Лекция 4

Загрузка файлов по протоколу HTTP

Загрузка файлов на сервер

Загрузка файлов на сервер осуществляется пользователями сети интернет довольно часто:

- Веб-интерфейсы почтовых сервисов
- Интерактивные фотогалереи и фотоальбомы
- Порталы бесплатного программного обеспечения, которые используют для обмена файлами различных программ и т.д.

MIME

- **MIME** (произн. «майм», англ. Multipurpose Internet Mail Extensions — многоцелевые расширения интернет-почты) — спецификация для кодирования информации и форматирования сообщений таким образом, чтобы их можно было пересылать по Интернету.
- Формат MIME поддерживает передачу нескольких сущностей в пределах одного сообщения.
- Для обозначения множественного содержимого используются медиатипы, обозначающие составные типы данных: **multipart/***.

Некоторые MIME-типы

Внутренний формат прикладной программы:

- application/json: JavaScript Object Notation JSON
- application/javascript: JavaScript
- application/octet-stream: двоичный файл без указания формата
- application/ogg: Ogg
- application/pdf: Portable Document Format, PDF
- application/postscript: PostScript
- application/xhtml+xml: XHTML
- application/zip: ZIP
- application/x-gzip: Gzip

Некоторые MIME-типы

Аудио:

- audio/mp4: MP4
- audio/mpeg: MP3 или др. MPEG
- audio/ogg: Ogg Vorbis, Speex, Flac или др. аудио
- audio/vnd.wave: WAV

Видео:

- video/mpeg: MPEG-1
- video/mp4: MP4
- video/ogg: Ogg Theora или другое видео
- video/quicktime: QuickTime
- video/x-flv: FLV

Некоторые MIME-типы

Изображения:

- image/gif: GIF
- image/jpeg: JPEG
- image/pjpeg: JPEG
- image/png: Portable Network Graphics
- image/svg+xml: SVG
- image/tiff: TIFF

Причины появления MIME

MIME расширяет функционал электронной почты (а впоследствии и протокола HTTP) для поддержки:

- Текстовых данных в кодировках, отличных от ASCII
- Не текстовых вложений
- Тел заголовков с множественным содержимым

Множественное содержимое

- Множественное содержимое (**Multipart Content**) – наличие больше одного типа данных в едином теле запроса.
- Перед каждой порцией содержимого обязан быть разделитель (**boundary**), представляющий собой случайный набор символов латиницы.
- Наименование разделителя указывается в заголовке **Content-Type** в основном заголовке запроса.

Структура HTTP-заголовков

POST /send-message.html HTTP/1.1

Host: webmail.example.com

Referer: http://webmail.example.com/send-message.html

User-Agent: BrowserForDummies/4.67b

Content-Type: multipart/form-data; boundary=Asrf456BGe4h

Content-Length: (суммарный объём, включая дочерние заголовки)

Connection: keep-alive

Keep-Alive: 300

(пустая строка)

(отсутствующая преамбула)

--Asrf456BGe4h

Content-Disposition: form-data; name="DestAddress"

(пустая строка)

brutal-vasya@example.com

--Asrf456BGe4h

Content-Disposition: form-data; name="MessageTitle"

(пустая строка)

Я негодую

--Asrf456BGe4h

Content-Disposition: form-data; name="MessageText"

(пустая строка)

Привет, Василий! Твой ручной лев, которого ты оставил у меня на прошлой неделе, разодрал весь мой диван.

Пожалуйста забери его скорее!

Во вложении две фотки с последствиями.

--Asrf456BGe4h

Content-Disposition: form-data; name="AttachedFile1"; filename="horror-photo-1.jpg"

Content-Type: image/jpeg

(пустая строка)

(двоичное содержимое первой фотографии)

(отсутствующий эпилог - пустая строка)

Атрибут enctype у форм

- Определяет способ кодирования данных формы при их отправке на сервер. Обычно устанавливать значение атрибута **enctype** не требуется. Однако если используется поле для отправки файла (input type="file"), следует определить атрибут enctype как **multipart/form-data**.

Атрибут enctype у форм

- **application/x-www-form-urlencoded**
  - Вместо пробелов ставится +, символы в нестандартных кодировках (например на кириллице) кодируются их шестнадцатеричными значениями (например, %D0%90%D0%BD%D1%8F вместо Аня).
- **multipart/form-data**
  - Данные не кодируются. Это значение применяется при отправке файлов.
- **text/plain**
  - Пробелы заменяются знаком +, буквы и другие символы не кодируются.

Multipart-формы

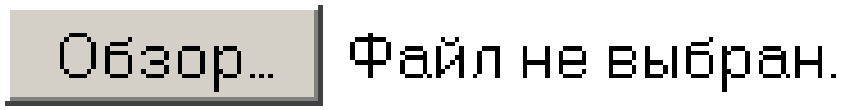
- Загрузка файлов на сервер осуществляется с помощью multipart-формы, в которой есть поле загрузки файла. В качестве атрибута **enctype** указывается значение **multipart/form-data**.

