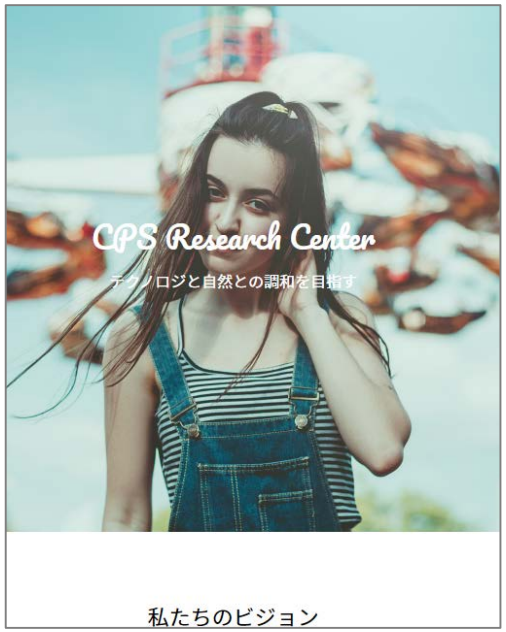
# ITスペシャリスト養成コース

第２回

Webアプリケーション開発入門

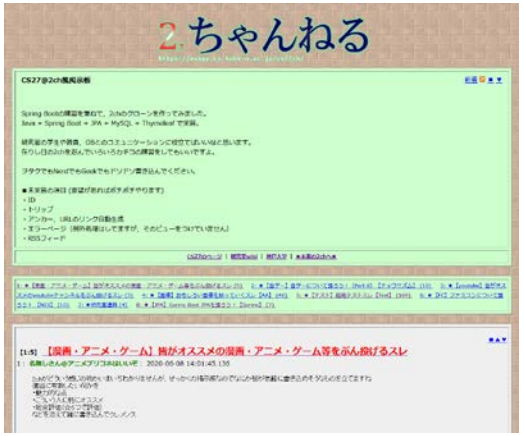
# 前回身に付けた知識

- クラウドのこころ
- ネットワークの基礎





- AWSハンズオン
  - AWS EC2を作成
  - 実行環境構築
    - httpd, java, tomcat, mysql
  - Apache httpdでWebサイト配備
  - Apache tomcatでWebアプリ配備



