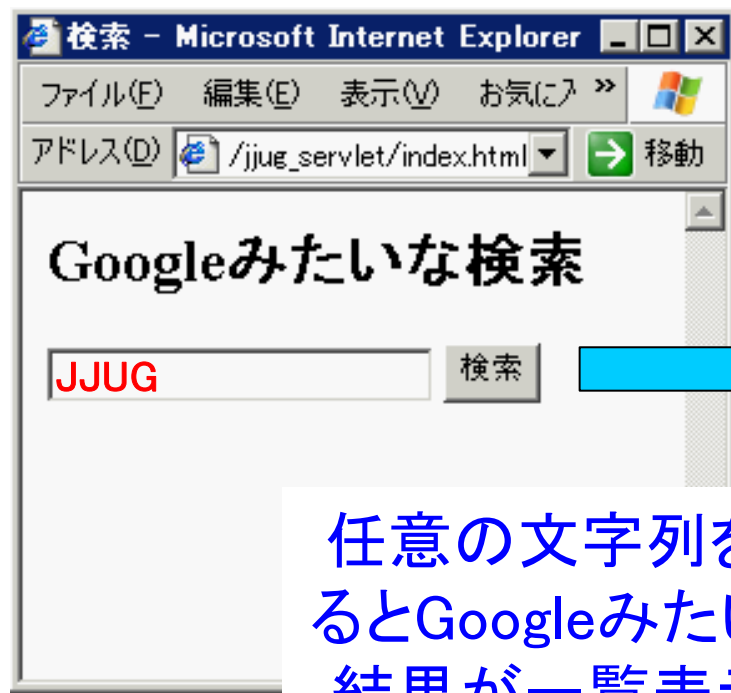

基礎セミナー 初級者シリーズ HTTPとサーブレット

日本Javaユーザグループ／株式会社アットウェア
牧野隆志

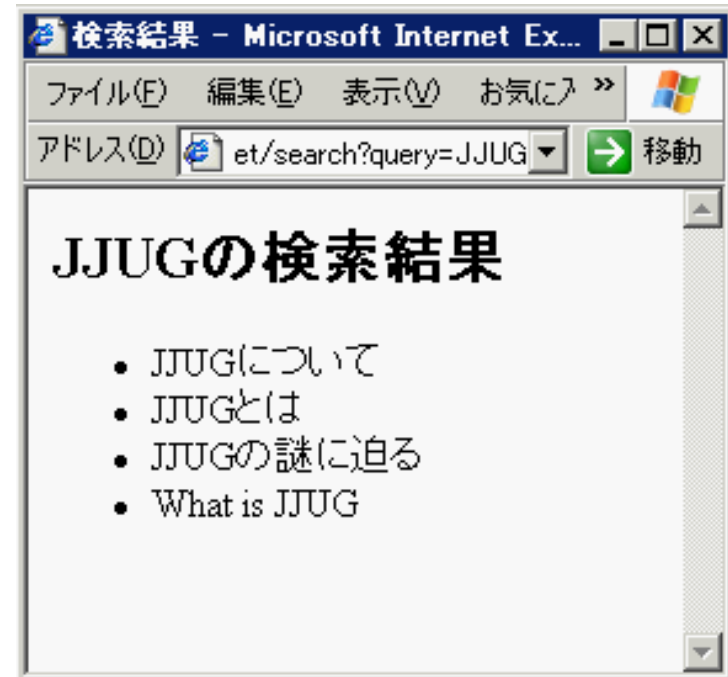
目的

- サーブレットが動作する仕組みを学びます
 - 深く詳細まででは取り上げません
 - APIドキュメントを読んだり、書籍、Webサイトなどを参照してください
 - ミドルウェアやWebアプリケーションフレームワークが何をやっているかなんとなく想像がつくようになるとうれしいです

