


Spring Bootで簡単な検索アプリケーションを開発する

 spring-boot 1.24 157

 Java 8 3669

 (/rubytomato@github)rubytomato@github (/rubytomato@github)が2015/07/26に投稿(2015/08/28に編集)・編集履歴(5) (/rubytomato@github/items/e4fda26faddbcfd84d16/revisions)・問題がある投稿を報告する

 18
ストック

 0
コメント

 ストック

 (/kimukou) いゝ (/Noboruhi) (/shiena) (/n_slender) (/nagi244) (/noriya) (/irisawa@github) (/Reds) (/davaa) (/neogym) ... (/rubytomato@github/items/e4fda26faddbcfd84d16/stockers)

概要

Spring Bootを使用して、簡単な検索ができるWebアプリケーションを開発します。

完成

開発するアプリケーションは「俳優」の情報を扱い、データの一覧表示、登録、削除を行います。

完成図

俳優

一覧 新規

Search for...

Search!

id	名前	身長	血液型	誕生日	出身地	更新日時	
1	丹波哲郎	175	O	1922-07-17	13:東京都	2015-07-25 23:44:15.739952	delete
2	森田健作	175	O	1949-12-16	13:東京都	2015-07-25 23:44:15.739952	delete
3	加藤剛	173	-	1938-02-04	22:静岡県	2015-07-25 23:44:15.739952	delete
4	島田陽子	171	O	1953-05-17	43:熊本県	2015-07-25 23:44:15.739952	delete
5	山口果林	-	-	1947-05-10	13:東京都	2015-07-25 23:44:15.739952	delete
6	佐分利信	-	-	1909-02-12	1:北海道	2015-07-25 23:44:15.739952	delete
7	緒形拳	173	B	1937-07-20	13:東京都	2015-07-25 23:44:15.739952	delete
8	松山政路	167	A	1947-05-21	13:東京都	2015-07-25 23:44:15.739952	delete
9	菅井きん	155	B	1926-02-28	13:東京都	2015-07-25 23:44:15.739952	delete
10	笠智衆	-	-	1904-05-13	43:熊本県	2015-07-25 23:44:15.739952	delete

spring boot sample application

(https://qiita-image-store.s3.amazonaws.com/0/22772/f1395e78-537b-e6c2-3346-0bbcc631aa93.png)

環境

- Windows7 (64bit)
- Java 1.8.0_45
- Spring Boot 1.2.4
 - thymeleaf 2.1.4
 - logback 1.1.3
- MySQL 5.6
- Eclipse 4.4
- Maven 3.3.3

参考

下記のサイトを参考にさせていただきました。

Spring

- Spring Boot (<http://projects.spring.io/spring-boot/>)

- Spring Boot Reference Guide (<http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/>)
- spring-projects/spring-boot (<https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples/spring-boot-sample-data-jpa>)

Thymeleaf

- Thymeleaf - Tutorial: Using Thymeleaf (ja) (http://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf_ja.html)

Logback

- logback - Manual (<http://logback.qos.ch/manual/index.html>)
- logback - マニュアル (http://logback.qos.ch/manual/index_ja.html)

Qiita

- Spring Boot 使い方メモ (<http://qiita.com/opengl-8080/items/05d9490d6f0544e2351a>)
- SpringBoot(with Thymeleaf)チートシート[随時更新] (<http://qiita.com/uzresk/items/31a4585f7828c4a9334f>)
- Spring-Bootの設定プロパティと環境変数 (<http://qiita.com/NewGyu/items/d51f527c7199b746c6b6>)

github

ソースコードはrubytomato/actor-search-example (<https://github.com/rubytomato/actor-search-example>)にあります。

事前準備

Java,Eclipse,Maven,MySQLのインストール、設定方法については省略します。

サンプルデータは下記の通り準備します。

- データベース: sample_db
- ユーザー : test_user
- テーブル: actor, prefecture

データベース

```
sample_db
create database if not exists sample_db;
```

ユーザー

```
create_user
create user 'test_user'@'localhost' identified by 'test_user';
```

```
grant
grant all on sample_db.* to 'test_user'@'localhost';
```

actorテーブル

actor

```
create table if not exists actor (  
  id int not null auto_increment,  
  name varchar(30) not null,  
  height smallint,  
  blood varchar(2),  
  birthday date,  
  birthplace_id smallint,  
  update_at timestamp(6) not null default current_timestamp(6) on update current_timestamp(6),  
  primary key (id)  
) engine = INNODB;
```