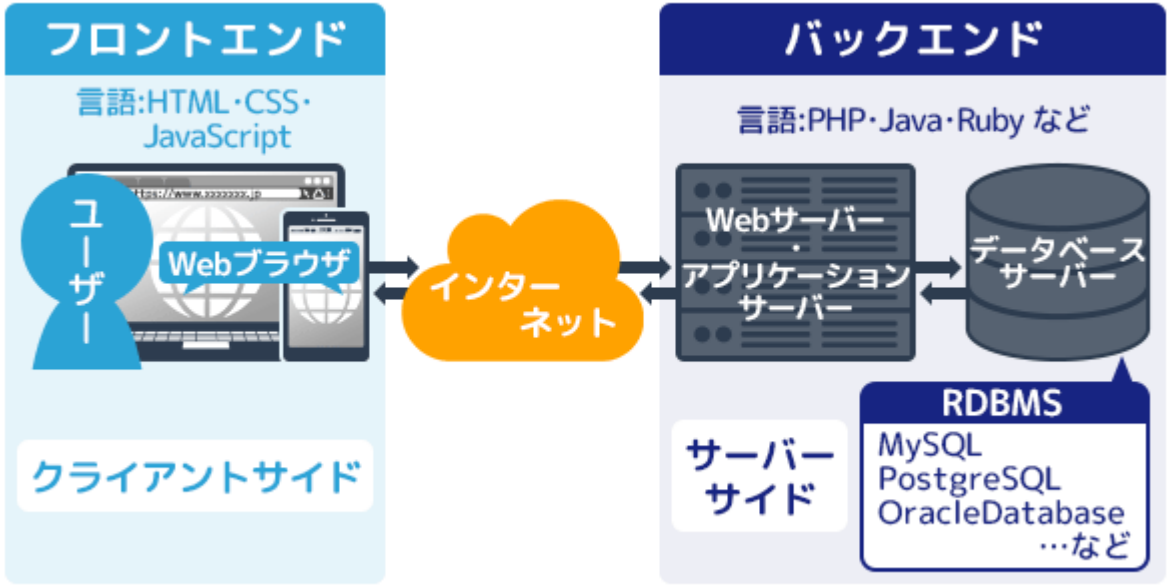
# 今日の目的

- Webアプリケーションのアーキテクチャを理解
  - Webアプリケーションとは
  - HTTPメソッド

- Spring Bootを利用し簡単なWebアプリを作成

# Webアプリケーション

- Webサーバ上で動作するアプリケーション
- Webブラウザを用いて利用
- Web経由でクライアントがサーバにアクセスする



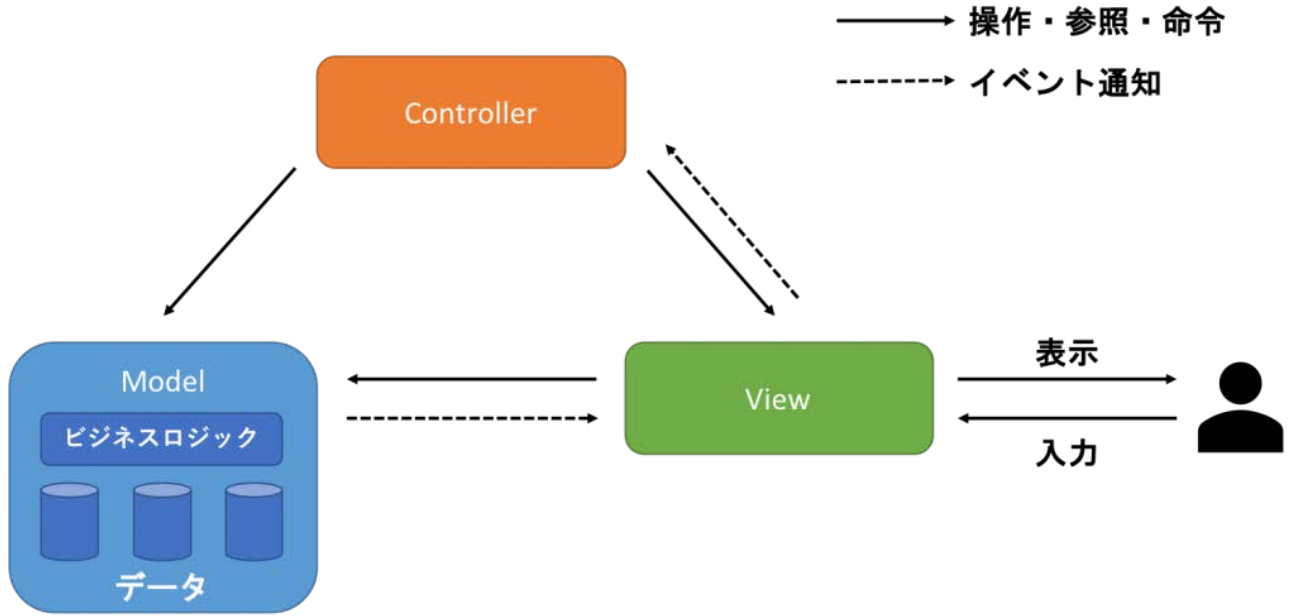

一般的なWebアプリケーションの構成

# 似たような言葉

- Webサービス
  - 広義：Web上で提供されるサービス
  - 狭義： HTTPやXMLなどWebコンテンツの送受信に用いられる標準技術を基盤に、ソフトウェア間でデータや機能を連携できるようにしたもの
- WebAPI
  - （狭義の）Webサービスを利用するためのインタフェース．ほぼWebサービスと同義.
- Webシステム
  - Webを利用したもの．ここまでのWebxxx全部.

# MVC（オリジナル）

- Smalltalkで生まれた概念
- アプリケーションを，Model, View, Controllerに分けるソフトウェアのデザインパターン（概念）

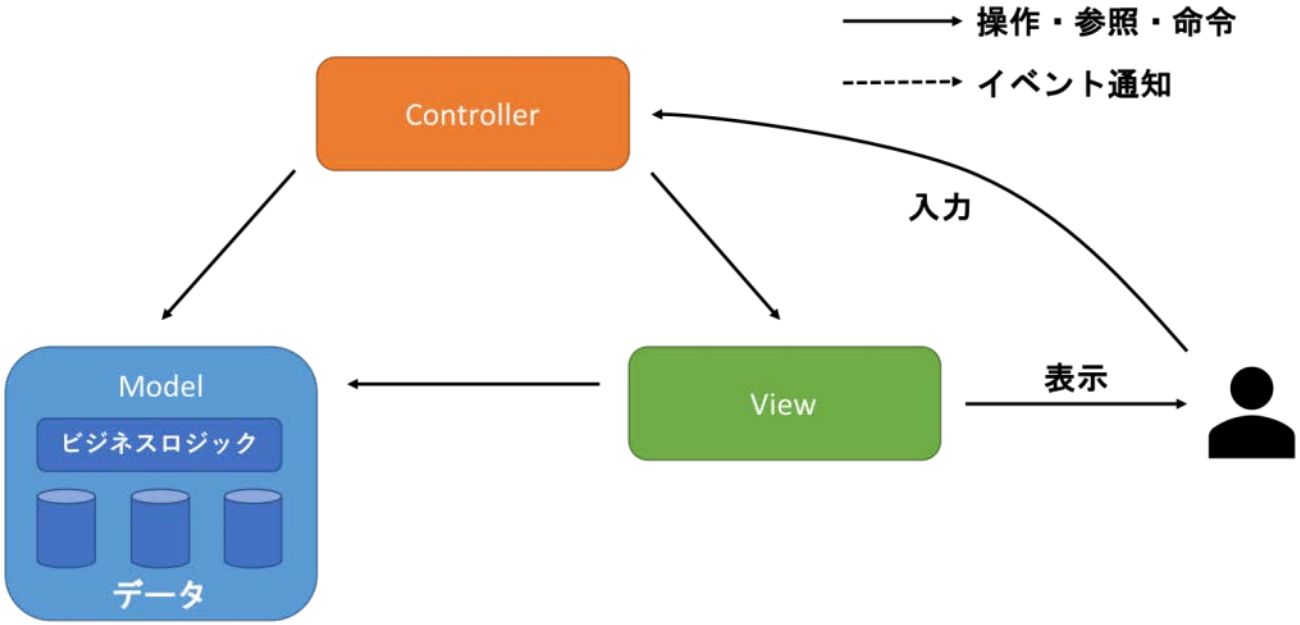



**Model:** データとビジネスロジックを担当．データに変更があったら，ビューに通知
**View:** **Model**からデータを取り出し，整形して出力．ユーザーから入力があったら，コントローラに通知
**Controller:** **View**からの入力を受けて，Modelに入力を伝達．また，ビューに表示命令を出す

https://eejanaica.com/need-attention-mvc-of-web-application/

# MVC （Webアプリ）

- データに変更があったら，Viewに通知する
- ユーザーから入力があったら，コントローラに通知　　→ Webアプリでは難しい



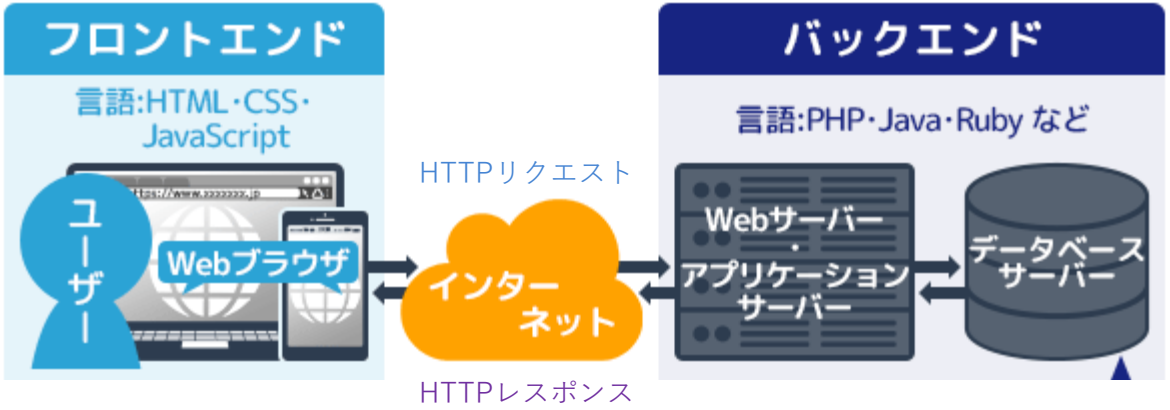

**Model:** データとビジネスロジックを担当.
**View:** **Model**からデータを取り出し，整形して出力.
**Controller:** **ユーザーからの入力を受けて，Modelに入力を伝達．また，Viewに表示命令を出す.**