こういうのを作ります

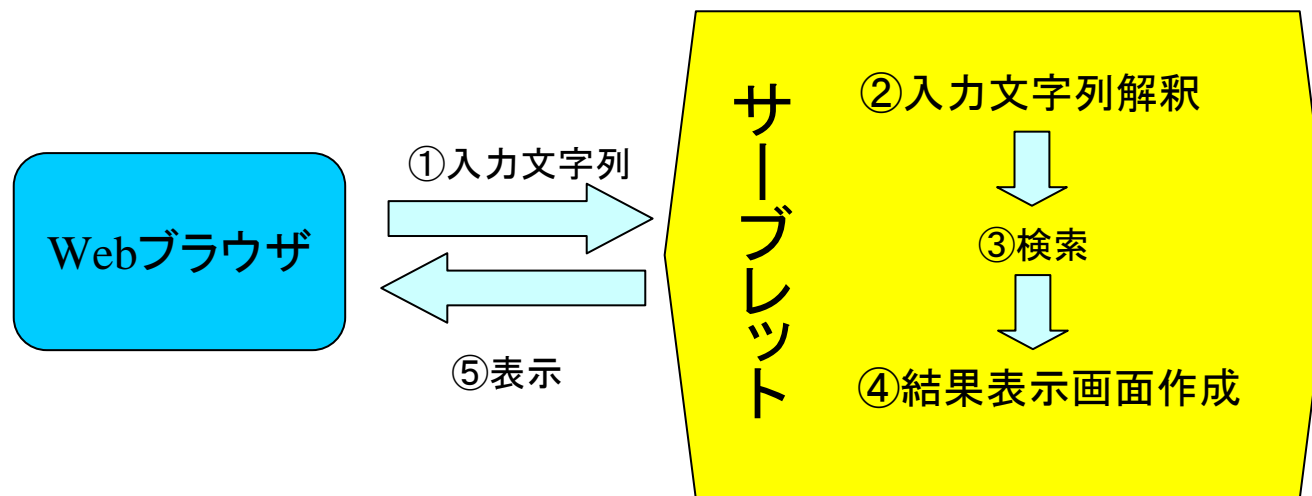


任意の文字列を入力するとGoogleみたいに検索結果が一覧表示される



これがサーブレットでできてます

- Webブラウザから入力した文字列(検索キーワード)を解釈して、検索して、結果を一覧で表示しています
 - あのGoogleだって、基本的にやってることは同じ(はず)



サーブレット (Servlet) とは

- ・ JavaでWebアプリケーションを構築するための仕組み
 - ・ 余談ですが、WebアプリケーションはJavaでなくても作れます
- ・ J2EE / JavaEEの一機能として仕様が決まっていて、仕様に従ってコードを書けば動的なWebサイトを立ち上げることができます

むずかしい話はとりあえずおいておきましょう

まずはWebブラウザに表示

- Webブラウザは、HTMLで記述された内容を画面に表示する



```
<html>
<head>
  <title>検索</title>
  <meta http-equiv="Content-Type"
        content="text/html; charset=UTF-8">
</head>
<body>
  <h2>Googleみたいな検索</h2>
  <form method="GET" action="search">
    <input type="text" name="query">
    <input type="submit" value="検索" >
  </form>
</body>
</html>
```

サーブレットを見てみましょう

```
public class SearchServlet extends HttpServlet {  
  
    protected void doGet(HttpServletRequest request,  
                          HttpServletResponse response)  
        throws ServletException, IOException {  
        :  
    }  
  
    protected void doPost(HttpServletRequest request,  
                          HttpServletResponse response)  
        throws ServletException, IOException {  
        :  
    }  
}
```

サーブレットのクラス

```
import javax.servlet.http.*;
```

```
public class SearchServlet  
    extends HttpServlet {
```

javax.servlet.http.HttpServlet
のサブクラスにします
クラス名は何でも構いません

javax.servlet.http.HttpServlet

- ・ JavaEEで規定された抽象(abstract)クラス
- ・ Webアプリケーションとしてブラウザからの要求に対して行う処理をHttpServletを継承したクラスに書きます
- ・ 処理に応じて、HttpServletのメソッドをオーバーライドします

