無料電子ブック

学習

HTTP

Free unaffiliated eBook created from
Stack Overflow contributors.

#http

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: http

1: HTTPをいめる

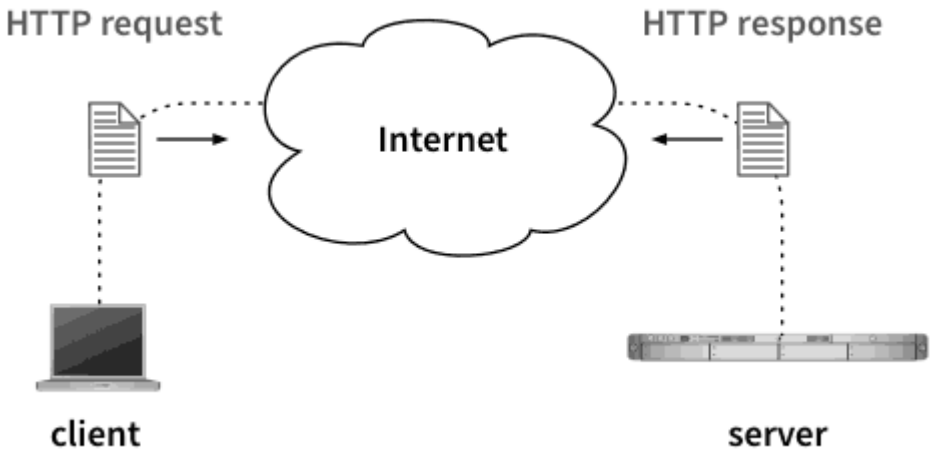
HTTP Hypertext Transfer Protocol は、クライアント/サーバーモデルをします。 HTTPはステートレスなプロトコルです。つまり、のリクエストの、ユーザーのやステータスをするはありません。ただし、パフォーマンスのから、TCPのレイテンシのをするために、、、またはパイプラインなどのをできます。

バージョン

バージョン	ノート	リリース
HTTP / 0.9	「されたまま」	1991-01-01
HTTP / 1.0	HTTP / 1.0ののバージョン、それをRFCにれなかったのバージョン	1992-01-01
HTTP / 1.0 r1	HTTPののRFC	1996-05-01
HTTP / 1.1	の、ベースのホストのサポート	1997-01-01
HTTP / 1.1 r1	あいまいなキーワードのをクリーンアップしました。メッセージフレーミングのがされました。	1999-06-01
HTTP / 1.1 r2	なオーバーホール	2014-06-01
HTTP / 2	HTTP / 2のの	2015-05-01

Examples

HTTPリクエストとレスポンス



HTTPは、WebブラウザなどのHTTPクライアントがHTTPをネットワークでHTTPサーバにし、HTTPサーバからクライアントにHTTPをすをします。

HTTPリクエストは、、ウェブページやイメージなどのオンラインリソースにするであるが、フォームにされたデータなどのもむことができる。 HTTPは、、ウェブページまたはなどのオンラインリソースのである。

HTTP / 1.0

HTTP / 1.0はRFC 1945にされています 。

HTTP / 1.0には、ホストの`Host`ヘッダーなど、、Webでとされるはいくつかあります。

しかし、HTTPクライアントとサーバは、HTTP / 1.1プロトコルのが チャンクエンコーディング やパイプラインなしなどのや、パフォーマンスがパフォーマンスよりもとえられるにHTTP / 1.0をするとしていることがありますローカルプロキシサーバー。

```
GET / HTTP/1.0
User-Agent: example/1

HTTP/1.0 200 OK
Content-Type: text/plain

Hello
```

HTTP / 1.1

HTTP / 1.1は、もともとRFC 2616プロトコルとRFC 2617の1999にされていますが、はされており、としてすべきではありません。

> RFC2616をしないでください。ハードドライブ、ブックマークからし、されたコピーをすべてくまたはをってリサイクルする。
>
> - ノッティンガム、HTTP WG

のHTTPのとするHTTP / 1.1のは、しいRFC 723xにあります

- RFC 7230メッセージのとルーティング
- RFC 7231セマンティクスと
- RFC 7232き
- RFC 7233
- RFC 7234キャッシング
- RFC 7235

HTTP / 1.1がされました。

- サーバーがのサイズのをにすることをにするチャンク、
- なTCP / IPHTTP / 1.0では
- ダウンロードをするためにされる、
- キャッシュ。

HTTP / 1.1では、HTTPクライアントがをたずにのをにすることで、 - のちをすることをにするパイプラインをしようとしました。ながら、このはのプロキシではしくされていないため、パイプラインによってがされたりべえられたりします。

```
GET / HTTP/1.0
User-Agent: example/1
Host: example.com

HTTP/1.0 200 OK
Content-Type: text/plain
Content-Length: 6
Connection: close

Hello
```

HTTP / 2

HTTP / 2 RFC 7540 は、なテキストベースのとヘッダーからフレームでされるバイナリデータにHTTPのオンザフライをしました。 HTTP / 2はヘッダー HPACK のをサポートしています。

これにより、のオーバーヘッドがされ、のTCP / IPでのをにできるようになりました。

データフォーマットのきなにもかかわらず、HTTP / 2はとしてHTTPヘッダーをしており、とはHTTP / 1.1と2のでにできます。

HTTP / 0.9

するHTTPののバージョンは0.9で、しばしば「 HTTP As Implemented 」とばれます 。 0.9のなは、「なHTTP [ie 1.0]プロトコルのサブセット」です。しかし、これは0.9と1.0ののののをすることができません。

0.9のヘッダーでのももありません。は、のCRLFでする`GET`と、スペースと、されたリソースURL

でされます。はのHTMLであるとされます。のわりは、サーバをすることによってマークされる。のまたはをすはありません。インタラクティブなプロパティは、 `<isindex>` HTMLタグににするだけです。

