebForms etc.)
- The current idea are Single Page Applications consisting of HTML/CSS layout utilized by JavaScript application with content provided from REST services in a form of JSON objects

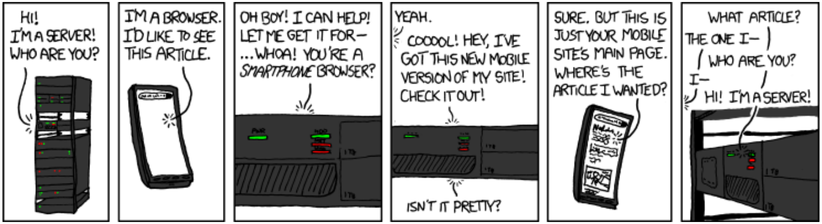
Examples

- CGI approach
- Different forms encoding (tutorial: Lab12FromWebsite/Task1)
- Image generation (tutorial: Lab12FromWebsite/Task5)
- Cache (tutorial: LectureExample_09_PlainWebForms)

Basic Common Gateway Interface (CGI) script

```
#include "stdafx.h"
int _tmain(int argc, _TCHAR* argv[]) {
  printf("Content-type: text/html\n\n");
  printf("<HTML><HEAD><TITLE>The first CGI script</TITLE>\n");
  printf("<BODY>The first line of CGI</BODY></HEAD></HTML>");
  return 0;
}
```

Discover your user


HI! I'M A SERVER! WHO ARE YOU?
I'M A BROWSER. I'D LIKE TO SEE THIS ARTICLE.
OH BOY! I CAN HELP! LET ME GET IT FOR— ...WHOA! YOU'RE A SMARTPHONE BROWSER?
YEAH.
COOOOL! HEY, I'VE GOT THIS NEW MOBILE VERSION OF MY SITE! CHECK IT OUT!
ISN'T IT PRETTY?
SURE. BUT THIS IS JUST YOUR MOBILE SITE'S MAIN PAGE. WHERE'S THE ARTICLE I WANTED?
WHAT ARTICLE? THE ONE I—
WHO ARE YOU?
I—
HI! I'M A SERVER!

alt-text

They have to keep the adjacent rack units empty. Otherwise, half the entries in their /var/log/syslog are just 'SERVER BELOW TRYING TO START CONVERSATION *AGAIN*.' and 'WISH THEY'D STOP GIVING HIM SO MUCH COFFEE IT SPLATTERS EVERYWHERE.'

Source: https://xkcd.com/869/

User preferences and browser type

Another tool in Responsive Web Design

- Browser file types preferences
- Browser language preferences
- Browser and system type

Session and cookies

Identifying users

- Hidden form fields
- URL rewriting
- Cookies

Session

An abstract concept allowing for storing user-related data between HTTP calls. Nowadays, mostly obsolete due to poor scaling in cloud environments. However, the **idea** persists.

Cookies

- Domain
- Path
- Expiration date

Session and cookies

Identifying users

- Hidden form fields
- URL rewriting
- Cookies

Session

An abstract concept allowing for storing user-related data between HTTP calls. Nowadays, mostly obsolete due to poor scaling in cloud environments. However, the **idea** persists.

Cookies

- Domain
- Path
- Expiration date

Session and cookies

Identifying users

- Hidden form fields
- URL rewriting
- Cookies

Session

An abstract concept allowing for storing user-related data between HTTP calls. Nowadays, mostly obsolete due to poor scaling in cloud environments. However, the **idea** persists.

Cookies

- Domain
- Path
- Expiration date

Session and cookies

Identifying users

- Hidden form fields
- URL rewriting
- Cookies

Session

An abstract concept allowing for storing user-related data between HTTP calls. Nowadays, mostly obsolete due to poor scaling in cloud environments. However, the **idea** persists.

Cookies

- Domain
- Path
- Expiration date

Cookies I

Features

- New cookies are identified by a name and can store a string value
- Cookie should be added to the response in order to be sent to the client's browser or deleted from it
- Cookies are send by the browser with each of the requests
- Up till 20 cookies, 4kB each can be created by one server on one client!

Cookies II

Domain

The form of the domain name is specified by RFC 2109. A domain name begins with a dot (e.g. .domain.com) and means that the cookie is visible to servers in a specified Domain Name System (DNS) zone (for example, www.domain.com, but not www.another.com).
That property of the cookies is the reason of redirection in big web services (e.g. Google: gmail.com ← mail.google.com)

Cookies III

Path

The cookie is visible to all the pages in the directory you specify, and all the pages in that directory's subdirectories. A cookie's path must include the server address that set the cookie, for example, /catalog, which makes the cookie visible to all directories on the server under /catalog.

Cookies IV

Expiration date

Sets the maximum age of the cookie in seconds / expiration date:

- A positive value / date later than now indicates that the cookie will expire after that many seconds have passed.
- A negative value / no date means that the cookie is not stored persistently and will be deleted when the Web browser exits.
- A zero value / date earlier than now causes the cookie to be deleted.

Examples

- Content adaptation (tutorial: LectureExample_09_PlainWebForms)
- Session and cookies (tutorial: Lab12FromWebsite/Task2)