**ここからのMVCは全てWebアプリの世界の話**

https://eejanaica.com/need-attention-mvc-of-web-application/

# HTTPアクセス

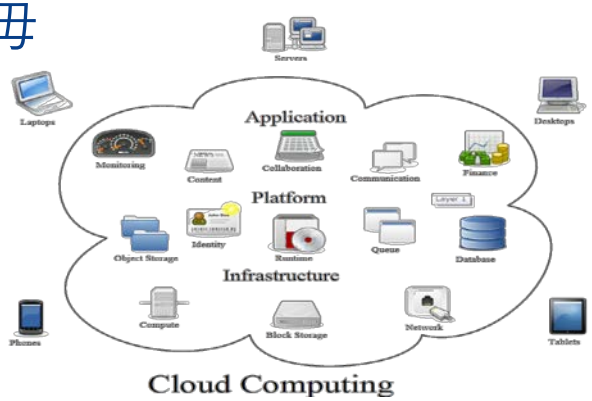
■Webアプリケーションへのアクセスは基本的にブラウザから行う　→　**HTTPによるアクセス**




Webサーバに対し，HTTPリクエストメソッドで機能の要求
リクエストに対し，HTTPレスポンスをサーバは返答

# Webはリソースの海

- Webは様々なリソースが存在する海
- リソース　＝　情報資源
  - 投稿，会員情報，つぶやき，画像，動画，．．
  - c.f. クラウドのこころ




- リソースの識別　＝　URI
  - https://twitter.com/muretti0114/status/1396001518294208513
  - https://qiita.com/masato44gm/items/dffb8281536ad321fb08

- リソースへのアクセス (CRUD) = HTTPメソッド
  - Create, Read,  Update, Delete

# リソース指向アーキテクチャ（ROA）

- それ自体を参照するに値するものをリソースとして定義し，リソースを中心に考えるアーキテクチャ
- ROAに則り実装されたWebサービス：RESTfull API
- ROAの概念
  - 名前：リソースは少なくとも1つのURIを持っていなければならない
  - 表現：Webサーバはリソースを特定の言語(英語，日本語など)，フォーマット(XML，JSONなど)で送信しなければならない
  - リソース間のリンク：リソースには別のリソースへのリンクを含めることができる．リンクを含めることで別のリソースに接続することができる．

詳しくは　https://qiita.com/NagaokaKenichi/items/0f3a55e422d5cc9f1b9c

# HTTPリクエストメソッド

■Webのリソースに対する要求　全８種類

| メソッド | 意味 |
| --- | --- |
| GET | リソースの取得 |
| POST | 子リソースの作成<br>リソースへのデータ追加<br>その他処理 |
| PUT | リソースの更新<br>リソースの作成 |
| DELETE | リソースの削除 |
| HEAD | リソースのヘッダ (メタデータの取得) |
| OPTIONS | リソースがサポートしているメソッドの取得 |
| TRACE | プロキシ動作の確認 |
| CONNECT | プロキシ動作のトンネル接続への変更 |

# HTTPリクエストメソッド

■Webのリソースに対する要求　全８種類

| メソッド | 意味 |
| --- | --- |
| GET | リソースの取得 |
| POST | 子リソースの作成<br>リソースへのデータ追加<br>その他処理 |
| PUT | リソースの更新<br>リソースの作成 |
| DELETE | リソースの削除 |
| HEAD | リソースのヘッダ（メタデータの取得） |
| OPTIONS | リソースがサポートしているメソッドの取得 |
| TRACE | プロキシ動作の確認 |
| CONNECT | プロキシ動作のトンネル接続への変更 |