HTTP / 0.9のははではにまれです。 tftpのとしてみみシステムでられます 。

オンラインでHTTPをいめるをむ https://riptutorial.com/ja/http/topic/984/httpをいめる

2: APIのHTTP

HTTP APIはのHTTPをし、はJSONまたはXMLをします。

Examples

リソースをする

リソースのもにしいがであるかについてもがするわけではありません。したがって、APIは`POST`または`PUT`をけれることができます。

リソースがにされた、サーバーは`201 Created`するがあります。もなエラーコードをします。

たとえば、レコードをするAPIをする、/はのようになります。

```
POST /employees HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "name": "Charlie Smith",
    "age": 38,
    "job_title": "Software Developer",
    "salary": 54895.00
}
```

```
HTTP/1.1 201 Created
Location: /employees/1/charlie-smith
Content-Type: application/json

{
    "employee": {
        "name": "Charlie Smith",
        "age": 38,
        "job_title": "Software Developer",
        "salary": 54895.00
        "links": [
            {
                "uri": "/employees/1/charlie-smith",
                "rel": "self",
                "method": "GET"
            },
            {
                "uri": "/employees/1/charlie-smith",
                "rel": "delete",
                "method": "DELETE"
            },
            {
                "uri": "/employees/1/charlie-smith",
                "rel": "edit",
                "method": "PATCH"
            }
        ]
```

```
    },
    "links": [
        {
            "uri": "/employees",
            "rel": "create",
            "method": "POST"
        }
    ]
}
```

レスポンスに`links` JSONをめることで、クライアントはURIやメソッドをにらなくても、しいリソースやアプリケーションにするリソースにアクセスできます。

リソースをする

リソースのやは、APIののです。 `POST` 、 `PUT`または`PATCH`をそれぞれのリソースにすることによって、をうことができます。 `POST`では、リソースののにデータをすることがされていますが、 `PUT`または`PATCH`どちらかをすることをします。これは、よりなをえるためです。

サーバーは、がされたは`200 OK` 、まだされていないは`202 Accepted`するがあります。できないは、もなエラーコードをします。

な

`PUT`は、のをにまれるペイロードにきえるセマンティクスがあります。ペイロードが、すべきリソースののとじでない、サーバはどのアプローチをとるかをすることができる。 RFC7231は、サーバが

- しいメディアタイプをするようにターゲットリソースをする
- しいリソースとしてするに、PUTをリソースのとするにします
- ターゲットリソースがのセットのメディアタイプにされていることをす`415 Unsupported Media Type`レスポンスでをします。

のようなJSON HALをむベースリソース

```
{
    "name": "Charlie Smith",
    "age": 39,
    "job_title": "Software Developer",
    "_links": {
        "self": { "href": "/users/1234" },
        "employee": { "href": "http://www.acmee.com" },
        "curies": [{ "name": "ea", "href": "http://www.acmee.com/docs/rels/{rel}", templated": true}],
        "ea:admin": [
            "href": "/admin/2",
            "title": "Admin"
        ]
    }
}
```

...このようなをけることがあります

```
PUT /users/1234 HTTP/1.1
Host: http://www.acmee.com
Content-Type: "application/json; charset=utf-8"
Content-Length: 85
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

{
    "name": "Charlie Gold-Smith",
    "age": 40,
    "job_title": "Senior Software Developer"
}
```

サーバーはリソースのをされたリクエストにきえ、コンテンツタイプを`application/hal+json`から`application/json`か、JSONペイロードをJSON HALにしてから、リソースのをきえます`415 Unsupported Media Type`をつなメディアタイプのために、されたものとをする。

コンテンツをきえるか、またはをみのモデルにしてから、のコンテンツをみのモデルにきえるかのいがあります。の`GET`は、でのをします。

```
GET /users/1234 HTTP/1.1
Host: http://www.acmee.com
Accept-Encoding: gzip, deflate
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
ETag: "e0023aa4e"

{
    "name": "Charlie Gold-Smith",
    "age": 40,
    "job_title": "Senior Software Developer"
}
```

してからアプローチをすると、のがされます。

```
GET /users/1234 HTTP/1.1
Host: http://www.acmee.com
Accept-Encoding: gzip, deflate
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
ETag: e0023aa4e

{
    "name": "Charlie Gold-Smith",
    "age": 40,
    "job_title": "Senior Software Developer",
    "_links": {
        "self": { "href": "/users/1234" },
        "employee": { "href": "http://www.acmee.com" },
        "curies": [{ "name": "ea", "href": "http://www.acmee.com/docs/rels/{rel}", templated":
true}],
        "ea:admin": [
            "href": "/admin/2",
            "title": "Admin"
        ]
```

```
    }
}
```

PUTは、のとしてされていますが、があることにしてください。これはRFC7231でされています。

> ターゲット・リソースにされるPUTは、のリソースにをもたらすがあります。えば、あるのバージョンのでのバージョンリソースとじをしているなるリソースをするURIとはの「のバージョン」リソースをするためのURIをアーティクルにめることができます。したがって、のバージョンのURIにするPUTリクエストがすると、ターゲットリソースのをするだけでなく、しいバージョンのリソースがされ、リソースにリンクがされるがあります。

のログエントリをすることは、、にリソースのがないため、はとはみなされません。

な

RFC7231ではなにしてこれについてしています

> にされたリソースを、よりきなリソースのとなるでターゲティングするか、なたとえば、 RFC5789でされているPATCHメソッドのためににされたのをして、

したがって、なは2つのでできます。

- あるリソースにのさなサブリソースをめみ、 PUTしてフルリソースのわりにそれぞれのサブリソースのみをさせる
- PATCHをしてサーバーにをするかをする

オーバーラップの

ユーザをのにさせるためにユーザがにされるがある、ユーザをするのではなく、ユーザのにされるをするがある。

のに、ユーザはのをしていた

```
GET /users/1234 HTTP/1.1
Host: http://www.acmee.com
Accept: application/hal+json; charset=utf-8
Accept-Encoding: gzip, deflate
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
ETag: "e0023aa4e"

{
    "name": "Charlie Gold-Smith",
    "age": 40,
    "job_title": "Senior Software Developer",
```

```
    "_links": {
        "self": { "href": "/users/1234" },
        "employee": { "href": "http://www.acmee.com" },
        "curies": [{ "name": "ea", "href": "http://www.acmee.com/docs/rels/{rel}", templated":
true}],
        "ea:admin": [
            "href": "/admin/2",
            "title": "Admin"
        ]
    },
    "_embedded": {
        "ea:address": {
            "street": "Terrace Drive, Central Park",
            "zip": "NY 10024"
            "city": "New York",
            "country": "United States of America",
            "_links": {
                "self": { "href": "/address/abc" },
                "google_maps": { "href": "http://maps.google.com/?ll=40.7739166,-73.970176" }
            }
        }
    }
}
```