初期化

init

```
insert into actor (name, height, blood, birthday, birthplace_id) values  
( '丹波哲郎', 175, 'O', '1922-07-17', 13),  
( '森田健作', 175, 'O', '1949-12-16', 13),  
( '加藤剛', 173, null, '1938-02-04', 22),  
( '島田陽子', 171, 'O', '1953-05-17', 43),  
( '山口果林', null, null, '1947-05-10', 13),  
( '佐分利信', null, null, '1909-02-12', 1),  
( '緒形拳', 173, 'B', '1937-07-20', 13),  
( '松山政路', 167, 'A', '1947-05-21', 13),  
( '加藤嘉', null, null, '1913-01-12', 13),  
( '菅井きん', 155, 'B', '1926-02-28', 13),  
( '笠智衆', null, null, '1904-05-13', 43),  
( '殿山泰司', null, null, '1915-10-17', 28),  
( '渥美清', 173, 'B', '1928-03-10', 13);
```

prefectureテーブル

prefecture

```
create table if not exists prefecture (  
  id smallint not null,  
  name varchar(6) not null,  
  primary key (id)  
) engine = INNODB;
```

初期化

init

```
insert into prefecture (id, name) values  
(1, '北海道'), (2, '青森県'), (3, '岩手県'), (4, '宮城県'), (5, '秋田県'), (6, '山形県'), (7, '福島県'),  
(8, '茨城県'), (9, '栃木県'), (10, '群馬県'), (11, '埼玉県'), (12, '千葉県'), (13, '東京都'), (14, '神奈川県'),  
(15, '新潟県'), (16, '富山県'), (17, '石川県'), (18, '福井県'), (19, '山梨県'), (20, '長野県'), (21, '岐阜県'),  
(22, '静岡県'), (23, '愛知県'), (24, '三重県'), (25, '滋賀県'),  
(26, '京都府'), (27, '大阪府'), (28, '兵庫県'), (29, '奈良県'), (30, '和歌山県'),  
(31, '鳥取県'), (32, '島根県'), (33, '岡山県'), (34, '広島県'), (35, '山口県'),  
(36, '徳島県'), (37, '香川県'), (38, '愛媛県'), (39, '高知県'),  
(40, '福岡県'), (41, '佐賀県'), (42, '長崎県'), (43, '熊本県'), (44, '大分県'), (45, '宮崎県'), (46, '鹿児島県'), (47, '沖縄県');
```

アプリケーションの作成

プロジェクトの雛形を生成

アプリケーション名: actor

mavenでアプリケーションの雛形を作成

```
> mvn archetype:generate -DgroupId=com.example.actor -DartifactId=actor -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

```
> cd actor
> mvn eclipse:eclipse
```

eclipseにインポート

メニューバーの"File" -> "Import..." -> "Maven" -> "Existing Maven Projects"を選択します。
プロジェクトのディレクトリを選択し、"Finish"ボタンをクリックします。

pom.xmlの編集

pom.xml

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example.actor</groupId>
  <artifactId>actor</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>actor</name>
  <url>http://maven.apache.org</url>

+  <properties>
+    <java.version>1.8</java.version>
+    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
+  </properties>

+  <parent>
+    <groupId>org.springframework.boot</groupId>
+    <artifactId>spring-boot-starter-parent</artifactId>
+    <version>1.2.4.RELEASE</version>
+  </parent>

  <dependencies>
+  <dependency>
+    <groupId>org.springframework.boot</groupId>
+    <artifactId>spring-boot-starter-web</artifactId>
+  </dependency>
+  <dependency>
+    <groupId>org.springframework.boot</groupId>
+    <artifactId>spring-boot-starter-data-jpa</artifactId>
+  </dependency>
+  <dependency>
+    <groupId>org.springframework.boot</groupId>
+    <artifactId>spring-boot-starter-thymeleaf</artifactId>
+  </dependency>
+  <dependency>
+    <groupId>org.springframework.boot</groupId>
+    <artifactId>spring-boot-actuator</artifactId>
+  </dependency>
+  <dependency>
+    <groupId>mysql</groupId>
+    <artifactId>mysql-connector-java</artifactId>
+  </dependency>
+  <dependency>
+    <groupId>org.apache.commons</groupId>
+    <artifactId>commons-lang3</artifactId>
+    <version>3.4</version>
+  </dependency>

  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
-    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>

+  <build>
+    <pluginManagement>
+      <plugins>
+        <plugin>
+          <groupId>org.springframework.boot</groupId>
+          <artifactId>spring-boot-maven-plugin</artifactId>
+          <version>1.2.5.RELEASE</version>
+        </plugin>
+      </plugins>
+    </pluginManagement>

