Other languages: 繁體中文

File Handler

File parsing is extracting the file (or files) from a multipart-data POST request to an `HttpFile` object through `MultiPartParser`, here is some information about:

`MultiPartParser` Object

Summary

```
It is the object that you will use to extract and temporarily store the
request files.
```

- `parse(const std::shared_ptr<HttpRequest> &req)`

  Summary

  Receives the request object as a parameter, reads and identifies the files (if heard) and transfers it to the MultiPartParser variable.

  Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);
  }
```

- `getFiles()`

  Summary

  Must be called after `parse()`, returns files of request in the format std::vector&ltHttpFile&gt.

  Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);

      // Check if have files
      if (fileParser.getFiles().empty()) {
          // No files found
      }

      size_t num_of_files = fileParser.getFiles().size();
  }
```

- `getParameters()`

Summary

Must be called after `parse()`, returns the list of other parts from the MultiPartData form.

Returns

std::unordered_map&ltstd::basic_string&ltchar&gt&gt (key, value)

Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);

      if (!fileParser.getFiles().empty()) {
          for (const auto &header : fileParser.getParameters()){
              header.first // Key form
              header.second // Value from key form
          }
      }
  }
```

- `getParameter<Typename T>(const std::string &key)`

  Summary

  Must be called after `parse()`, individual version of getParameters().

  Inputs

  The type of the expected object (will be converted automatically), the key value of the parameter.

  Returns

  The content of the parameter corresponding to the key in the informed format, if it does not exist, will return the default value of the T Object.

  Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);

      std::string email = fileParser.getParameter<std::string>
("email_form");

      // Default type of string is ""
      if (email.empty()) {
          // email_form not found
      }
  }
```

`HttpFile` Object

Summary

It is the object that represents a file in memory, used by `MultiPartParser`.

- `getFileName()`

  Summary

  Self-explanatory name, gets the original name of the file that was received.

Returns

std::string.

Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);

      std::string filename = fileParser.getFiles()[0].getFileName();
  }
```

- `fileLength()`

Summary

Gets the file size.

Returns

`size_t`

Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);

      size_t filesize = fileParser.getFiles()[0].fileLength();
  }
```

- `getFileExtension()`

Summary

Gets the file extension.

Returns

std::string

Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);

      std::string file_extension = fileParser.getFiles()
[0].getFileExtension();
  }
```

- `getMd5()`

Summary

Get MD5 hash of file to check integrity.

Returns

std::string

- `save()`

Summary

Save the file to the file system. The folder saving the file is `UploadPath` configured in config.json (or equivalent). The full path is

```
drogon::app().getUploadPath()+"/"+this->getFileName()
```

Or to simplify, it is saved as: UploadPath/filename

- `save(const std::string &path)`

Summary

Version if parameter is not omitted, uses the &path parameter instead of UploadPath.

Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);

      // Relative path
      fileParser.getFiles()[0].save("./"); // Writes the file to the
same directory on the server, with the original name

      // Absolute path
      fileParser.getFiles()[0].save("/home/user/downloads/"); //
Writes the file in the indicated directory, with the original name
  }
```

- `saveAs(const std::string &path)`

Summary

Writes the file to the path parameter with a new name (ignores the original name).

Example

```
  #include "mycontroller.h"

  using namespace drogon;

  void mycontroller::postfile(const HttpRequestPtr &req,
std::function<void (const HttpResponsePtr &)> &&callback) {
      // Only Post Requests (File Form)

      MultiPartParser fileParser;
      fileParser.parse(req);

      // Relative path
      fileParser.getFiles()[0].saveAs("./image.png"); // Same path of
server
      /* Just example, Don't do this, you would overwrite the file
```

```
format without checking if it really is png */

      // Absolute path
      fileParser.getFiles()[0].save("/home/user/downloads/anyname." +
fileParser.getFiles()[0].getFileExtension());
  }
```

Next: Plugins