ユーザーはしいにしているので、のようなのをします。

```
PUT /address/abc HTTP/1.1
Host: http://www.acmee.com
Content-Type: "application/json; charset=utf-8"
Content-Length: 109
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

{
    "street": "Standford Ave",
    "zip": "CA 94306",
    "city": "Pablo Alto",
    "country": "United States of America"
}
```

したように、のアドレスリソースとのアドレスとののミスマッチメディアタイプのセマンティクスにより、アドレスリソースがされ、ユーザリソースにするのGETリクエストユーザーのしいアドレスがされます。

```
GET /users/1234 HTTP/1.1
Host: http://www.acmee.com
Accept: application/hal+json; charset=utf-8
Accept-Encoding: gzip, deflate
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
ETag: "e0023aa4e"

{
    "name": "Charlie Gold-Smith",
    "age": 40,
    "job_title": "Senior Software Developer",
    "_links": {
        "self": { "href": "/users/1234" },
        "employee": { "href": "http://www.acmee.com" },
```

```
        "curies": [{ "name": "ea", "href": "http://www.acmee.com/docs/rels/{rel}", templated":
true}],
        "ea:admin": [
            "href": "/admin/2",
            "title": "Admin"
        ]
    },
    "_embedded": {
        "ea:address": {
            "street": "Standford Ave",
            "zip": "CA 94306",
            "city": "Pablo Alto",
            "country": "United States of America"
            "_links": {
                "self": { "href": "/address/abc" },
                "google_maps": { "href": "http://maps.google.com/?ll=37.4241311,-122.1524475"
}
            }
        }
    }
}
```

データへのパッチ

PATCHはRFC5789でされており、HTTPのではありません。なは、 にするがあるフィールドのみをすることは 、 PATCHでであるということです。って、は

> PATCHメソッドは、エンティティにされたのがRequest-URIによってされるリソースにされることをします。のセットは、メディアタイプによってされる「パッチドキュメント」とばれるフォーマットでされます。

つまり、クライアントは、リソースをAからBにし、これらのをサーバーにするためになステップをするがあります。

のJSONベースのメディアタイプのパッチはJSONパッチです。

サンプルユーザーのとがされ、ユーザーのをすフィールドをするがある、JSONパッチをしてPATCHをしてにすると、のようになります。

```
PATCH /users/1234 HTTP/1.1
Host: http://www.acmee.com
Content-Type: application/json-patch+json; charset=utf-8
Content-Length: 188
Accept: application/json
If-Match: "e0023aa4e"

[
    { "op": "replace", "path": "/age", "value": 40 },
    { "op": "replace", "path": "/job_title", "value": "Senior Software Developer" },
    { "op": "add", "path": "/salery", "value": 63985.00 }
]
```

PATCHはにのリソースをし、すべてのをするがあるか、APIにトランザクションのをかけることが

ないかのいずれかをするをアトミックにするがあります。

がすると、のようなものがされます

```
HTTP/1.1 200 OK
Location: /users/1234
Content-Type: application/json
ETag: "df00eb258"

{
    "name": "Charlie Smith",
    "age": 40,
    "job_title": "Senior Software Developer",
    "salary": 63985.00
}
```

`200 OK`コードのみにされるものではありません。

ののののフェッチとのでわれた `ETag` 、 `If-Match`または`If-Unmodified-Since`ヘッダーをするがあります。

エラー

`PATCH`のでは、のエラーをしています。

タイプ	エラーコード
なのパッチ	`400 Bad Request`
サポートされていないパッチドキュメント	`415 Unsupported Media Type`
な、つまりパッチをしてresoureがになった	`422 Unprocessable Entity`
リソースがつかりません	`404 Not Found`
、すなわち、しないフィールドのの	`409 Conflict`
する、つまり、クライアントががした`If-Match`または`If-Unmodified-Since`ヘッダーをしている`If-Match`がないは、のエラーコードをすがあります	`412 Precondition Failed` または`409 Conflict`
、すなわち、がにされるがある、さらなる`PATCH`	`409 Conflict`

リソースをする

HTTP APIののなは、のリソースをすることです。これは、、 `DELETE`をしてうがあります。

がした、サーバーは`200 OK`をします。そうでなければなエラーコード。

チャーリー·スミスがしてレコードをしたいは、のようになります。

```
DELETE /employees/1/charlie-smith HTTP/1.1
Host: example.com
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'links': [
        {
            'uri': '/employees',
            'rel': 'create',
            'method': 'POST'
        }
    ]
}
```

リソースをする

HTTP APIののなは、サーバーののリソースのリストをすることです。このようなリストは、 `GET` リクエストをしてするがあります`GET`リクエストはデータをするためです。

サーバーは、リストをできるは`200 OK`、そうでないはなエラーコードをすがあります。

をリストすると、のようになります。

```
GET /employees HTTP/1.1
Host: example.com
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'employees': [
        {
            'name': 'Charlie Smith',
            'age': 39,
            'job_title': 'Software Developer',
            'salary': 63985.00
            'links': [
                {
                    'uri': '/employees/1/charlie-smith',
                    'rel': 'self',
                    'method': 'GET'
                },
                {
                    'uri': '/employees/1/charlie-smith',
                    'rel': 'delete',
                    'method': 'DELETE'
                },
                {
                    'uri': '/employees/1/charlie-smith',
                    'rel': 'edit',
                    'method': 'PATCH'
```

```
                }
            ]
        },
        {
            'name': 'Donna Prima',
            'age': 30,
            'job_title': 'QA Tester',
            'salary': 77095.00
            'links': [
                {
                    'uri': '/employees/2/donna-prima',
                    'rel': 'self',
                    'method': 'GET'
                },
                {
                    'uri': '/employees/2/donna-prima',
                    'rel': 'delete',
                    'method': 'DELETE'
                },
                {
                    'uri': '/employees/2/donna-prima',
                    'rel': 'edit',
                    'method': 'PATCH'
                }
            ]
        }
    ],
    'links': [
        {
            'uri': '/employees/new',
            'rel': 'create',
            'method': 'PUT'
        }
    ]
}
```