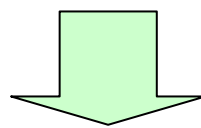
サーブレットで特に重要なメソッド

- ・ doGet

HTMLのform action="GET"に対してsubmitボタンが押された場合に呼び出されるメソッド

- ・ doPost

HTMLのform action="POST"に対してsubmitボタンが押された場合に呼び出されるメソッド



それぞれのactionに対する処理をメソッドの中に記述する

doGetメソッド

```
protected void doGet (HttpServletRequest req,  
                      HttpServletResponse resp)  
    throws ServletException,  
           IOException
```

パラメータ :

req - Webブラウザからの要求

resp - Webブラウザへの応答

doPostメソッド

- ・ doGetとパラメータ、戻り値、例外が一緒

GETとPOSTで処理の内容が違ふことはほとんどないので、doGetとdoPostのいずれからも別のメソッドを単に呼び出すだけにすることが多い

```
protected void processRequest(...) {  
    // ここに実際の処理を書く  
}  
protected void doGet(...) {  
    processRequest(req, resp);  
}  
protected void doPost(...) {  
    processRequest(req, resp);  
}
```

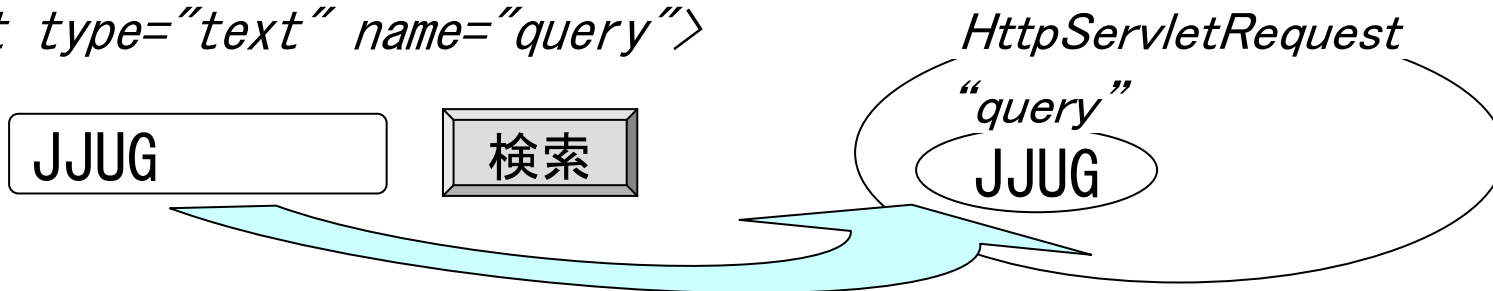
Webブラウザからの要求の内容

- ・ Webブラウザで入力した文字列がdoGet (あるいはdoPost) メソッドの一つ目のパラメータHttpServletRequestオブジェクトにセットされます

より正確には、HTMLのsubmitボタンを押されたformに含まれるinput要素の内容(テキストボックスに入力した文字列やチェックしたラジオボタンの名称など)がセットされます

HttpServletRequestのパラメータ

`<input type="text" name="query">`



```
protected void doGet(HttpServletRequest req, ...  
    String query = req.getParameter("query");  
    => "JJUG"
```

サーブレットの主になる処理

- ・ HttpServletRequestの内容に応じて、サーブレット毎に必要な処理を行います

今回の例だとGoogleみたいな検索処理

```
String result[] = search(query);
```

- サーブレットとは別のクラスに処理を委譲したほうがよい
 - ビジネスロジックがサーブレット (HttpServlet) に依存するのは汎用性が無くなるため (他で使えない)
 - 一般的なWebアプリケーションフレームワークの役割

表示したい内容をHTMLにします

- 処理が終わったら、結果をWebブラウザで表示できるように加工します
 - HTML文書を生成します
 - 最初に「Webブラウザは、HTMLで記述された内容を画面に表示する」とあるとおり
- doGet(あるいはdoPost)メソッドの二つ目のパラメータHttpServletResponseオブジェクトからgetWriterにてWriterオブジェクトを取り出し、HTML文書を書き込みます