リソースに対するCRUDに対応

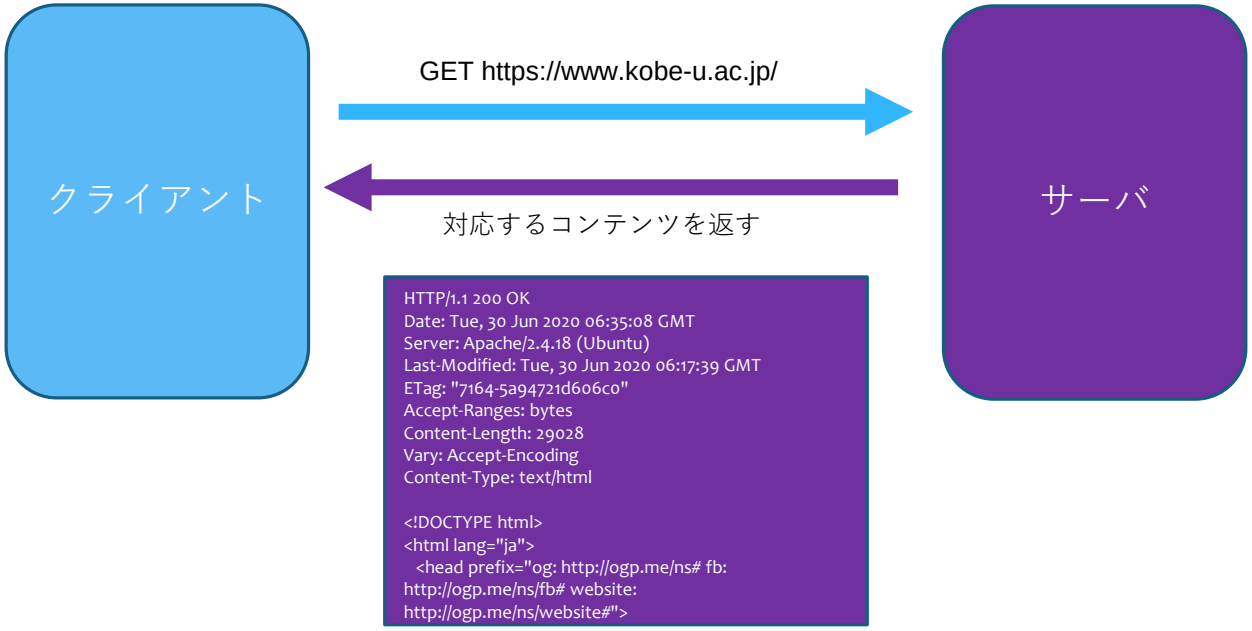
# HTTPメソッド（CRUD）

| CRUD名 | 意味 | メソッド |
| --- | --- | --- |
| Create | 作成 | POST/PUT |
| Read | 読み込み | GET |
| Update | 更新 | PUT |
| Delete | 削除 | DELETE |

Webアプリケーション実装に必要となるデータ操作を実現

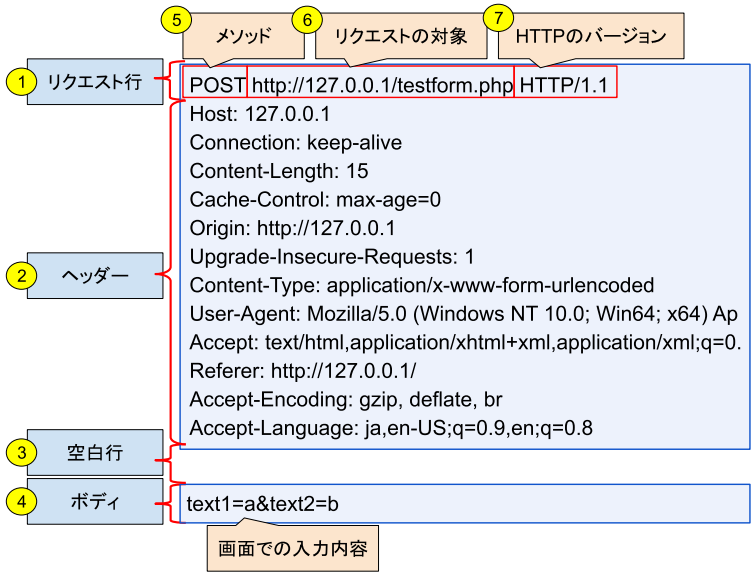
# GET：リソースの取得

- 指定したURIの情報を取得するメソッド
  - ブラウザでURIを入れてアクセスする動作
  - 呼び出しに成功すると，対応するリソースを返却



# POST：リソースの作成，追加

- 指定したURIに情報を送信するメソッド
  - ブラウザのフォームに値を入れて「送信」
  - リソースの新規作成，リソースの追加
  - 業務処理の呼び出しにも用いる
- HTTPリクエストのボディで内容を送る




https://itsakura.com/network-http-get-post

# PUT：リソースの更新，作成

- 指定したURIに情報を置く（上書きする）メソッド
  - HTMLフォームからは実行できない
  - JavaScript等プログラムからの利用を想定

初等的なWebアプリケーションでは利用されない
→ GET/POSTで代用する

スマートフォンアプリやSPA等，クライアント側で画面を生成するモダンなWebアプリで使われる

# DELETE：リソースの削除

- 指定したURIの情報を削除するメソッド
  - HTMLフォームからは実行できない
  - JavaScript等プログラムからの利用を想定

初等的なWebアプリケーションでは利用されない
→ GET/POSTで代用する

スマートフォンアプリやSPA等，クライアント側で画面を生成するモダンなWebアプリで使われる

# RESTful API

- リソースのCRUDを行うWebAPIを，URI+HTTPメソッドで対応させる設計理念
- 例： 管理者・ユーザ管理業務

BaseURI http://kobespiral.com/admin

| | 処理指向のAPI | RESTful API |
|---|---|---|
| **ユーザの作成** | POST /createUser<br>{id=masa-n&name=**中村**} | POST /users<br>{id=masa-n&name=**中村**} |
| **ユーザの取得** | GET /getUser?id=masa-n | GET /users/masa-n |
| **ユーザの更新** | POST /updateUser?id=masa-n<br>{id=masa-n&name=**まさえぬ**} | PUT /users/masa-n<br>{id=masa-n&name=**まさえぬ**} |
| **ユーザの削除** | GET /deleteUser?id=masa-n | DELETE /users/masa-n |

# HTTPレスポンス

- HTTPリクエストに対するサーバのレスポンス
  - 番号によってそれぞれ意味が異なる
    - 1xx：情報
    - 2xx：成功
    - 3xx：リダイレクション
    - 4xx：クライアントエラー
    - 5xx：サーバエラー
  - 詳しくは

https://developer.mozilla.org/ja/docs/Web/HTTP/Status

# URLパラメータ

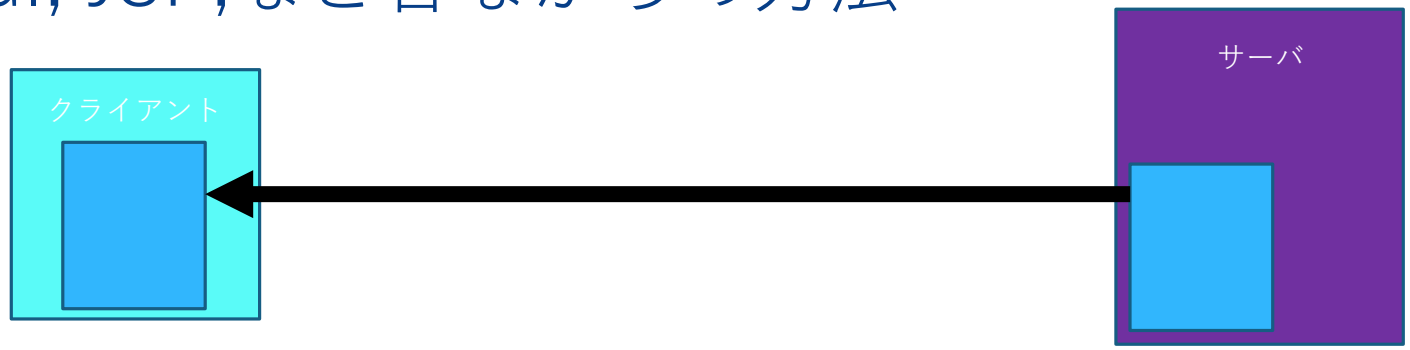
- HTTPメソッドを発行する際のパラメータ
  - サーバに特定の情報を送りたい
    - ID, 名前, 文字列, ・・・