オンラインでAPIのHTTPをむ https://riptutorial.com/ja/http/topic/3423/apiのhttp

3: HTTPステータスコード

き

HTTPでは、ステータスコードは、にされたのをするのメカニズムです。 RFC 7231、から。 6 "status-codeは、をしてさせるためのみのをえる3のコードです。"

なは、コードがにもすることができるようになります000と999 。しかし、からののみ100への599がりてられています。

HTTP / 1.1では、レスポンスでクライアントがをすべきかをするために、ステータスHTTPレスポンスののにされるのHTTPステータスコードがいくつかされています。

ステータスコードののはのクラスをします。

- 1xx
- 2xx クライアントがしました
- 3xx Request redirected - しいリクエストなどのアクションが
- 4xx クライアントエラー - じリクエストをりさない
- 5xx サーバーエラー - するがありまず

には、もなステータスコードをすることはずしもではありません。

Examples

500サーバーエラー

HTTP 500サーバーエラーは、サーバーがせぬものにしたというなメッセージです。アプリケーションまたはなWebサーバーは、のにエラーがしたに500をするがあります。つまり、がスローされた、またはリソースのによってがしないがあります。

ステータスの

```
HTTP/1.1 500 Internal Server Error
```

404おしのページがつかりませんでした

「HTTP 404 Not Found」は、クライアントがしたURIをしてサーバーがパスをできなかったことをします。

```
HTTP/1.1 404 Not Found
```

ほとんどの、されたファイルはされましたが、ドキュメントルートのやアクセスのなどがありま

すただし、しているアクセスはHTTP 403をにトリガーします。

たとえば、MicrosoftのIISは、されたファイルがされたときに、ログファイルに404.00はサブですをきみます。しかし、がフィルタリングによってブロックされると、をブロックするにってログファイルに404.5-404.19をきみます。なエラーコードのは、 マイクロソフトサポートにあります。

されたファイルへのアクセスをする

クライアントがのアクセスのためにアクセスできないリソースをしたは、403 Forbiddenをします。たとえば、をつユーザーのみがアクセスできる`/admin`ルートがアプリケーションにある、のユーザーがページをリクエストすると、403をできます。

```
GET /admin HTTP/1.1
Host: example.com
```

```
HTTP/1.1 403 Forbidden
```

リクエストの

がしたことをすステータスコード`200` HTTPをします。 HTTPステータスはのようになります。

```
HTTP/1.1 200 OK
```

ステータステキスト`OK`はです。メッセージペイロードには、されたリソースのがまれているがあります。201がないは、コンテンツをしないでください。

キャッシュされたコンテンツにするきへの

`If-Modified-Since`と`If-None-Match`ヘッダーがまれているクライアントにして、リソースがされていないは、サーバーからされた**304 Not Modified**ステータスをします。

たとえば、Webページのクライアントリクエストにヘッダーがまれている`If-Modified-Since: Fri, 22 Jul 2016 14:34:40 GMT`、それページがされていないは、 `HTTP/1.1 304 Not Modified`ステータスラインでし`HTTP/1.1 304 Not Modified`。

10のHTTPステータスコード

2xxの

- **200 OK** - なHTTPにするな。
- **201 Created** - リクエストがし、しいリソースがされました。
- **204 No Content** - サーバーはをにしましたが、コンテンツはされません。

3xxリダイレクション

- **304 Not Modified** - ヘッダー`If-Modified-Since`または`If-None-Match`されたバージョンにリソースがされていないことをします。

4xxクライアントエラー

- **400 Bad Request** - サーバは、らかなクライアントエラーなのリクエスト、サイズがきすぎる、なリクエストメッセージフレーミング、なリクエストルーティングなどによりリクエストをできない、またはしない。
- **401 Unauthorized** - *403 Forbidden*にしていますが、にがで、したか、まだされていないにします。レスポンスには、されたリソースになチャレンジをむ`WWW-Authenticate`ヘッダーフィールドがまれているがあります。
- **403** - はなでしたが、サーバーはをしています。ユーザーはログインしているがありますが、リソースになはありません。
- **404 Not Found** - されたリソースはつかりませんでしたが、になるがあります。クライアントによるのはされます。
- **409 Conflict** - ののののなど、ののためにをできなかったことをします。

5xxサーバーエラー

- **500**サーバーエラー - しないにし、さらにのメッセージがでないにされるなエラーメッセージ。

オンラインでHTTPステータスコードをむ https://riptutorial.com/ja/http/topic/2577/httpステータスコード

4: HTTPリクエスト

パラメーター

HTTPメソッド	
OPTIONS	されたURIでなオプションなメソッドとヘッダーにするをします。
GET	URIによってされるデータ、またはURIでなスクリプトによってされたデータをします。
HEAD	GETとじですが、サーバーからされるメッセージはなく、ヘッダーのみです。
POST	リクエストURIでされたリソースにするために、メッセージでされたデータブロックをサーバにします。フォームにもにされます。
PUT	まれたをメッセージにされたURIのにしいリソースまたはされたリソースとしてします。
DELETE	URIによってされるリソースをするか、またはちにする。
TRACE	にechoコマンドしているのHTTPサーバーは、200OKのとしてをりすがあります。

CONNECTメソッドは、プロキシ·モードとトンネリング·モードSSLトンネリングなどをりえることができるプロキシでするためのメソッドによってされています。

Examples

Telnetをしてでの HTTPをする

このでは、HTTPはテキストベースのインターネットプロトコルであり、なHTTPとするHTTPをしています。

Telnetをして、のようにのHTTPをコマンドラインからでできます。

1. ポート80のWebサーバwww.example.orgへのTelnetセッションをします。

   ```
   telnet www.example.org 80
   ```

   Telnetは、サーバーにしたことをします。

```
Connected to www.example.org.
Escape character is '^]'.
```

