

ご注文はSpringですか？

Springで心弾むアプリケーション開発

Spring Framework



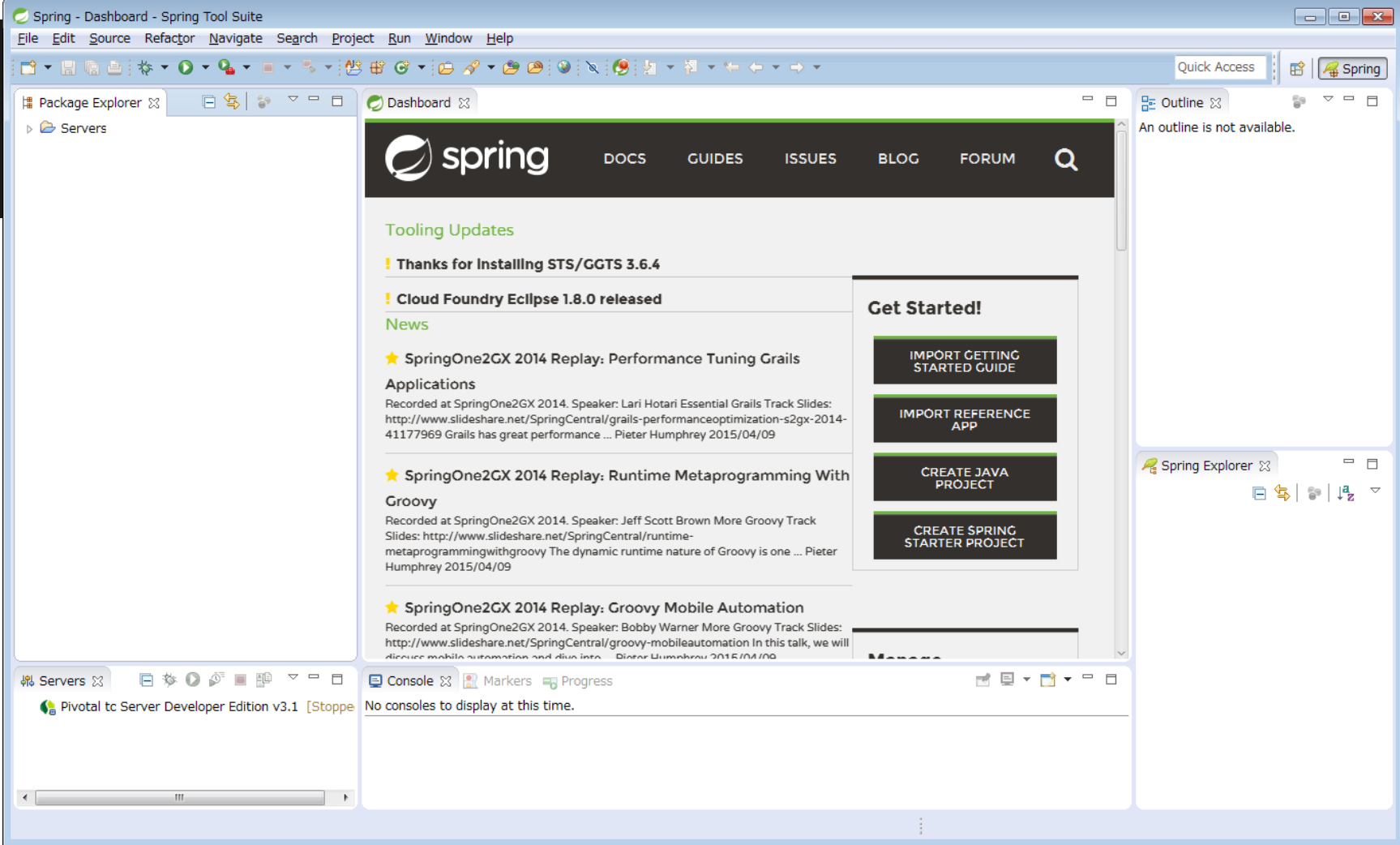
- Java向けのアプリケーションフレームワーク
- Webアプリの使用実績が多いみたい？
- 多機能
 - DI、AOP、Web、ORマッパー、...

<http://spring.io/projects>

なにはともあれ...