- Request Parameter
  - URLの後ろにクエリとして付け加える
    - server/person?name=hoge&age=20

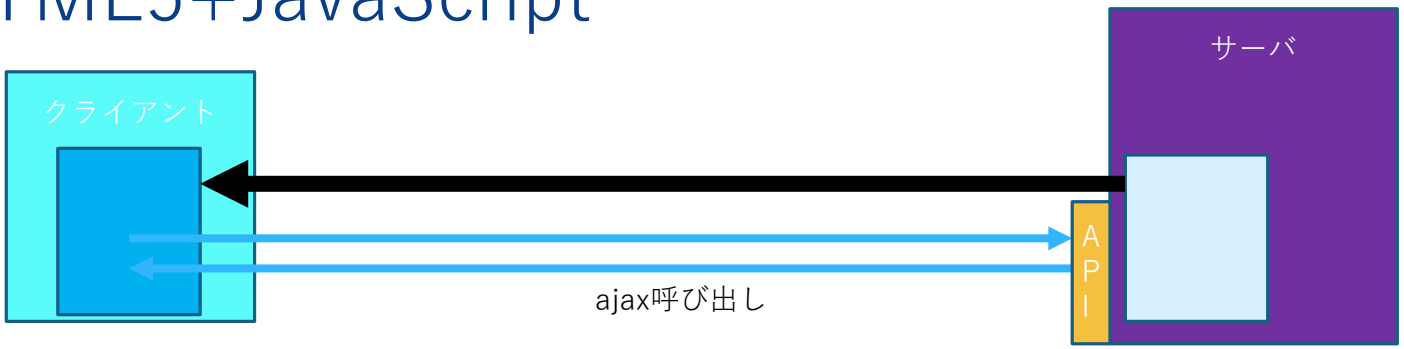
- Path Parameter
  - URLそのものがパラメータになる
    - server/person/hoge/name

# Webアプリ実装のパターン

- サーバ側でViewを生成するパターン
  - CGI, JSP,など昔ながらの方法




- クライアント側でViewを生成するパターン
  - HTML5+JavaScript



# Webアプリケーションの本質

- 目的とするWeb上のリソースに対し，様々なHTTPリクエストを用い操作を行えるインタフェースを提供する
- クライアントからの要求に対し，適切なレスポンスコード，コンテンツを返す

# HTTPリクエストを送ってみる

- Windows Power Shellを起動

- 基本操作：curl [option] [URL]
  - -i レスポンスヘッダを出力
  - -v リクエスト・レスポンス　ヘッダを出力
  - -X POSTリクエスト
    - curl -X POST [url] -d "name=hoge"

# POSTMAN

- HTTPリクエストを送るツール
  - https://www.postman.com/downloads/

# はじめてのSpring Boot

# Webアプリ開発

- 業務（Enterprise）アプリでどのような言語が使われているか


Now We're Talking

Over the last nine months, Java and JavaScript[2] have remained the two dominant languages in the enterprise developer landscape throughout two rounds of research. Industry research firm RedMonk published language rankings in March 2018 that also placed Java and JavaScript at the top tier of development languages–corroborating our findings that Java is alive and well.

In contrast to our findings, however, RedMonk found Python and PHP used more frequently than C# and C++, but only marginally. Indeed, RedMonk's Stephen O'Grady writes that "the numerical ranking is substantially less relevant than the language's tier or grouping."

Overall, IT Decision Makers (ITDMs) report using over 25 total languages–but over half of those languages are used so infrequently as to receive a single digit percentage. In November 2017, ITDMs reported their language preferences, the top six of which are used a significant portion of the time, while many other languages being used for application development are employed at lower rates.

The ordinal ranking and clear separation between Java and JavaScript and all other languages was even more stark in research from March 2018.

| LANGUAGE | NOV 2017 | MAR 2018 | NOV-MAR |
|---|---|---|---|
| Java | 57 | 58 | +1 |
| JavaScript | 54 | 57 | +3 |
| C++ | 45 | 46 | +1 |
| C# | 28 | 26 | -2 |
| Python | 30 | 25 | -5 |
| PHP | 22 | 22 | -- |
| VB.NET | 19 | 17 | -2 |
| C | 19 | 16 | -3 |
| Visual Basic 6 | 18 | 16 | -2 |
| VBA | 16 | 15 | -1 |
| Perl | 16 | 12 | -4 |
| Ruby | 9 | 7 | -2 |
| Swift | 6 | 6 | -- |
| Typescript | 5 | 5 | -- |
| Objective-C | 6 | 5 | -1 |
| Assembly | 6 | 4 | -2 |
| Matlab | 7 | 4 | -3 |
| R | 4 | 3 | -1 |
| Scala | 5 | 3 | -2 |
| Go | 3 | 2 | -1 |
| Groovy | 3 | 2 | -1 |
| Haskell | 2 | 2 | -- |
| CoffeeScript | 2 | 1 | -1 |
| Lua | 3 | 1 | -2 |
| Other | 3 | 3 | -- |

2 JavaScript is needed for many web applications. Question did not differentiate between server-side JavaScript (i.e., Node.js) and browser-based use of the language.