HTMLを生成するのは面倒だけど

```
// まず結果がHTMLであることを明確にして  
resp.setContentType("text/html;charset=UTF-8");
```

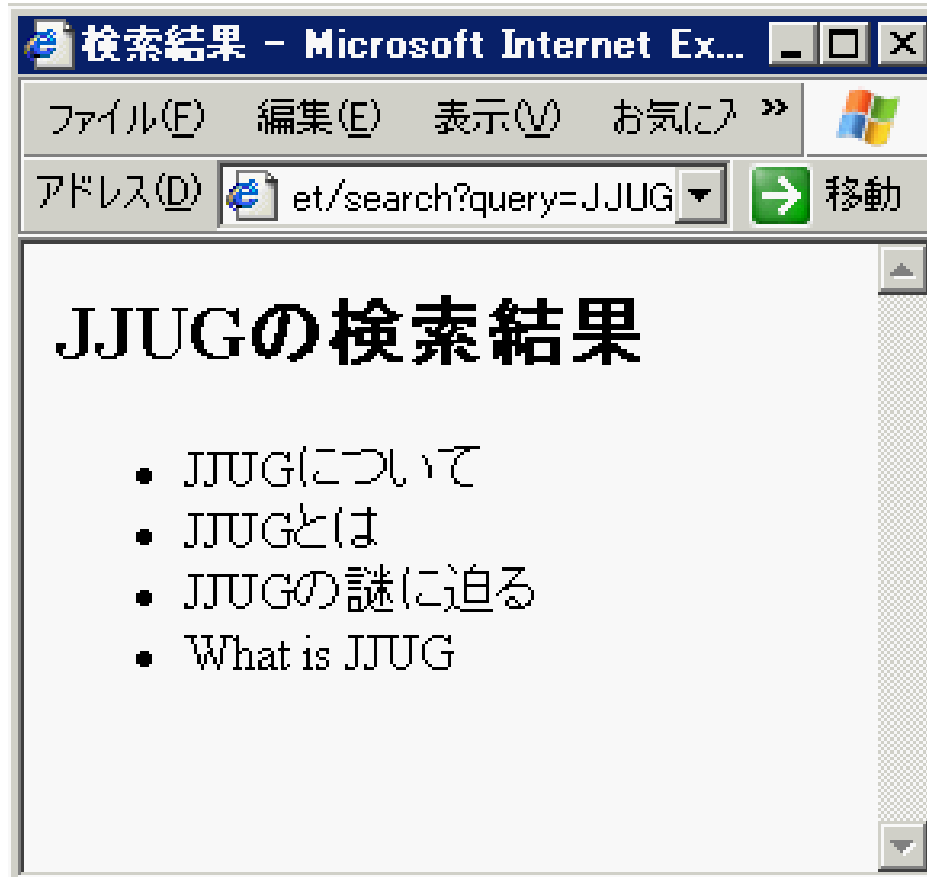
```
// Writerを取り出して  
PrintWriter out = resp.getWriter();
```

```
// ひたすらHTMLを書き込んでいく...
```

```
out.println("<html>");  
out.println("<head>");  
    :  
    :  
out.println("</body>");  
out.println("</html>");  
out.close();
```

Javaソースコード中にHTMLを書くわずらわしさを解消するJSPという技術がありますが、ここでは紹介しません

こんな感じで表示



```
<html>
<head>
<title>検索結果</title>
</head>
<body>
<h2>JJUGの検索結果</h2>
<ul>
<li>JJUGについて
<li>JJUGとは
<li>JJUGの謎に迫る
<li>What is JJUG
</ul>
</body>
</html>
```

ひとつおりの動作を見ました

- ・ Webブラウザでsubmitボタンを押すとサーブレットのdoGetかdoPostが呼ばれる
- ・ formに入力したデータがHttpServletRequestにセットされている
- ・ サーブレットでHttpServletRequestの内容に基づき処理をして、結果をHTMLにしてHttpServletResponseにセットする
- ・ Webブラウザに表示される