2. GETリクエストのURLパスライン/ HTTP 1.1をして、

```
GET / HTTP/1.1
```

3. なURLのホストをするHTTPヘッダーフィールドをします。これはHTTP 1.1でです

```
Host: www.example.org
```

4. をしてをします。

   WebサーバーはTelnetセッションにされるHTTPをします。

なセッションはのりです。ののは*HTTP*ステータスで、ステータスコード200とテキストがにされたことをすステータステキスト*OK*がまれています。これには、いくつかのHTTPヘッダーフィールド、、およびHTMLがきます。

```
$ telnet www.example.org 80
Trying 2606:2800:220:1:248:1893:25c8:1946...
Connected to www.example.org.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.example.org

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Thu, 21 Jul 2016 15:56:05 GMT
Etag: "359670651"
Expires: Thu, 28 Jul 2016 15:56:05 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (lga/1318)
Vary: Accept-Encoding
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270

<!doctype html>
<html>
<head>
    <title>Example Domain</title>
    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
</head>
<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is established to be used for illustrative examples in documents. You may
use this
    domain in examples without prior coordination or asking for permission.</p>
    <p><a href="http://www.iana.org/domains/example">More information...</a></p>
```

```
</div>
</body>
</html>
```

にするために、 `style`とをHTMLからしました。

HTTP 1.1では、のHTTPはと`Host`ヘッダーでされます。

```
GET /search HTTP/1.1 \r\n
Host: google.com \r\n
\r\n
```

ののはのとおりです。

```
Method Request-URI HTTP-Version CRLF
```

`Method`はなHTTP `Method`なければなりません。 [1] [2]のいずれか

- `OPTIONS`
- `GET`
- `HEAD`
- `POST`
- `PUT`
- `DELETE`
- `PATCH`
- `TRACE`
- `CONNECT`

`Request-URI`は、URIまたはクライアントがしているリソースへのパスをします。のいずれかになります。

- スキーマ、ホスト、オプションのポートとパスをむURI。または
- パスをします。この、ホストは`Host`ヘッダーでするがあります

`HTTP-Version`は、クライアントがしているHTTPプロトコルのバージョンをします。 HTTP 1.1の、これはに`HTTP/1.1`なければなりません。

は、 `\r\n`されるフィードでわります。

リクエストヘッダフィールド

ヘッダーフィールドはに「ヘッダー」とばれるをHTTPにして、にするをすることができます。ヘッダは、そのようなことをサポートするプログラミングのメソッドにされるパラメータとのセマンティクスをっています。

`Host` 、 `User-Agent` 、および`Referer`ヘッダーのはのようになります。

```
GET /search HTTP/1.1 \r\n
Host: google.com \r\n
```

```
User-Agent: Chrome/54.0.2803.1 \r\n
Referer: http://google.com/ \r\n
\r\n
```

サポートされているHTTP 1.1ヘッダーのなリストは、にされています 。もなのはのとおりです。

- `Host` - リクエストURLのホストHTTP / 1.1で
- `User-Agent` - しているユーザエージェントをす。
- `Referer` - ここでクライアントがされたURI。そして
- `If-Modified-Since` - サーバーがリソースがされたかどうかをし、クライアントがキャッシュされていないコピーをできることをすためにサーバーができるをします。

ヘッダーは、 `Name: Value CRLF`としてするがあります。 `Name`は、 `User-Agent`などのヘッダーです。 `Value`はそれにりてられたデータであり、はCRLFでわるがあります。ヘッダはとをしないとのみ、、やをすることができます`!#$%&'*+-.^_`|~` RFC7230セクション3.2.6フィールド 。

`Referer`ヘッダーフィールドはRFC1945でってされた ' referrer 'のタイプミスです。

メッセージ

のHTTPリクエストにはメッセージがまれているがあります。これは、サーバーがをするためにするデータです。メッセージは、サーバーがリソースにするしいデータをするために、POSTまたはPATCHおよびPUTでもよくされます。

メッセージをむは、さをに`Content-Length`ヘッダーとともにバイトでめるがあります。

すべてのヘッダーのにメッセージがまれ、 CRLFが2つまれます。をむPUTリクエストのは、のようになります。

```
PUT /files/129742 HTTP/1.1\r\n
Host: example.com\r\n
User-Agent: Chrome/54.0.2803.1\r\n
Content-Length: 202\r\n
\r\n
This is a message body. All content in this message body should be stored under the
/files/129742 path, as specified by the PUT specification. The message body does
not have to be terminated with CRLF.
```

`HEAD`と`TRACE`にはメッセージをめないでください。

オンラインでHTTPリクエストをむ https://riptutorial.com/ja/http/topic/2909/httpリクエスト

5: HTTPレスポンス

パラメーター

ステータスコード	フレーズ -
100	- クライアントは、のからなるののをするがあります。
101	スイッチングプロトコル - このでされるプロトコルのバージョンまたはタイプがサーバーによってされています。
200	**OK** - サーバーはクライアントのをしてしました。
201	**Created** - サーバーはをけれ、しいリソースをしました。これは`Location`ヘッダーのURIのでできます。
202	**Accepted** - サーバーはクライアントのをしてけれましたが、まだをまたはしていません。
203	のない - サーバーはのサーバーでなのサブセットまたはスーパーセットであるのあるデータをしています。にプロキシによってされます。
204	**No Content** - レスポンスのボディがないに200OKのわりにされます。
205	コンテンツのリセット - 204No Contentとじですが、クライアントはアクティブなドキュメントビューをみみするがあります。
206	な - クライアントが`Range`ヘッダーをしたときに200OKのわりにされます。
300	の - されたリソースはのURIでであり、クライアントはをメッセージのリストにされたURIにリダイレクトするがあります。
301	**Permanently Moved** - されたリソースはこのURIでできなくなり、クライアントはこれとのすべてのを`Location`ヘッダーでされたURIにリダイレクトするがあります。
302	**Found** - リソースはにのURIのにあります。このリクエストは、 `Location`ヘッダーのURIへのユーザーののにリダイレクトするがありますが、のリクエストはしないでください。
303	その - 302Foundとよくていますが、されたURIにリダイレクトするためのユーザーはありません。されたURIは、GETでするがあります。
304	されていない - クライアントは`If-Modified-Since`またはのヘッダーをしました。