```

```
+ <plugins>
+   <plugin>
+     <groupId>org.apache.maven.plugins</groupId>
+     <artifactId>maven-compiler-plugin</artifactId>
+     <configuration>
+       <verbose>true</verbose>
+       <source>${java.version}</source>
+       <target>${java.version}</target>
+       <encoding>${project.build.sourceEncoding}</encoding>
+     </configuration>
+   </plugin>
+   <plugin>
+     <groupId>org.springframework.boot</groupId>
+     <artifactId>spring-boot-maven-plugin</artifactId>
+     <dependencies>
+       <dependency>
+         <groupId>org.springframework</groupId>
+         <artifactId>springloaded</artifactId>
+         <version>1.2.3.RELEASE</version>
+       </dependency>
+     </dependencies>
+   </plugin>
+   <plugin>
+     <groupId>org.codehaus.mojo</groupId>
+     <artifactId>versions-maven-plugin</artifactId>
+   </plugin>
+ </plugins>
+ </build>

</project>
```

resourcesフォルダの作成

src/main/resourcesフォルダを作成します。

- "Build Path" -> "Configure Build Path" -> "Java Buld Path" -> "Source"タブを選択する。
- "Add Folder"ボタンをクリック -> 作成した"resources"フォルダにチェックを入れる。

application.ymlの作成

src/main/resourcesフォルダ内にapplication.ymlを作成します。

application.yml

```
# EMBEDDED SERVER CONFIGURATION (ServerProperties)
server:
  port: 9000

spring:
# THYMELEAF (ThymeleafAutoConfiguration)
  thymeleaf:
    enabled: true
    cache: false
# DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
  datasource:
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost/sample_db
    username: test_user
    password: test_user
# JPA (JpaBaseConfiguration, HibernateJpaAutoConfiguration)
  jpa:
    hibernate:
      show-sql: true
      ddl-auto: update
    database-platform: org.hibernate.dialect.MySQLDialect

# INTERNATIONALIZATION (MessageSourceAutoConfiguration)
  messages:
    basename = messages
    cache-seconds = -1
    encoding = UTF-8

# ENDPOINTS (AbstractEndpoint subclasses)
  endpoints:
    enabled: true
```

logback.xmlの作成

src/main/resourcesフォルダ内にlogback.xmlを作成します。
ログの出力先フォルダを"D:/logs"に指定しました。

logback.xml


```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

  <property name="LOG_DIR" value="D:/logs" />

  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %-5level [%thread] %logger{35} - %msg %n</pattern>
    </encoder>
  </appender>
  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>${LOG_DIR}/spring-boot-sample-logger-example.log</file>
    <encoder>
      <charset>UTF-8</charset>
      <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %-5level [%thread] - %msg %n</pattern>
    </encoder>
  </appender>

  <logger name="com.example.actor" level="DEBUG" />

  <logger name="org.hibernate" level="ERROR"/>
  <logger name="org.springframework" level="INFO"/>
  <logger name="org.thymeleaf" level="INFO"/>
  <root>
    <appender-ref ref="STDOUT" />
    <appender-ref ref="FILE" />
  </root>
</configuration>
```

