Sending Form Data over HTTP

Péter Jeszenszky

October 8, 2025

What is a Form?

- A **form** is an interactive web page component that consists of **form controls** that allow users to enter data.
  - Examples of form controls: text fields, drop-down lists, radio buttons, check boxes
  - Form controls are often called controls for short; however, they are also referred to as widgets.
- The data entered by the user on the form can be submitted to a server for further processing.

Forms in HTML

- The `form` element represents a form in HTML.
- See: `<form>`: The Form element (MDN)

Sending Form Data vs Submitting Form Data

- **Sending form data**: means the process of sending the data over a network protocol such as HTTP.
- **Submitting form data**: refers to an action – typically initiated by the user or a script – that triggers the sending of the data.
    - For example, a user can submit form data by clicking on a *Submit* button.

Using Forms

- Forms can be used to control the behavior of interactive web pages without submitting data.
- Using forms does not require scripts, but an API is available for interacting with forms from scripts.

Form Validation

- HTML allows specific validations on the data entered via form controls to be performed on the client side without requiring scripting.
  - For example, form controls can be marked as required, or the maximum length of the strings entered can be specified for text fields.
- More complex validations can be performed using scripting.

Forms and Form Fields (1)

- Source of terminology:
  - Larry Masinter. Returning Values from Forms: multipart/form-data. RFC 7578, July 2015.
- A **form** is a sequence of **form fields** associated with a name that can get a value when a user fills out the form.
- The result of filling out the form is a sequence of name-value pairs, which are sent when the user submits the form.
- In the name-value pairs, the name is a string, and the value is usually a string, and less often a sequence of bytes.
- The names are not necessarily unique.

Forms and Form Fields (2)

- The terms “form field” and “form control” do not mean the same!
  - A form control is an interactive component of a web page via which a form field can get a value.
  - There is no one-to-one correspondence between form controls and form fields.
    - For example, the same form field can be associated with a group of associated radio buttons.

HTTP Methods for Sending Form Data (1)

In practice, the GET and POST methods are used to submit form data:

- **GET**: the name-value pairs are sent in the query component of the URI separated by & characters, where the characters are escaped with percent encoding if necessary.
- **POST**: the name-value pairs are sent as the content of messages in formats identified by the `application/x-www-form-urlencoded` or `multipart/form-data` media types.

HTTP Methods for Sending Form Data (2)

- Is the ordering of the name-value pairs significant?
  - Although the name-value pairs are sent in a specific order when using both methods, their order is generally not significant.

Sending Form Data with a GET Request (1)

- The primary purpose of the GET method is to retrieve information; it requests the server to transfer a representation of the resource.
- Form data can be sent in GET requests if their purpose is to retrieve information from the server.
- Form data is transmitted in the format identified by the media type `application/x-www-form-urlencoded`.

Sending Form Data with a GET Request (2)

- Search forms represent a typical usage scenario, see for example:
  - Amazon: https://www.amazon.com/s?k=spork
  - Google: https://www.google.com/search?q=http
  - YouTube: https://www.youtube.com/results?search_query=simons%27s+cat

Sending Form Data with a GET Request (3)

Problems:

- Not suitable for sending sensitive information, such as passwords.
- Browsers may limit the maximum length of URIs.
  - See, for example: Chromium Docs: Guidelines for URL Display – URL Length

Sending Form Data with a POST Request (1)

- The POST method requests from the server that the target resource should process the representation included in the request according to its own semantics.
- Thus, the target resource of a POST request is a data handling process.
- Form data can be transmitted in formats identified by the `application/x-www-form-urlencoded` or `multipart/form-data` media types.

Sending Form Data with a POST Request (2)

- Suitable for sending sensitive data, such as passwords.
- A POST request can be used to send the contents of one or more text or binary files.
  - There is no limitation on the size of the files.
- Form data can be used not only for information retrieval, but also for creating new resources.

Formats for Sending Form Data

- `application/x-www-form-urlencoded`:
  - Defined by the URL Standard of the WHATWG: https://url.spec.whatwg.org/
  - It allows form data to be sent either in the query component of URIs or as the content of messages.
- `multipart/form-data`:
  - Defined by RFC 7578:
    - Larry Masinter. Returning Values from Forms: multipart/form-data. RFC 7578, 2015.
  - It can only be used to send form data as the content of messages, typically using POST requests in practice.

Receiving Form Data: httpbin

- First, httpbin.org is a free service for testing HTTP clients.
  - It provides URIs to which developers can send arbitrary HTTP requests, and the server responds with a response that contains metadata about the original request.
  - For example, arbitrary POST requests can be sent to the https://httpbin.org/post resource. The server parses the form data included in the request and returns it.
- Second, the underlying server program is available as free and open source software on GitHub, which allows us to operate our own instance of a httpbin server.
  - Repository: https://github.com/postmanlabs/httpbin
  - Written in: Python
  - License: ISC License

Receiving Form Data: Webhook.site

- First, Webhook.site is an online service for testing HTTP requests: it generates URIs that developers can use as targets for arbitrary HTTP requests.
  - It is a commercial service; all of its functionality is only available to paid users.
  - Some of its features can be used without registration or login; however, with limitations only.
- Second, the underlying server program is available as free and open source software on GitHub, which allows us to operate our own instance of the Webhook.site server.
  - Repository: https://github.com/webhooksite/webhook.site
  - Written in: JavaScript, PHP
  - License: MIT License
  - It lacks some of the commercial features.

Example: GET request with `application/x-www-form-urlencoded` (1)

Command to make a request with HTTPie:

```
http -v https://httpbin.org/get \
name=="University of Debrecen" \
founded==1538 \
url==https://unideb.hu/
```

Example: GET request with application/x-www-form-urlencoded (2)

```
GET /get?name=University+of+Debrecen&founded=1538&url=https%3A%2F%2Funideb.hu%2F HT
Accept: */*
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Host: httpbin.org
User-Agent: HTTPie/3.2.4
```

Example: POST request with application/x-www-form-urlencoded (1)

Command to make a request with HTTPie:

```
http -v --form POST https://httpbin.org/post \
name="University of Debrecen" \
founded=1538 \
url=https://unideb.hu/
```

Example: POST request with application/x-www-form-urlencoded (2)

```
POST /post HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Content-Length: 71
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Host: httpbin.org
User-Agent: HTTPie/3.2.4

name=University+of+Debrecen&founded=1538&url=https%3A%2F%2Funideb.hu%2F
```

Example: POST request with `multipart/form-data` (1)

Command to make a request with HTTPie:

```
http -v --multipart POST https://httpbin.org/post \
name="University of Debrecen" \
founded=1538 \
url=https://unideb.hu/
```

Example: POST request with `multipart/form-data` (2)

```
POST /post HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
Content-Length: 339
Content-Type: multipart/form-data; boundary=3f78585e0c0047adbb245c933ab0bdb0
Host: httpbin.org
User-Agent: HTTPie/3.2.4

--3f78585e0c0047adbb245c933ab0bdb0
Content-Disposition: form-data; name="name"

University of Debrecen
--3f78585e0c0047adbb245c933ab0bdb0
Content-Disposition: form-data; name="founded"

1538
--3f78585e0c0047adbb245c933ab0bdb0
Content-Disposition: form-data; name="url"

https://unideb.hu/
--3f78585e0c0047adbb245c933ab0bdb0--
```