1. Java
2. Java Script
3. C++
4. C#
5. Python

Cloud Foundry Foundation: https://www.cloudfoundry.org/wp-content/uploads/Developer-Language-Report_FINAL.pdf

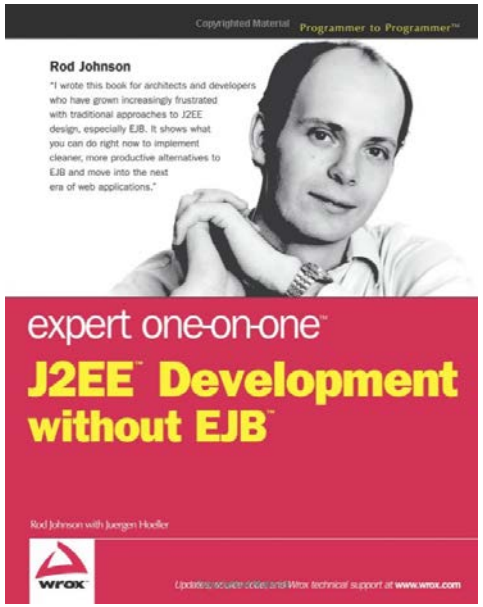
# JavaによるWebアプリ開発

- J2EE: Sun Microsystemsが策定したJavaによる企業システム構築のための仕様
- Java業務アプリケーションのデファクト
- Java1.5以降はJavaEEと改名 → JakartaEE

- 中心機能
  - EJB: ビジネスロジックを実装するオブジェクト
  - Servlet: Webサーバ上で動作するJavaプログラムの仕様を定めた標準の一つ
  - JSP: Webページ内にJavaプログラムを埋め込み、これをサーバ上で実行して結果を反映したページを動的に生成することができる技術

# EJBの仕様が肥大化

■設定方法が複雑，定義が大変，DBパフォーマンス




Rod Johnson
オーストラリアの天才プログラマ

JavaWorld DAY 2005, TOKYO

エンティティBean（EJB：含まれるデータベース・アクセスのカプセル化機能）
なんてないほうがよかった。エンティティBeanのせいで2〜3年が無駄に失われてしまった

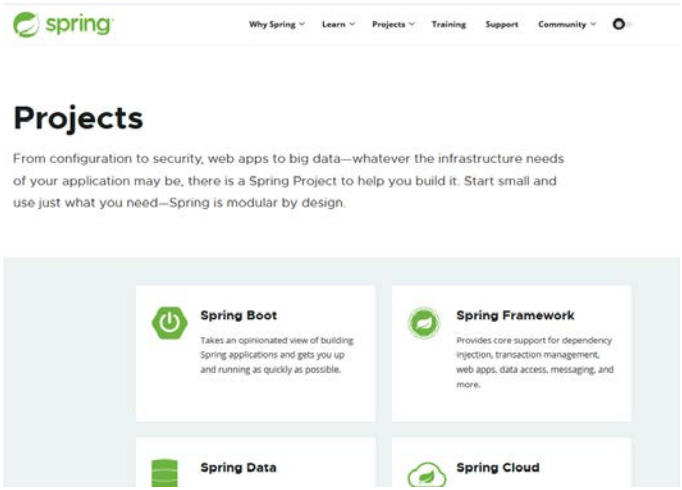
新規のプロジェクトがエンティティBeanを採用したという話は，もはや1件も聞かない

TopLinkはうまく動いていたが，
エンティティBeanは3年もかけてうまく動かないことがわかっただけ

TopLinkはベンダーによる囲い込みという弊害はあるが，
それでも動かないEJBよりはマシ．悪い標準は，標準がないことよりも悪い

# Springプロジェクト

- 2004 Rod Johnsonによりリリース
  - Java向けのOSSフレームワーク
    - 24種類の Spring プロジェクトが存在（2020）
  - Spring FrameworkがCore
    - 当初はDI実現のための小さなフレームワーク




https://spring.io/projects

# Spring Boot

- Springのプロジェクト
  - Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can “just run”
  - 巨大化して分かりにくいSpring Frameworkをベースに手軽にWebアプリを開発できるフレームワーク
    - Spring MVCを用いたWebアプリ作成