サーブレットが複数ある場合は？

- ・ formひとつでサーブレットがひとつだったけど、ページが複数あったり、formが複数あったらどうする？
 - web.xmlにformのaction(パス)毎に呼び出すサーブレットを指定します

web.xml

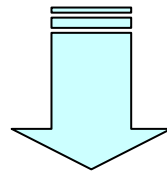
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...省略...>
  <servlet>
    <servlet-name>SearchServlet</servlet-name>
    <servlet-class>jjug. SearchServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>SearchServlet</servlet-name>
    <url-pattern>/search</url-pattern>
  </servlet-mapping>
</web-app>
```

サーブレット
のクラス名

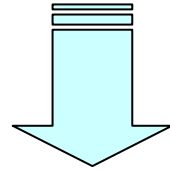
HTMLのform
actionに設
定したパス

Webブラウザから入力すると

Webブラウザから入力すると、web.xmlの
servlet-mappingに従って、サーブレットを
呼び出す...



web.xmlを参照して、サーブレットを呼び出
しているのは誰？



Webブラウザ？

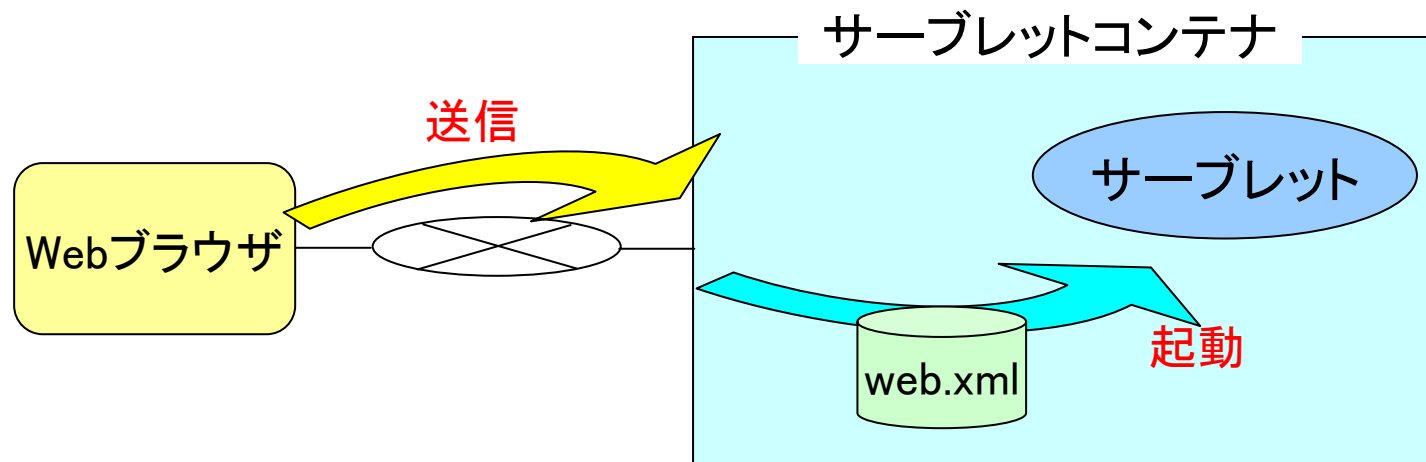
ではありません

そもそもサーブレットを動かしてるのは

- ・ サーブレットコンテナ (Webコンテナ)
 - サーブレットを実行するためのミドルウェア
 - J2EE/JavaEEのサーブレットの仕様に基づき、サーブレットアプリケーションを実行します
 - 代表的なものはApache Tomcat
 - サーブレットに限らず、J2EE/JavaEEの全ての仕様を満たしたGlassFish、BEA WebLogic、IBM WebSphereなどのアプリケーションサーバでもサーブレットは実行できます

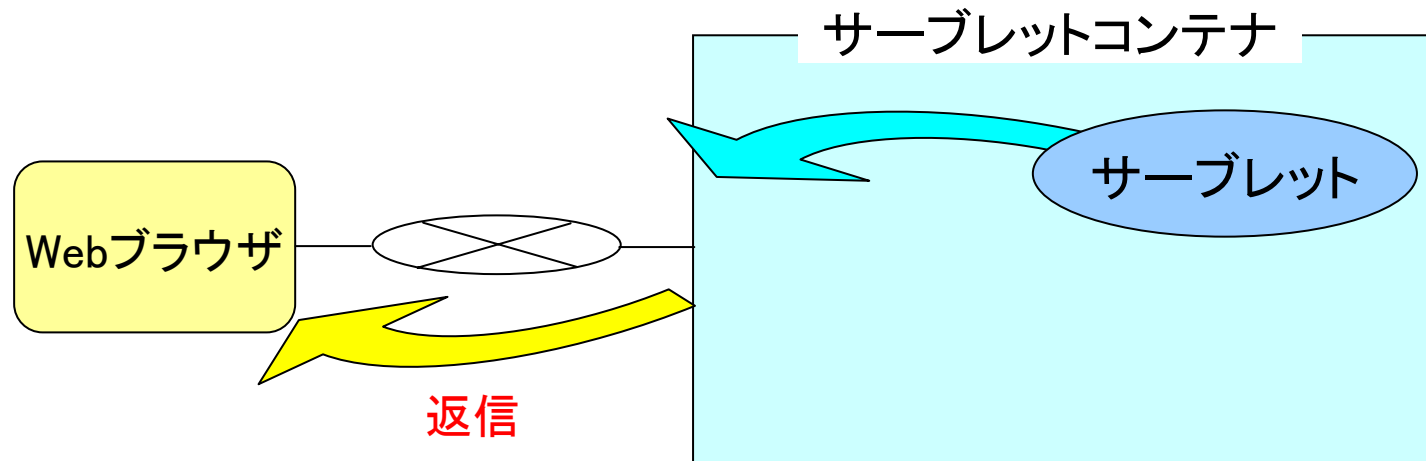
Webブラウザから入力すると

- Webブラウザから入力すると、ネットワークを介してサーブレットコンテナに送信され、内容をHttpServletRequestにセットしてサーブレットを起動する

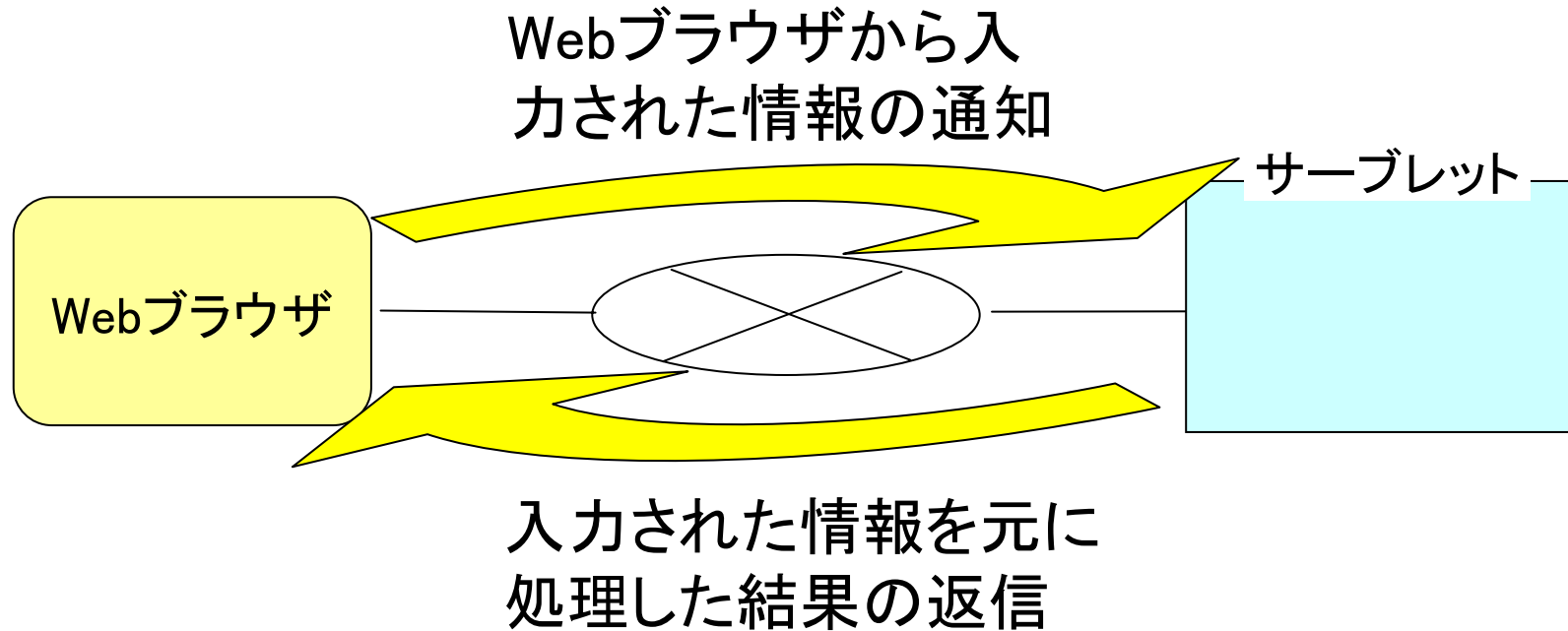


サーブレットが結果をセットすると

- サーブレットがHttpServletResponseにセットした内容をサーブレットコンテナがWebブラウザに返す



Webブラウザとサーバレットの通信



ネットワーク上で情報のやり取り(通信)が行われている

Webブラウザからサーバレットへ

- こんな情報が通知されているはず
 - HTMLのformに書かれたaction (パス)
 - formのinputにWebブラウザで入力された文字列など

こんなのが送られています