始めてみよう

- STS (Spring Tool Suite)
プラグイン全部入りのEclipse
<https://spring.io/tools>
- Maven
<http://maven.apache.org/download.cgi>



最初のプロジェクト

- Mavenで空のプロジェクトを作成

```
$ mvn -B archetype:generate \  
-DgroupId=com.example.joniburn \  
-DartifactId=spring-hello \   ← プロジェクト名  
-DarchetypeArtifactId=maven-archetype-quickstart
```

Spring Boot

- pom.xmlにSpring Bootを追加する
- Spring Bootとは？
 - 簡単にアプリ作成するための仕組み(らしい)
 - pom.xmlやservlet-context.xml等の記述量を減らせる
 - とにかく今私はアプリを作りたいんだ！

pom.xml

...

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.2.3.RELEASE</version>

</parent>

<dependencies>

...

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

...

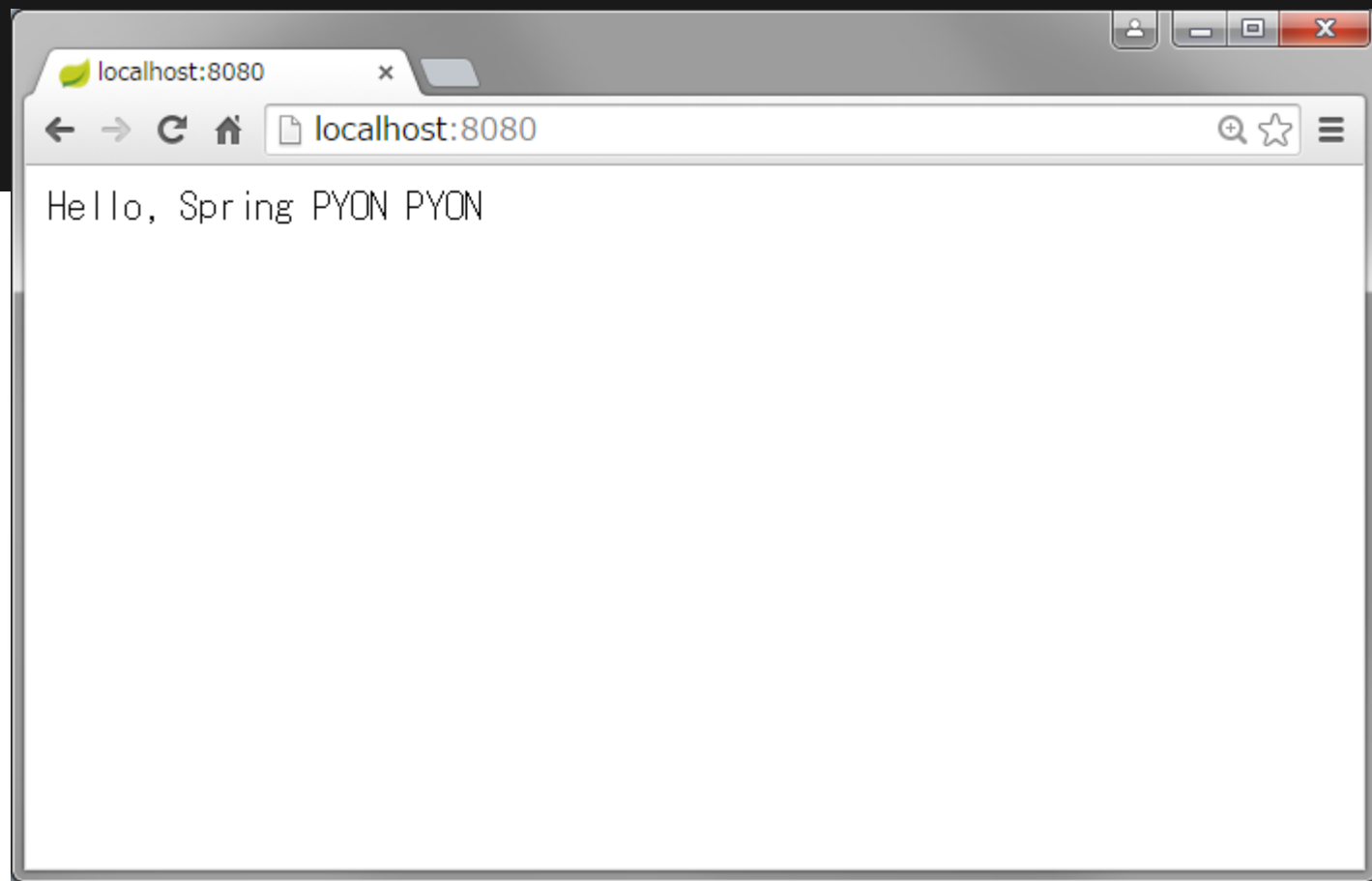
追加



```
@Controller
@EnableAutoConfiguration
public class IndexController {

    @RequestMapping (value="/", produces="text/plain")
    @ResponseBody
    public String index() {
        return "Hello, Spring PYON PYON";
    }

    public static void main(String[] args) {
        SpringApplication.run(IndexController.class, args);
    }
}
```