Поле для ввода файла

HTML-код поля для ввода имени файла, который будет загружен на Web-сервер выглядит так:

- `<input type="file" name="myFile">`

В браузере будет отображаться так:



Общий вид формы загрузки файла

```
<form action="upload.cgi" method="POST"
enctype="multipart/form-data">
      <input type="file" name="myFile">
      <input type="submit" value="Загрузить">
</form>
```

Обработка файла на сервере

- CGI-сценарий должен получить файл через множественное содержимое и сохранить на сервере его во **временную директорию** (обычно **/tmp**, но зависит от типа ОС и предпочтений программиста).
- Во временной директории файл хранится под новым **уникальным именем**.
- После завершения работы скрипта, временный файл должен быть **удалён**.

Структура UploadedFile

Для каждого загружаемого файла предлагается создать структуру, в которой хранить необходимые свойства этого файла:

```
typedef struct {
  std::string filename;  // реальное имя файла
  std::string type; // MIME-тип файла
  std::string tmp_name; // временное имя файла
  int error;    // код ошибки (0, если нет)
  int size;     // размер загружаемого файла
} UploadedFile;
```

Перемещение временного файла

```
int move_uploaded_file(UploadedFile tmpFile,
std::string path);
```

- Функция проверяет, является ли файл *tmpFile* загруженным на сервер (переданным по протоколу HTTP POST). Если файл действительно загружен на сервер, он будет перемещён в место, указанное в аргументе *path*.

Пример работы скрипта

```
void main() {
	UploadedFile tstFile;
	HTTP http = new HTTP();
	if(tstFile = http.getFile("attachedFile"))
			http.move_uploaded_file(tstFile,
"/home/pavel/files/"+tstFile["filename"]);
}
```

Дополненная структура класса HTTP

```
class HTTP
{
    public:
        …
        std::string getHeader(std::string name);
        // возвращает значение HTTP-заголовка “name”
        std::string rawURLDecode(std::string name);// декодирует строку из 16-ричного
представления в plain-text
        UploadedFile getFile(std::string name);
        // возвращает свойства файла “name”
        int move_uploaded_file(UploadedFile tmpFile, std::string path);
        // загружает файл “tmpFile” в директорию “path”
        …
   private:
        std::map <std::string, std::string> headers;
        std::map <std::string, std::string> getData;
        std::map <std::string, std::string> postData;
        std::map <std::string, UploadedFile> filesData;
        std::bool isMultipart = false;
        // флаг множественного содержимого
        std::string boundary; // разграничитель

        void checkMultipart();
        // проверяет, является ли запрос multipart/form-data и если да, то заполняет boundary и
isMultipart
        …
}
```

Лабораторная работа №2

- Написать метод rawURLDecode(), который декодирует строку в формате **application/x-www-form-urlencoded** в нормальное представление;
- Вызывать данный метод для всех данных, для которых необходимо преобразование;
- Пояснение: строка %D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D1%82%2C%20%D0%BC%D0%B8%D1%80%21
- Обозначает фразу «Привет, мир!»
- Кириллица занимает 2 байта, символы «,», пробел, и «!» по 1-му байту.

Лабораторная работа №3

- Дополнить библиотеку HTTP.h следующими методами:

```
std::string getHeader(std::string name);
// возвращает значение HTTP-заголовка “name”
std::string rawURLDecode(std::string name);// декодирует строку из 16-ричного представления в plain-text
UploadedFile getFile(std::string name);
// возвращает свойства файла “name”
int move_uploaded_file(UploadedFile tmpFile, std::string path);
// загружает файл “tmpFile” в директорию “path”
```

- Написать скрипт, позволяющий загружать несколько файлов в отдельную директорию на сервере. Выбрать MIME-тип содержимого и максимальный размер.
- Создать скрипт для вывода файлов на экран с возможностью удаления любого файла.
- Предусмотреть защиту от взлома.

Лабораторная работа №3

- В конструкторе HTTP() осуществить парсинг заголовков в зависимости от типа содержимого: application/x-www-form-urlencoded, multipart/form-data, text/plain;
- Информацию о GET и POST-данных хранить в Map'ах "getData" и "postData";
- Информацию о файлах хранить в Map'е "filesData". Содержимое файла необходимо сохранить во временной директории. Сохранять содержимое файла в структуре не нужно;
- Создать свойство headers типа Map в котором будут расположены основные HTTP-заголовки запроса;
- Доступ к заголовкам осуществлять через метод getHeader();
- Реализовать метод getFile(), который будет возвращать структуру типа UploadedFile со свойствами загруженного файла;
- Реализовать функцию move_uploaded_file() для перемещения загруженного файла в нужную директорию.

Перевод систем счисления

```cpp
#include <iostream>     // std::cout, std::dec, std::hex, std::oct

int main () {
    std::string s = "FE";
    unsigned int x = std::stoul(s, nullptr, 16);
    // unsigned long int strtoul (const char* str, char** endptr, int base);
    // str - строка, содержащая представление числа
    // endptr - ссылка на объект, значение которого указывает следующий
    // символ в str после числового значения
    // base - система счисления
    std::cout << x << std::endl; // Выведет 254
    return 0;
}
```