ステータスコード	フレーズ -
	その、リソースはされていません。クライアントはリソースのキャッシュされたコピーをするがあります。
305	プロキシをする - されたリソースは、 `Location`ヘッダーフィールドでされたプロキシをしてされなければなりません。
307	リダイレクト - 302Foundとじですが、HTTP 1.0クライアントは307をサポートしていません。
400	なリクエスト - クライアントはエラーをむなリクエストをしましたが、リクエストをしてそれをしてからそれをりすがあります。
401	**Unauthorized** - されたリソースはなしでできません。クライアントは、のをするために`Authorization`ヘッダーをしてをりすがあります。
402	い - コンテンツをするためにユーザーのをとするアプリケーションでするためのみ、のステータスコード。
403	されている - サーバーはをしますが、のアクセスのためにをすることをします。このはりさないでください。
404	つからない - されたURIとするこのサーバーでなリソースがありません 。アクセスのがされないように、403のわりにできます。
405	メソッドがされていません - リソースがメソッドHTTPをサポートしていません。 `Allow`ヘッダーにはけれなメソッドがされます。
406	**Not Acceptable** - リソースには、でされたけれヘッダーにするがあります。
407	プロキシが - 401Unauthorizedとていますが、クライアントがにプロキシでするがあることをします。
408	**Request Timeout** - サーバーはクライアントからののをしていましたが、けれなにはもされませんでした。
409	**Conflict** - リソースののとしているため、をできませんでした。
410	**Gone** - 404Not Foundとですが、なをします。アドレスはできません。
411	**Length Required** - クライアントはな`Content-Length`ヘッダーをしておらず、サーバーがこのをけれるにそれをわなければなりません。
412	**Precondition Failed** - クライアントがしたきヘッダーでされたすべてのでリソースをできません。

ステータスコード	フレーズ -
413	エンティティがきすぎる - サーバーは、クライアントがしたさのメッセージをできません。
414	**Request-URI**がすぎる - Request-URIがサーバーがしようとするよりもいため、サーバーはをしています。
415	サポートされていないメディアタイプ - サーバーは、クライアントによってされたMIMEまたはメディアタイプをサポートしておらず、このをできません。
416	**Requested Range Not Satisfiable** - クライアントがバイトをしましたが、サーバーはそのにコンテンツをできません。
417	**Expectation Failed** - クライアントは、サーバーがたすことができない`Expect`ヘッダーのをしました。
500	サーバーエラー - しないまたはエラーがしたため、サーバーはこのをできませんでした。
501	されていない - サーバーはをするためになをサポートしていません。、どのリソースでもサポートされていないメソッドをすためにされます。
502	**Bad Gateway** - サーバーがプロキシであり、こののにアップストリームサーバーからなをしました。
503	**Service Unavailable** - サーバーがまたはメンテナンスで、このをするがありません。
504	ゲートウェイタイムアウト - サーバーはプロキシであり、タイムリーにアップストリームサーバーからのをしませんでした。
505	サポートされていない**HTTP**バージョン - サーバーは、クライアントがしたHTTPプロトコルのバージョンをサポートしていません。

Examples

な

HTTPサーバーがのHTTPをすると、にまれるをしてクライアントにをさなければなりません。なHTTP 1.1レスポンスは、のようにえるかもしれません。くの、いくつかのヘッダーフィールドができます。

```
HTTP/1.1 200 OK \r\n
```

```
HTTP/1.1 404 Not Found \r\n
```

```
HTTP/1.1 503 Service Unavailable \r\n
```

シンプルなHTTP 1.1レスポンスはのをとります

```
HTTP-Version Status-Code Reason-Phrase CRLF
```

とに、 `HTTP-Version`はのHTTPプロトコルのバージョンをします。 HTTP 1.1ではに`HTTP/1.1`というでなければなりません。

`Status-Code`は、クライアントののステータスをす3のコードです。このコードののはステータスクラスで、ステータスコードを5つのカテゴリのレスポンス[1]のいずれかにします。

- `1xx` - サーバーがをし、がです。
- `2xx` - サーバーがをけれてしました
- `3xx` リダイレクション - をするためにクライアントでさらにアクションが
- `4xx` **Client Errors** - クライアントがなのをしたか、またはできませんでした
- `5xx` サーバーエラー - はでしたが、サーバーはそれをできません

`Reason-Phrase`は、ステータスコードのなです。たとえば、コード`200`は`OK`フレーズがあります。コード`404`は、 `Not Found`フレーズをする。フレーズのなリストは、パラメータ、、またはHTTPでできます。

このははは`\r\n`されるキャリッジリターンラインフィードのペアでわります。

ヘッダー

HTTPリクエストとに、HTTPレスポンスには、レスポンスをまたはするためのヘッダーがまれているがあります。

なヘッダーのなリストは、の§6.2でされています 。もにされるヘッダーはのとおりです。

- `Server` `User-Agent`ヘッダーのようにします。
- `Location` リダイレクトのURIをす201および3xxのステータスでされます。そして
- `ETag`は、クライアントがをキャッシュできるように、このバージョンのされたリソースののです。

レスポンスヘッダーはステータスのにて、 リクエストヘッダーはのようにされます。

```
Name: Value CRLF
```

`Name`は、 `ETag`や`Location`などのヘッダー`Name`し、 `Value`は、サーバーがそのヘッダーにするをします。はCRLFでわります。

ヘッダーのはのようになります。

```
HTTP/1.1 201 Created \r\n
Server: WEBrick/1.3.1 \r\n
Location: http://example.com/files/129742 \r\n
```

メッセージ

とに、HTTPにはメッセージがまれているがあります。これにより、クライアントがするデータがされます。に、のGETにする200 OKは、にされたデータをむメッセージをするがあります。しない、204 No Contentはよりなです。

すべてのヘッダーのにメッセージがまれ、CRLFが2つきます。にしては、さはバイトで`Content-Length`ヘッダーでするがあります。したがって、GETにするなは、のようになります。

```
HTTP/1.1 200 OK\r\n
Server: WEBrick/1.3.1\r\n
Content-Length: 39\r\n
ETag: 4f7e2ed02b836f60716a7a3227e2b5bda7ee12c53be282a5459d7851c2b4fdfd\r\n
\r\n
Nobody expects the Spanish Inquisition.
```