IndexController.java

```
@Controller
```

```
@EnableAutoConfiguration
```

Spring Bootのおまじない
よく知らない><;

```
public class IndexController {
```

```
    @RequestMapping (value="/", produces="text/plain")
```

```
    @ResponseBody
```

```
    public String index() {
```

```
        return "Hello, Spring PYON PYON";
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(IndexController.class, args);
```

```
    }
```

```
}
```

IndexController.java

@Controller

@EnableAutoConfiguration

public class IndexController {

MVCのCであることを示す
URLに対応したメソッドを書けるようになる

@RequestMapping (value="/", produces="text/plain")

@ResponseBody

public String index() {

URL

Content-Type

return "Hello, Spring PYON PYON";

}

メソッドの戻り値がHTTPボディになる

public static void main(String[] args) {

SpringApplication.run(IndexController.**class**, args);

}

}

Hello Worldはこの辺にして

- チャットアプリを作ろう
- DBなし、データはメモリ上
- ログインなし
- 名前入力あり

サーバ

ChatService

チャット発言

IndexController

クライアント

投稿

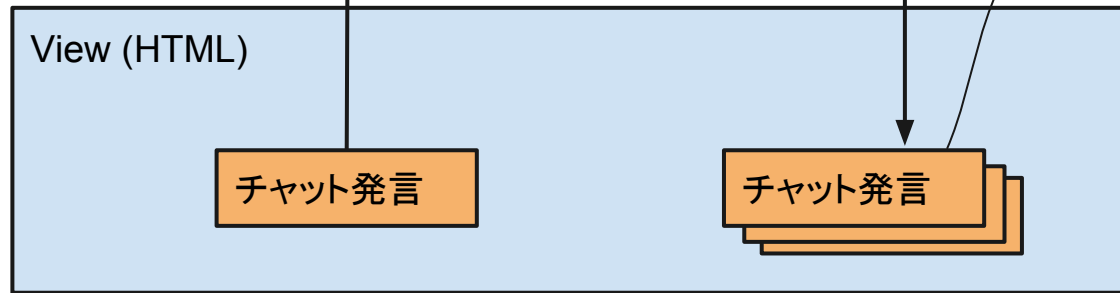
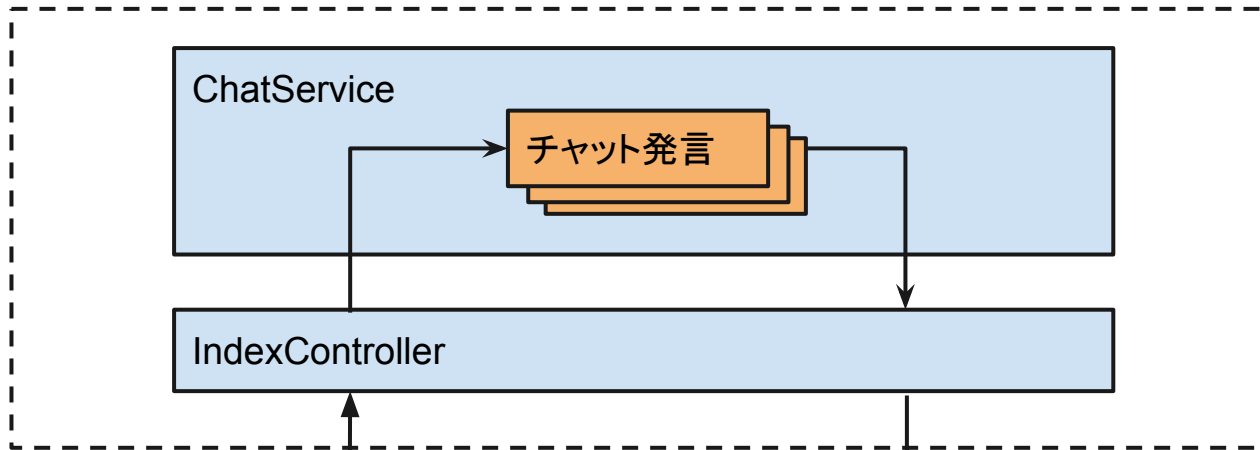
表示

View (HTML)