ビルド

この時点で動作検証を兼ねてビルドします。

```
package
> mvn package
```

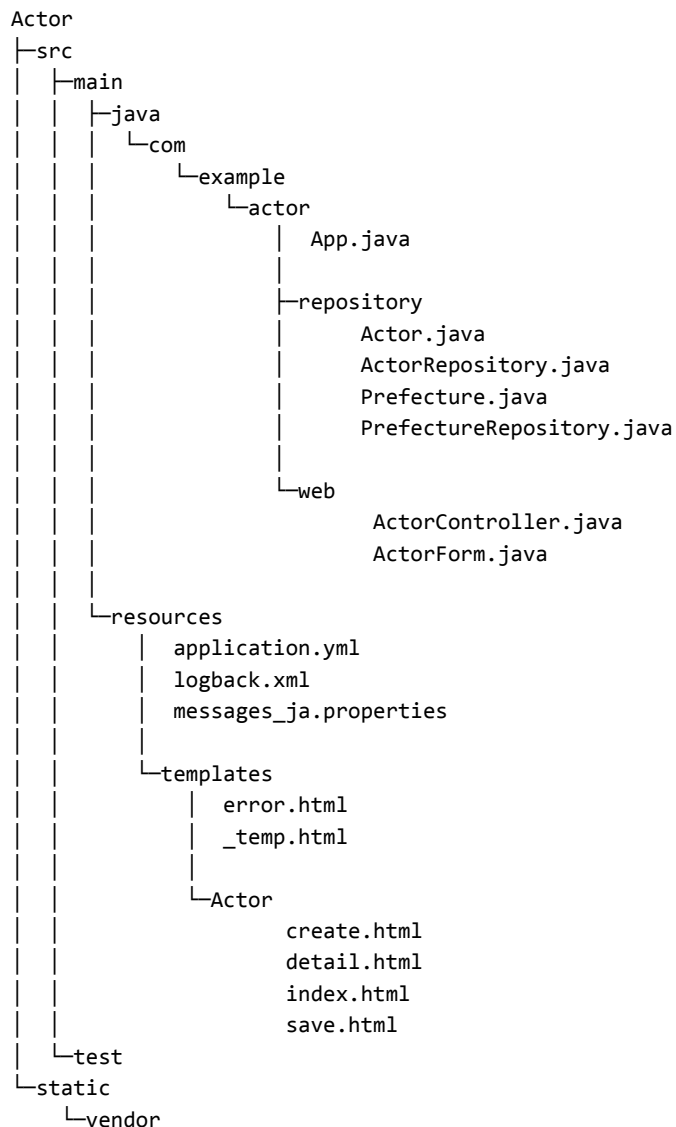
ビルドが成功したら生成したjarファイルを実行します。
コマンドプロンプトに"Hello World!"と表示されれば成功です。

```
> cd target
> java -jar actor-1.0-SNAPSHOT.jar
Hello World!
```

アプリケーションの開発

最終的には下記のディレクトリ／ファイル構成になります。

```
tree
```



App

エンドポイントとなるクラスを作成します。

すでにサンプルのApp.javaがありますので、このファイルを下記のように変更します。

App

```
package com.example.actor;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class App {

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

}
```

Repository

パッケージ: com.example.actor.repository

Prefecture

Prefectureテーブルに対応するリポジトリークラスです。

Prefecture

```
package com.example.actor.repository;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.apache.commons.lang3.builder.ToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;

@Entity
@Table(name = "prefecture")
public class Prefecture {

    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Integer id;
    @Column(name="name", nullable=false)
    private String name;

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return ToStringBuilder.reflectionToString(this, ToStringStyle.DEFAULT_STYLE);
    }
}
```

PrefectureRepository

```
package com.example.actor.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface PrefectureRepository extends JpaRepository<Prefecture, Integer> {

}
```

Actor

Actorテーブルに対応するリポジトリクラスです。

Actor

```
package com.example.actor.repository;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

import org.apache.commons.lang3.builder.ToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;

@Entity
@Table(name = "actor")
public class Actor {

    @Id
    @Column(name="id")
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer id;
    @Column(name="name", nullable=false)
    private String name;
    @Column(name="height")
    private Integer height;
    @Column(name="blood")
    private String blood;
    @Temporal(TemporalType.DATE)
    @Column(name="birthday")
    private Date birthday;
    @Column(name="birthplace_id")
    private Integer birthplaceId;
    @Temporal(TemporalType.TIMESTAMP)
    @Column(name="update_at")
    private Date updateAt;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="birthplace_id", insertable = false, updatable = false )
    private Prefecture pref;

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getHeight() {
        return height;
    }
    public void setHeight(Integer height) {
        this.height = height;
    }
    public String getBlood() {
        return blood;
    }
```

```

    }
    public void setBlood(String blood) {
        this.blood = blood;
    }
    public Date getBirthday() {
        return birthday;
    }
    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }
    public Integer getBirthplaceId() {
        return birthplaceId;
    }
    public void setBirthplaceId(Integer birthplaceId) {
        this.birthplaceId = birthplaceId;
    }
    public Date getUpdateAt() {
        return updateAt;
    }
    public void setUpdateAt(Date updateAt) {
        this.updateAt = updateAt;
    }

    public Prefecture getPref() {
        return pref;
    }
    public void setPref(Prefecture pref) {
        this.pref = pref;
    }

    @Override
    public String toString() {
        return ToStringBuilder.reflectionToString(this, ToStringStyle.DEFAULT_STYLE);
    }
}