オンラインでHTTPレスポンスをむ https://riptutorial.com/ja/http/topic/3077/httpレスポンス

6: HTTPレスポンスのキャッシュ

レスポンスは、URLとHTTPメソッドごとににキャッシュされます。

HTTPキャッシングはRFC 7234でされています 。

- まだれではないキャッシュされたのな。、なレスポンスは、レスポンスがされたサーバにするなしにリクエストをたすことができます。
- - をぎたキャッシュの。は、なをして、サーバーがまだかどうかをせずにをたすことはできません。
- -キャッシュされたは、のすべてのがキャッシュされたレスポンスとするをたし 、えば、それらがじHTTPメソッドとURLをっている、がであるか、がくなったレスポンスをにする、リクエストヘッダはのでされたヘッダがし`Vary`など、ヘッダーを。
- - キャッシュされたがであるかどうかをチェックする。これは、、 `If-Modified-Since`または`If-None-Match`およびステータス`304`をむきでわれます。
- のユーザのためのプライベートキャッシュ - キャッシュ、えばウェブブラウザ。プライベートキャッシュには、パーソナライズされたをできます。
- パブリックキャッシュ - くのユーザーでされるキャッシュプロキシサーバーなどこのようなキャッシュは、のユーザーにじをできます。

Examples

1のキャッシュ

```
Cache-Control: public, max-age=31536000
```

`public`は、がすべてのユーザーにしてじであることをしますされたはまれません。 `max-age`はからです。 31536000 = 60 * 60 * 24 * 365。

これはしてするつもりのないにおめします。

パーソナライズされたを1キャッシュする

```
Cache-Control: private, max-age=60
```

`private`は、リソースをしたユーザーにしてのみをキャッシュできることをし、のユーザーがじリソースをしたときにはできません。これは、Cookieにするレスポンスにしています。

にサーバーでせずにキャッシュされたリソースのをする

```
Cache-Control: no-cache
```

クライアントは、がキャッシュされていないかのようにします。これは、いつでもせずされるのあるリソースや、のバージョンでにされるのあるリソースにしています。

`no-cache`するレスポンスは、するたびにサーバーにするがあるため、がくなりますちがくなります。

ただし、をするために、クライアントはとしてそのようなをすることがあります。 `no-cache`レスポンスは、キャッシュされたレスポンスをできるかどうかをチェックするたびに、サーバにすることなくリクエストをたすためにされません。

レスポンスをまったくしないようする

```
Cache-control: no-store
```

クライアントnoにをキャッシュし、できるだけくをれるようにします。

このディレクティブはもともとのいデータにされていましたがはHTTPSをわりにするがあります、できないレスポンスをつキャッシュのをけるためにできます。

レスポンスデータがになるのケース、たとえばきなをすAPIエンドポイントでのみです。さもなければ、 `no-cache`およびrevalidationは "uncacheable"のいをっているが、まだいくらかのをすることができる。

された、でなヘッダー

- `Expires` - リソースがくなったをします。なクロックとタイムゾーンをサポートしているサーバーやクライアントにしています。 `Cache-control: max-age` `Expires`よりも`Cache-control: max-age`がされ、にがします。
- `post-check`と`pre-check`は、くなったレスポンスをできるのInternet Explorerです。なは、`stale-while-revalidate`です。
- `Pragma: no-cache` - 1999にされました。わりに`Cache-control`を`Cache-control`があります。

キャッシュされたリソースの

キャッシュをバイパスするもなは、URLをすることです。これは、URLにリソースのバージョンまたはチェックサムがまれているのベストプラクティスとしてされます。

```
http://example.com/image.png?version=1
http://example.com/image.png?version=2
```

これらの2つのURLは々にキャッシュされるので、 `…?version=1`がにキャッシュされたとしても、しいコピーはすぐに`…?version=2`としてりすことができます。

ランダムURLをしてキャッシュをバイパスしないでください。 `Cache-control: no-cache`または

`Cache-control: no-store`します。 `no-store`ディレクティブをせずにランダムなURLのをすると、キャッシュにながされ、キャッシュからよりながりされ、キャッシュのパフォーマンスがします。

オンラインでHTTPレスポンスのキャッシュをむ https://riptutorial.com/ja/http/topic/3296/httpレスポンスのキャッシュ

7: クロスオリジンとアクセス

クロスオリジンリソースは、 AJAXなどのをして、ドメインでなをうことができるようにされています 。スクリプティングはほとんどのをいますが、HTTPサーバーはしいヘッダーをしてをサポートするがあります。

Examples

クライアントクロスオリジンリソースCORSをする

`Origin`ヘッダーをめ、 `Origin`をえたをするがあります。これは、リクエストのをします。たとえば、からのクロスオリジン·リクエスト`http://example.com`に`http://example.org`のようになります。

```
GET /cors HTTP/1.1
Host: example.org
Origin: example.com
```

サーバーはこのをして、がされているかどうかをします。

サーバーCORSへの

CORSへのには、CORSリソースのをされている`Access-Control-Allow-Origin`す`Access-Control-Allow-Origin`ヘッダーがまれているがあります。このヘッダーは、の3つののいずれかをとります。

- 。これをうと、 そのからののみがされます。
- キャラクター`*` 。これにより、 のからのがされます。
- `null`これにより、CORSはされません 。

たとえば、`http://example.com`からのCORSのに、 `example.com`がされたである、サーバーはのをします。

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: example.com
```

ののもこのをする、すなわち

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
```

ユーザーのまたはセッションをする

ユーザーのまたはユーザーのセッションをCORSとともにできるようにすることで、サーバーはCORSをしてユーザーデータをできます。これは、データをするにユーザーがログインしている

かどうかをサーバーがするがあるにですユーザーがログインしているにのみアクションをします。これにより、とともにCORSをするがあります。

これは、 `OPTIONS`プリフライトにして`Access-Control-Allow-Credentials`ヘッダーをすることによって、プリフライトされたのサーバーでできます。リソースを`DELETE`するCORSののをします。

```
OPTIONS /cors HTTP/1.1
Host: example.com
Origin: example.org
Access-Control-Request-Method: DELETE
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: example.org
Access-Control-Allow-Methods: DELETE
Access-Control-Allow-Credentials: true
```