チャット発言

チャット発言

・名前
・発言テキスト



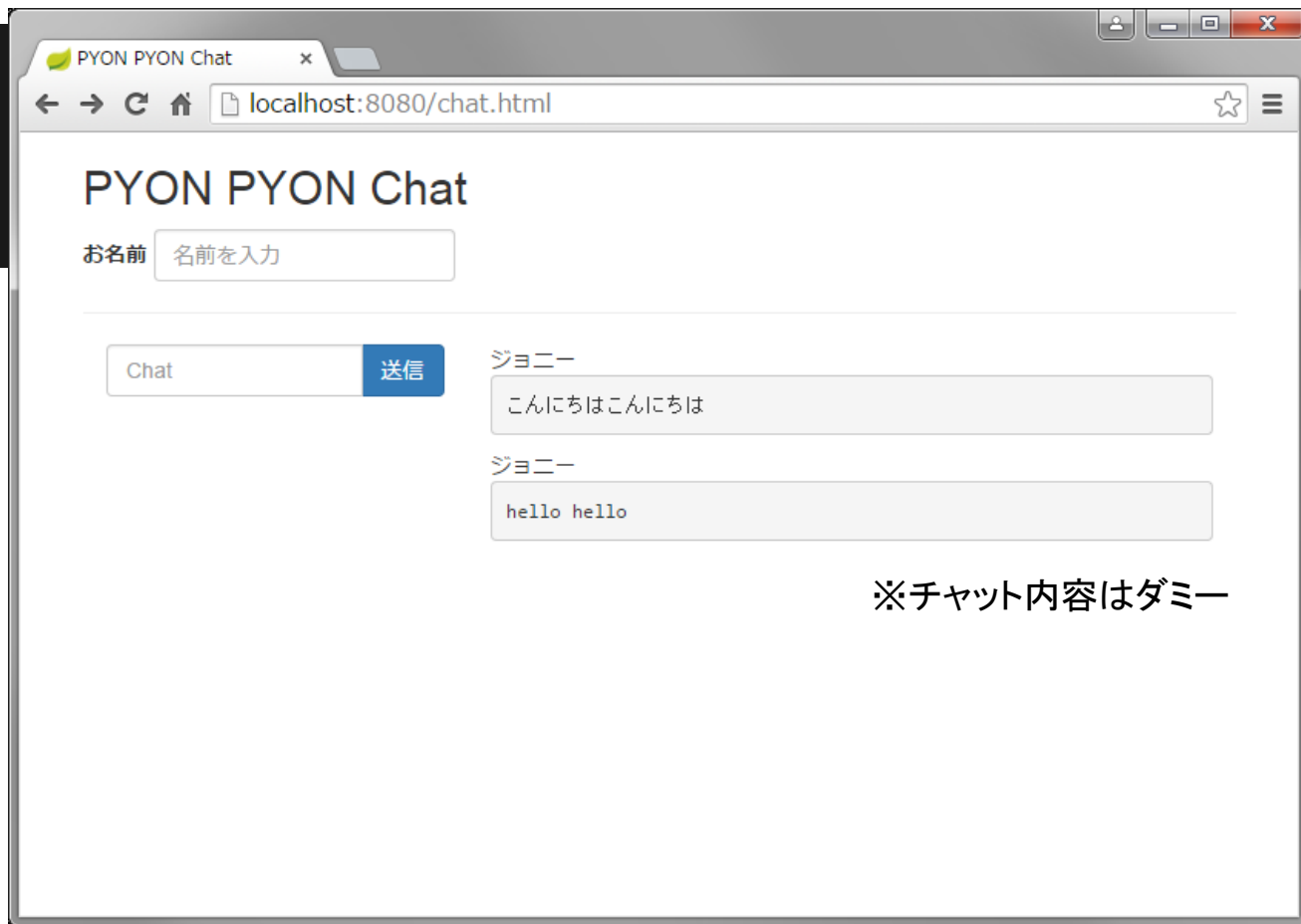
Viewの配置

- まずは、ただの静的なHTMLを置いてみる
 - 後でテンプレートファイルに置き換えます
- 置き場所

src/main/resources/static/chat.html

/css/...

/js/...