```
GET /jjug_servlet/search?query=JJUG HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Macintosh; U; ...
Accept: text/xml, application/xml, ...
Accept-Language: ja, en-us;q=0.7, en;q=0.3
Accept-Encoding: gzip, deflate
Accept-Charset: Shift_JIS, utf-8;q=0.7, *;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://localhost:8080/jjug_servlet/...
```



ちなみにWiresharkと
いうソフトを使いました

これがHTTPです

- なんかごちゃごちゃしてますが、ちゃんと決められた規則に従って送られています
- この規則のことを“**HTTP**”といいます
 - Hypertext Transfer Protocol
 - HTMLやXMLなどのハイパーテキストをWebブラウザとWebサーバで送受信するための**プロトコル**
 - 実際にはハイパーテキスト以外のコンテンツ、たとえば画像なども送受信できます
- HTTPは単純な送信（リクエスト）と返信（レスポンス）の組み合わせ

プロトコルとは

- ネットワーク上で情報をやり取りするための決まりごとを“**プロトコル**”といいます
- コンピュータ・ソフトウェアは融通が利かないので、厳格に決められたとおりに情報をやり取りしなければなりません

一行目に注目すると

GET /jjug_servlet/search?query=JJUG

- GET: GETメソッド
- /jjug_servlet/search: actionで指定したパス
- query=JJUG: inputタグのnameと入力した文字列

HTTP (リクエスト) の構成

一行目	リクエストライン	GET /jjug_servlet/search?query
二行目 ↳ 改行まで	リクエストヘッダ	Host: localhost:8080 User-Agent: Mozilla/5.0 (Macin Accept: text/xml,application/x Accept-Language: ja,en-us;q=0. :
	空行	
改行から ↳ 末尾まで	メッセージボディ	POSTメソッドの場合、ここ にformの内容が並ぶ

返信(レスポンス)の内容

HTTP/1. x 200 OK

X-Powered-By: Servlet/2.5

Server: Sun Java System Application Server 9.1_01

Content-Type: text/html; charset=UTF-8

Content-Length: 204

Date: Sun, 13 Apr 2008 16:17:10 GMT

<html>

<head>

<title>検索結果</title>

</head>

<body>

<h2>JJUGの検索結果</h2>

:



HTTP (レスポンス) の構成

一行目	ステータスライン	HTTP/1. x 200 OK
二行目 ↵ 改行まで	レスポンスヘッダ	X-Powered-By: Servlet/2.5 Server: Sun Java System Applic Content-Type: text/html;charse Content-Length: 204 :
	空行	
改行から ↵ 末尾まで	メッセージボディ	<html> <head> <title>検索結果</title> </head> <body> :

ステータスライン

HTTP/1. x 200 OK

- 200: ステータスコード(成功)
処理結果をコードで示します

主なステータスコード

200	OK. 成功
403	Forbidden. 権限がないなど、アクセスが禁止されている
404	Not Found. リソース(ファイル)が見つからない
500	Internal Server Error. サーバー内部にエラーが発生した
503	Service Unavailable. サービスが一時的に過負荷やメンテナンスで使用不可能である

レスポンスヘッダ

- ・ Content-Type: text/html;charset=UTF-8
 - メッセージボディ(コンテンツ)の種類を示します
 - サーブレットで設定した内容