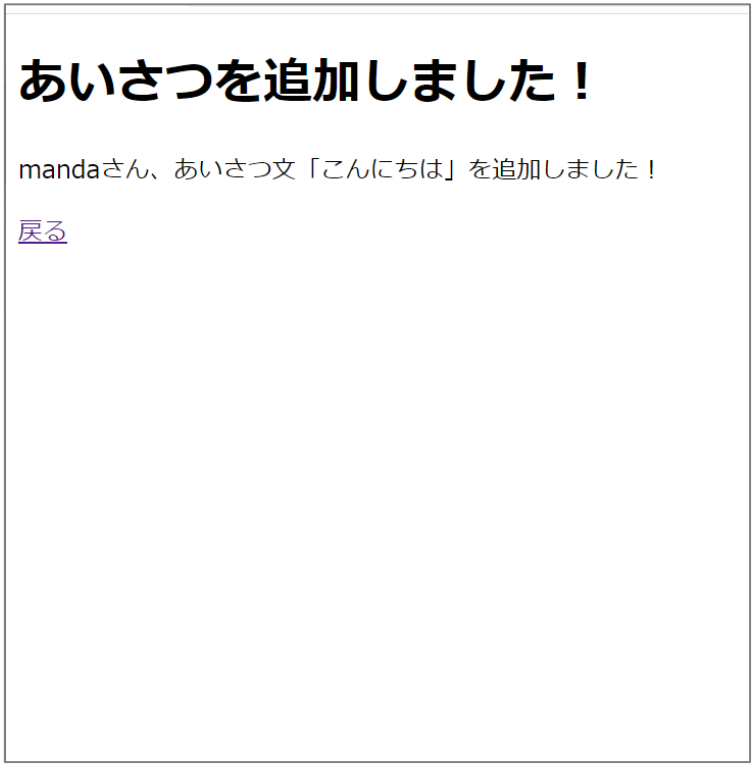
# Hello Spring Boot

# 猫アプリを作ってみよう

- wikiの手順通り猫アプリを作ってテストしてみよう


あいさつ追加 →

← 戻る



# HTMLソースを見てみよう

- 2つの画面それぞれでHTMLのソースを見てみよう
  - 右クリック→ページのソースを表示
- <head>:ヘッダ部
  - 文字セット，タイトル，CSSへのリンク
- <body>: ボディ部
  - h1: 見出し
  - div: ブロック
  - p: 段落
  - img: 画像
  - form: フォーム
  - input: 入力部品
  - a: リンク

```
<!DOCTYPE html>
<html lang="ja">

<head>
  <meta charset="UTF-8" />
  <title>Hello Spring Boot</title>
  <link rel="stylesheet" href="/css/hello.css" />
</head>

<body>
  <h1>Hello Spring Boot!</h1>

  <div class="balloon1">
    <p>おっす．おらTomcat！、こんにちは、</p>
    <p>mandaさん、Spring Bootへようこそにゃ！</p>
  </div>

  <div>
    <img src="/img/cat.jpg" />
  </div>

  <div>
    <form action="/manda/hello/add" method="post">
      <label for="text">あいさつを追加</label>
      <input type="text" name="aisatsu" id="aisatsu" />
      <input type="submit" value="追加" />
    </form>
  </div>
</body>

</html>
```

# 異なるURLでアクセスしてみる

■ページの一部が動的に生成されている

`http://localhost:28280/manda/hello`

```
<!DOCTYPE html>
<html lang="ja">

<head>
  <meta charset="UTF-8" />
  <title>Hello Spring Boot</title>
  <link rel="stylesheet" href="/css/hello.css" />
</head>

<body>
  <h1>Hello Spring Boot!</h1>

  <div class="balloon1">
    <p>おっす、おらTomcat！、こんにちは、</p>
    <p>mandaさん、Spring Bootへようこそにゃ！</p>
  </div>

  <div>
    <img src="/img/cat.jpg" />
  </div>

  <div>
    <form action="/manda/hello/add" method="post">
      <label for="text">あいさつを追加</label>
      <input type="text" name="aisatsu" id="aisatsu" />
      <input type="submit" value="追加" />
    </form>
  </div>
</body>

</html>
```

これまで入力した挨拶によって変わる

`http://localhost:28280/masa-n/hello`

```
<!DOCTYPE html>
<html lang="ja">

<head>
  <meta charset="UTF-8" />
  <title>Hello Spring Boot</title>
  <link rel="stylesheet" href="/css/hello.css" />
</head>

<body>
  <h1>Hello Spring Boot!</h1>

  <div class="balloon1">
    <p>おっす、おらTomcat！、こんにちは、</p>
    <p>masa-nさん、Spring Bootへようこそにゃ！</p>
  </div>

  <div>
    <img src="/img/cat.jpg" />
  </div>

  <div>
    <form action="/masa-n/hello/add" method="post">
      <label for="text">あいさつを追加</label>
      <input type="text" name="aisatsu" id="aisatsu" />
      <input type="submit" value="追加" />
    </form>
  </div>
</body>

</html>
```

# HTMLテンプレート

Thymeleaf

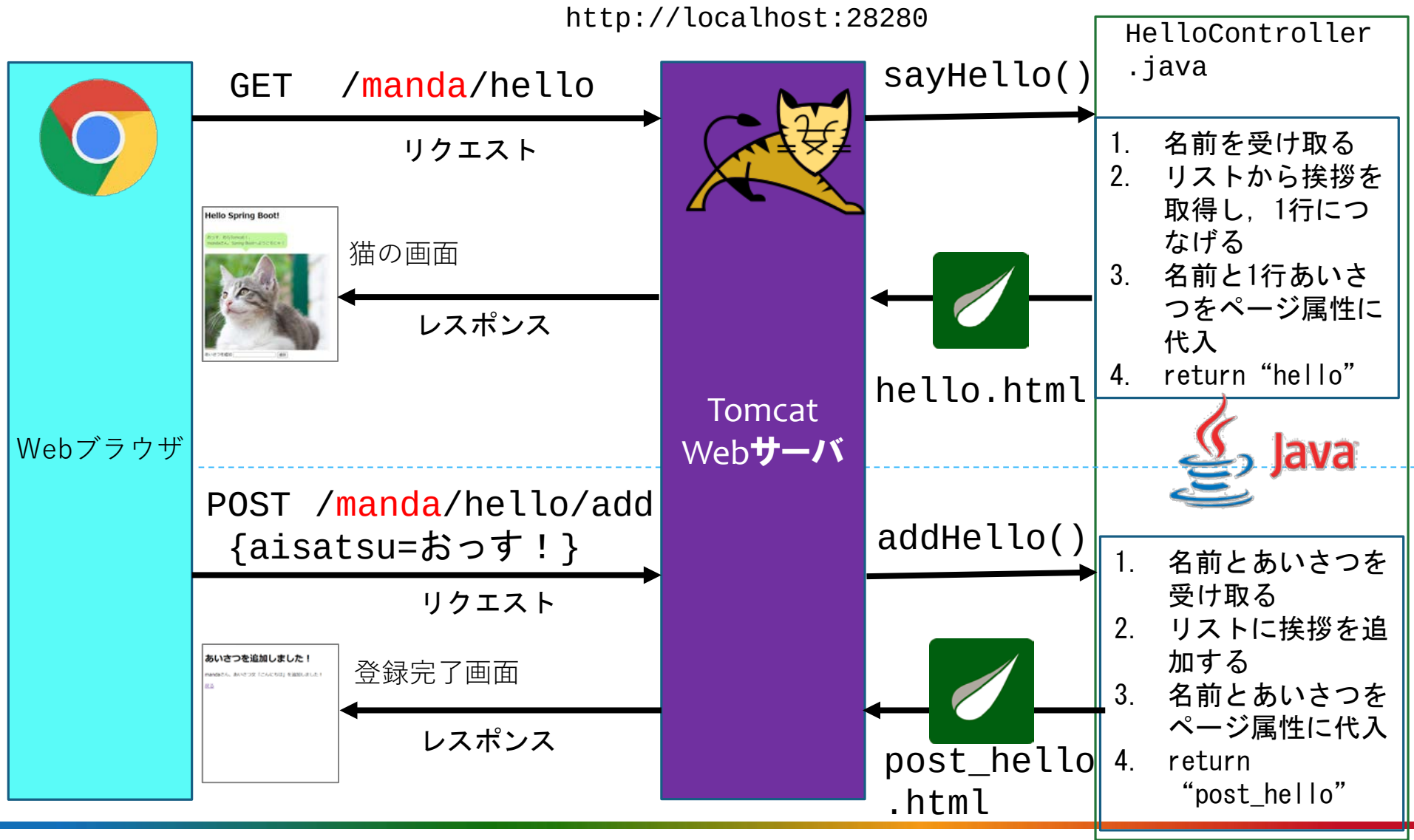
- 可変部が穴抜き（変数）になったHTMLのひな形
  - サーバ側で値をセット
  - テンプレートエンジン(**Thymeleaf**)が完全なHTMLに整形して返す
- 代表的な変数
  - 変数式
    - th:属性=“${変数}”
  - インライン式
    - [[${変数}]]
  - リンク式
    - @{/aa/{x}/bb(x=${変数})}}