送信ボタンを押すと /chat-api にPOSTするようにJavaScriptを作成。(詳細は割愛)

サーバとのAPIを決める

- せっかくなのでajaxでJSONをやりとり

```
{  
  name: "名前",  
  chatText: "発言テキスト"  
}
```


サーバプログラムを作る (1)

- class Chat → ただのデータクラス

```
public class Chat {  
    public String name;  
    public String chatText;  
}
```

サーバプログラムを作る (2)

- class ChatService → Chatを蓄積する

```
@Service
public class ChatService {

    private List<Chat> chatList = new LinkedList<Chat>();

    public void addChat(Chat chat) {
        chatList.add(chat);
    }
}
```

サーバプログラムを作る (3)

- class IndexController
 - ajaxリクエストを受ける

```
@Controller
@EnableAutoConfiguration
```

```
@ComponentScan
```

```
public class IndexController{
```

```
...
```

```
    @Autowired
```

```
    private ChatService chatService;
```

```
...
```

```
    @RequestMapping(value= "/chat-api",
                      method=RequestMethod.POST)
```

```
    @ResponseStatus(HttpStatus.OK)
```


```
    public void chat(@RequestBody Chat chat) {
```

```
        chatService.addChat(chat);
```

```
    }
```

```
}
```

ChatServiceのインスタンスを作成してもらう
(自分でnewする必要なし)



JSONとしてPOSTされたデータが、
Chatクラスに自動変換



ここまでで...

- 画面が表示できる
- 送信ボタンを押すと、サーバ側にチャット内容が蓄積される

→次は、表示する処理

サーバプログラムを作る (4)

- class ChatService

```
@Service
public class ChatService {

    private List<Chat> chatList = new LinkedList<Chat>();
    ...

    public List<Chat> getAllChat() {
        return chatList;
    }
}
```

サーバプログラムを作る (5)

- class IndexController

```
public class IndexController {  
    ...  
  
    @RequestMapping(value="/chat-api",  
                    method=RequestMethod.GET)  
    @ResponseBody  
    public List<Chat> getAllChat() {  
        return chatService.getAllChat();  
    }  
}
```

メソッドの戻り値が自動的に
JSONに変換される

ここまでで...

- 画面が表示できる
- 送信ボタンを押すと、サーバ側にチャット内容が蓄積される
- ブラウザにチャットの一覧を返す

ちなみに、画面は応答のJSONをよしなに表示するようにしてあります(割愛)

→まずは完成(―――)b

不満な点

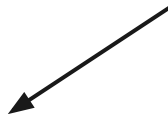
- ブラウザリロードすると名前が消えちゃう
 - セッション情報に保存する
 - サーバ側でHTMLを生成して、
名前欄に入力された状態でブラウザに返す

サーバプログラムを作る (6)

- class IndexController

```
public class IndexController {  
    ...  
    @Autowired  
    private HttpSession session;  
  
    @RequestMapping(value= "/chat-api",  
        method=RequestMethod.POST)  
    @ResponseStatus(HttpStatus.OK)  
    public void chat(Chat chat) {  
        chatService.addChat(chat);  
  
        session.addAttribute("name", chat.name);  
    }  
}
```

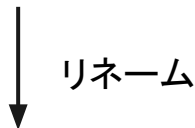
発言者の名前を
セッションに保存



Viewのテンプレート (1)

- 静的なHTMLを、動的なテンプレートに変更

src/main/resources/static/chat.html



src/main/resources/templates/chat.vm

Viewのテンプレート (2)

- class IndexController にメソッドを追加


```
public class IndexController {  
    ...  
    @Autowired  
    private HttpSession session;  
  
    @RequestMapping(value= "/chat-api",  
        method=RequestMethod.GET)  
    public void chatView(ModelMap model) {  
        model.put("name", session.getAttribute("name"));  
        return "chat";  
    }  
}
```

chat.vmのこと

Viewのテンプレート (3)

- chat.vmの修正

```
...  
<input type="text" placeholder="お名前"  
  value="$!{name}">  
...
```



名前欄に初期値が入る

Velocityの有効化

pom.xml

...

<dependencies>

...

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-**velocity**</artifactId>

</dependency>

</dependencies>

...

↑
starter-web を starter-velocity に変えるだけでよい。
あとはSpring Bootが依存ライブラリや ***.xmlのような
設定を全て引き受けてくれる。

ここまでで...

- 画面が表示できる
 - 送信ボタンを押すと、サーバ側にチャット内容が蓄積される
 - ブラウザにチャットの一覧を返す
 - ブラウザリロードしても名前が保持される
- おつかれさまでした！

まとめ

- Spring Frameworkを使うと、面倒なことを書かずにアプリの処理に注力できる
 - JSONパース/生成やURLのハンドリングなど...
- Spring Bootで面倒くさい設定を簡略化できる