```
resp.setContentType("text/html;charset=UTF-8");
```

- ・ Content-Length: 204
 - メッセージボディ(コンテンツ)の長さを示します

レスポンスメッセージボディ

- ・ Webブラウザに表示するHTML
 - サーブレットでWriterに書き込んだ内容

```
PrintWriter out = resp.getWriter();  
out.println("<html>");  
out.println("<head>");  
:  
out.println("</html>");  
out.close();
```

- ・ HTMLでなくてもよい

ちなみにWebブラウザじゃなくても

- ・ HTTPは単純な文字列で構成されたプロトコルなので、Webブラウザじゃなくても簡単に扱えます
- ・ 例えば、UNIXやWindowsのtelnetコマンドでサーバーコンテナに接続して、HTTPのリクエストを入力すれば結果が返ってきます

telnetで接続してみた

```
$ telnet 127.0.0.1 8080
```

← 入力

```
Trying 127.0.0.1...
```

```
Connected to 127.0.0.1.
```

```
Escape character is '^]'.
```

```
GET /jjug_servlet/search?query=JJUG HTTP/1.0
```

← 入力

```
HTTP/1.1 200 OK
```

```
X-Powered-By: Servlet/2.5
```

```
Server: Sun Java System Application Server 9.1_01
```

```
Content-Type: text/html;charset=UTF-8
```

```
Content-Length: 204
```

```
Date: Mon, 14 Apr 2008 05:08:41 GMT
```

```
Connection: close
```

```
<html>
```

```
:  
:  
:
```

```
</html>
```

Webブラウザじゃなくてもちゃんと結果が返ってきます

サーブレットコンテナも作ってみよう

- サーブレットコンテナはこんなことをやるはず
 - ① Webブラウザがネットワークを介してリクエストを送信してくるのを待ち受ける
 - ② HTTPリクエストの内容を解釈し、web.xmlに従い、呼び出すサーブレットを決定する
 - ③ パラメータをHttpServletRequestにセットする
 - ④ サーブレットを呼び出す
 - ⑤ HttpServletResponseにセットされた結果をWebブラウザに返す

そんなに難しくありませんね

- ホントはサーブレットの仕様はもっともっといっぱいあって、複雑です
- でも、基本的には全てこの動作の延長にあります

最後に

今日の話は明日からすぐ役に立つものではありませんが、何か問題が起きた時に解決のヒントになったり、ミドルウェアやフレームワークそのものに興味を持っていただくきっかけになればうれしいです

注)この解説の中には概要を理解していただくことを優先し、一部正確ではない記述もあります。ご了承ください。

makino@atware.co.jp