hello.html

```
<!DOCTYPE html>
<html lang="ja" xmlns:th="http://www.thymeleaf.org">

<head>
  <meta charset="UTF-8" />
  <title>Hello Spring Boot</title>
  <link rel="stylesheet" th:href="@{/css/hello.css}" />
</head>

<body>
  <h1>Hello Spring Boot!</h1>

  <div class="balloon1">
    <p th:text="${greeting}"></p>
    <p>[[${name}]]さん、Spring Bootへようこそにゃ！</p>
  </div>

  <div>
    <img th:src="@{/img/cat.jpg}" />
  </div>

  <div>
    <form th:action="@{/{n}/hello/add(n=${name})}" method="post">
      <label for="text">あいさつを追加</label>
      <input type="text" name="aisatsu" id="aisatsu" />
      <input type="submit" value="追加" />
    </form>
  </div>
</body>

</html>
```

https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf_ja.html

# 猫アプリの振る舞い

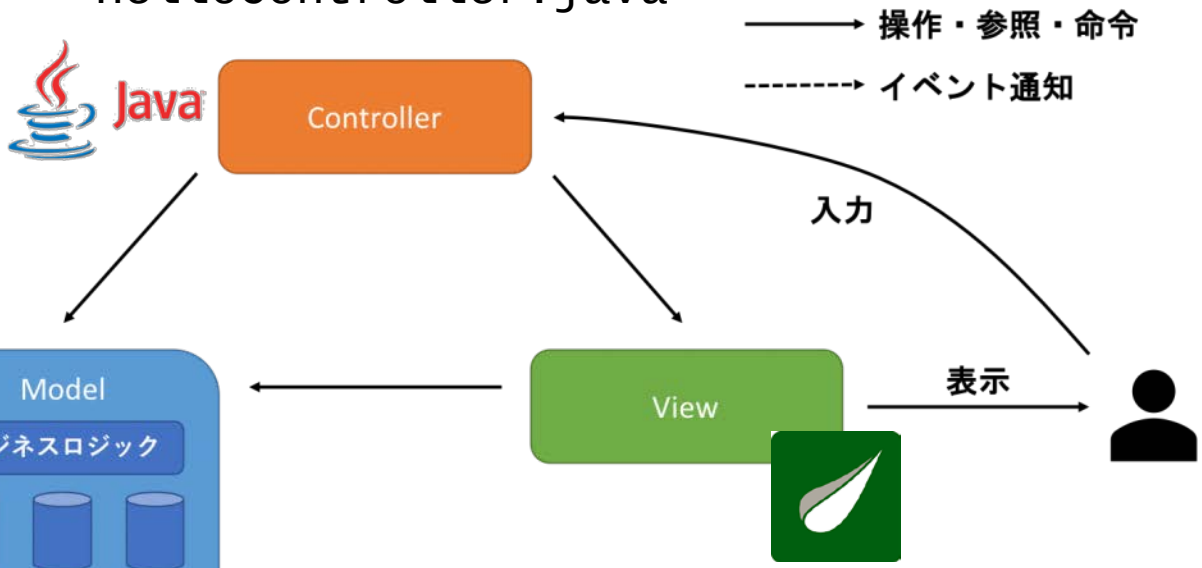

http://localhost:28280
HelloController
.java
Webブラウザ
GET /manda/hello
リクエスト
sayHello()
1. 名前を受け取る
2. リストから挨拶を取得し，1行につなげる
3. 名前と1行あいさつをページ属性に代入
4. return “hello”
Hello Spring Boot!
猫の画面
レスポンス
hello.html
Tomcat
Webサーバ
Java
POST /manda/hello/add
{aisatsu=おっす！}
リクエスト
addHello()
1. 名前とあいさつを受け取る
2. リストに挨拶を追加する
3. 名前とあいさつをページ属性に代入
4. return “post_hello”
あいさつを追加しました！
登録完了画面
レスポンス
post_hello
.html

# HelloController.java 解説

- @Controller: このクラスがWeb-MVCのコントローラ(C)であることを示す
- @GetMapping: GETリクエストをマッピング
- @PostMapping: POSTリクエストをマッピング
- @PathVariable: パス・パラメータの取り出し
- @RequestParam: リクエスト・パラメータの取り出し
- Model: 画面のモデル．HTMLテンプレートに変数を渡すために使う
- return “文字列”: 文字列.htmlをテンプレートにして出力する

# Web-MVCとしての猫アプリ

HelloController.java




ArrayList<List> list

/resources/templates/hello.html
/resources/templates/post_hello.html

# 猫アプリの問題点

- **あらゆる処理をコントローラで行っている**
  - リクエストマッピング，業務処理，画面変数セットなど，様々な処理が混ざっている
  - コントローラが肥大化する

  【対策】Webアプリ処理と業務処理を分ける

- **データが永続化されない**
  - アプリを再起動するとあいさつが全部消える

  【対策】データベースにデータを格納する

# ToDo管理アプリケーション

- Spring Boot で基本的なレイヤード・アーキテクチャのWebアプリに挑戦しよう
  - ビュー
  - コントローラ
  - サービス
  - レポジトリ

続きはWikiで！