`Access-Control-Allow-Credentials: true`は、の`DELETE` CORSがユーザーのとともにされることをします。

プリフライトリクエスト

なCORSでは、の2つののいずれかをできます。

- する
-

ほんののヘッダーがあります。 POST CORSでは、さらに3つのコンテンツタイプのみをできます。

このをするために、のメソッド、ヘッダー、またはコンテンツタイプをするは、まずアクセスヘッダーをむ`OPTIONS`であるプリフライトをするがあります。たとえば、サーバーが`DNT`ヘッダーをむ`PUT`をけれるかどうかをするプリフライトです。

```
OPTIONS /cors HTTP/1.1
Host: example.com
Origin: example.org
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: DNT
```

サーバープリフライトリクエストにする

サーバーがプリフライトをすると、されたメソッドとヘッダーをサポートしているかどうかをチェックし、をサポートするをすと、そののされたデータなどをすがあります。

これらは、アクセスのヘッダーのでされます。サーバはまた、プリフライトがキャッシュされるをすアクセス`Max-Age`ヘッダをすることができる。

これは、プリフライトリクエストのリクエスト/レスポンスサイクルがのようなものになります。

```
OPTIONS /cors HHTP/1.1
Host: example.com
Origin: example.org
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: DNT
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: example.org
Access-Control-Allow-Methods: PUT
Access-Control-Allow-Headers: DNT
```

オンラインでクロスオリジンとアクセスをむ https://riptutorial.com/ja/http/topic/3424/クロスオリジンとアクセス

8: レスポンスのエンコーディングと

Examples

HTTP

HTTPメッセージはできますHTTP / 1.1。サーバーがをし、`Content-Encoding`ヘッダーをするか、またはプロキシが`Transfer-Encoding`ヘッダーをしてします。

クライアントは`Accept-Encoding`ヘッダーをして、どのエンコーディングをけれるかをすことができます。

もにされるエンコーディングはのとおりです。

- "gzip"ファイルのプログラム RFC1952 でCRC32チェックサムをしたgzip - deflateアルゴリズムLZ77
- deflate - "zlib"データ RFC1950 、deflateアルゴリズムハイブリッドLZ77とハフマンとAdler32チェックサム

の

HTTPメッセージをすることはです。エンコーディングは、されたにカンマでります。たとえば、メッセージがdeflateでされてからgzipでされた、ヘッダーはのようになります。

```
Content-Encoding: deflate, gzip
```

の`Content-Encoding`ヘッダーもですが、されません。

```
Content-Encoding: deflate
Content-Encoding: gzip
```

gzip

クライアントはまず、gzipをサポートしていることをす`Accept-Encoding`ヘッダーでをします。

```
GET / HTTP/1.1\r\n
Host: www.google.com\r\n
Accept-Encoding: gzip, deflate\r\n
\r\n
```

サーバーはされたと、gzipエンコーディングがされたことをする`Content-Encoding`ヘッダーをむをすることができます。

```
HTTP/1.1 200 OK\r\n
Content-Encoding: gzip\r\n
```

```
Content-Length: XX\r\n
\r\n
... compressed content ...
```

オンラインでレスポンスのエンコーディングとをむ https://riptutorial.com/ja/http/topic/5046/レスポンスのエンコーディングと

9:

パラメーター

パラメータ	
ステータス	`401`サーバーがをするは`401`、プロキシがをするは`407`
ヘッダー	`WWW-Authenticate`オリジンサーバーによる`WWW-Authenticate`、`Proxy-Authenticate`プロキシによる`Proxy-Authenticate`
ヘッダーをする	`Authorization`サーバーにする`Proxy-Authorization`、プロキシにする`Proxy-Authorization`
	`Basic`には`Basic`ですが、`Digest`や`SPNEGO`などののものもできます。HTTPのレジストリをしてください。
レルム	サーバーのされたスペースの。サーバーはのそのようなスペースをつことができ、それぞれにのとメカニズムがあります。
	`Basic`ユーザーとパスワードはコロンでられ、base64でエンコードされています。たとえば、`username:password` base64-encodedは`dXNlcm5hbWU6cGFzc3dvcmQ=`

はRFC2617でされています。これは、`407 (Proxy Authentication Required)`に`401 Unauthorized`とプロキシサーバーをしたに、オリジンサーバーにしてするためにできます。デコードされたクレデンシャルでは、のコロンのにパスワードがまります。したがって、ユーザにはコロンをすることはできませんが、パスワードはできます。

Examples

HTTP

HTTPは、のためのなメカニズムをします。クレデンシャルはプレーンテキストでされるため、デフォルトではではありません。したはのようにします。

クライアントは、アクセスがされているページをします。

```
GET /secret
```

サーバーはコード`401 Unauthorized`し、クライアントにをします。

```
401 Unauthorized
WWW-Authenticate: Basic realm="Secret Page"
```

クライアントは`Authorization`ヘッダーをします。は`username:password` base64 encodedです

```
GET /secret
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

サーバーはをけれ、ページのでします。

```
HTTP/1.1 200 OK
```

オンラインでをむ https://riptutorial.com/ja/http/topic/3286/

クレジット

S. No		Contributors
1	HTTPをいめる	Community, DaSourcerer, Kornel, Peter Hilton
2	APIのHTTP	ArtOfCode, mnoronha, Peter Hilton, Roman Vottner
3	HTTPステータスコード	ArtOfCode, DaSourcerer, Deltik, Kornel, Lex Li, mnoronha, Peter Hilton, Rptk99, Sender, Xevaquor
4	HTTPリクエスト	artem, ArtOfCode, Jeff Bencteux, Peter Hilton
5	HTTPレスポンス	ArtOfCode, Jeff Bencteux, Peter Hilton
6	HTTPレスポンスのキャッシュ	DaSourcerer, Kornel
7	クロスオリジンとアクセス	ArtOfCode
8	レスポンスのエンコーディングと	Jeff Bencteux, Peter Hilton
9		DaSourcerer, Peter Hilton, Stefan Kögl