```

ActorRepository

```

package com.example.actor.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

@Repository
public interface ActorRepository extends JpaRepository<Actor, Integer> {

    @Query("select a from Actor a where a.name like %:keyword% order by a.id asc")
    List<Actor> findActors(@Param("keyword") String keyword);

}

```

Prefectureテーブルとの結合

JoinColumnアノテーションでPrefectureテーブルとの関係を定義します。

```
@JoinColumn
```

```
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name="birthplace_id", insertable = false, updatable = false )
private Prefecture pref;
```

Controller

パッケージ: com.example.actor.web

ActorForm

ActorForm は俳優の登録フォームのパラメータがバインドされるクラスです。
フィールドに @NotNull などのアノテーションを付けてバリデーションルールを指定することができます。

ActorForm

```
package com.example.actor.web;

import java.io.Serializable;

import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

import org.apache.commons.lang3.builder.ToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;

public class ActorForm implements Serializable {

    private static final long serialVersionUID = 1330043957072942381L;

    @NotNull
    @Size(min=1, max=30)
    private String name;
    @Min(1)
    @Max(200)
    private String height;
    @Pattern(regexp = "A|B|AB|O")
    private String blood;
    @Pattern(regexp = "\\d{4}-\\d{2}-\\d{2}")
    private String birthday;
    @Min(1)
    @Max(47)
    private String birthplaceId;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getHeight() {
        return height;
    }
    public void setHeight(String height) {
        this.height = height;
    }
    public String getBlood() {
        return blood;
    }
    public void setBlood(String blood) {
        this.blood = blood;
    }
    public String getBirthday() {
        return birthday;
    }
    public void setBirthday(String birthday) {
        this.birthday = birthday;
    }
    public String getBirthplaceId() {
        return birthplaceId;
    }
    public void setBirthplaceId(String birthplaceId) {
        this.birthplaceId = birthplaceId;
    }

    @Override
    public String toString() {
        return ToStringBuilder.reflectionToString(this, ToStringStyle.DEFAULT_STYLE);
    }
}
```



```
}
```

ActorController

ActorController

```
package com.example.actor.web;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.time.format.DateTimeFormatter;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Map.Entry;

import org.apache.commons.lang3.StringUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.propertyeditors.StringTrimmerEditor;
import org.springframework.context.MessageSource;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.CollectionUtils;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.example.actor.repository.Actor;
import com.example.actor.repository.ActorRepository;
import com.example.actor.repository.Prefecture;
import com.example.actor.repository.PrefectureRepository;

@Controller
public class ActorController {
    final static Logger logger = LoggerFactory.getLogger(ActorController.class);

    @Autowired
    ActorRepository actorRepository;

    @Autowired
    PrefectureRepository prefectureRepository;

    @Autowired
    MessageSource msg;

    @InitBinder
    public void initBinder(WebDataBinder binder) {
        binder.registerCustomEditor(String.class, new StringTrimmerEditor(true));
    }

    @RequestMapping(value = "/actor", method = RequestMethod.GET)
    public String index(Model model) {
        logger.debug("Actor + index");
        List<Actor> list = actorRepository.findAll();
        if (CollectionUtils.isEmpty(list)) {
            String message = msg.getMessage("actor.list.empty", null, Locale.JAPAN);
            model.addAttribute("emptyMessage", message);
        }
        model.addAttribute("list", list);
        modelDump(model, "index");
        return "Actor/index";
    }
}
```

```

}

@RequestMapping(value = "/actor/{id}", method = RequestMethod.GET)
public ModelAndView detail(@PathVariable Integer id) {
    logger.debug("Actor + detail");
    ModelAndView mv = new ModelAndView();
    mv.setViewName("Actor/detail");
    Actor actor = actorRepository.findOne(id);
    mv.addObject("actor", actor);
    return mv;
}

@RequestMapping(value = "/actor/search", method = RequestMethod.GET)
public ModelAndView search(@RequestParam String keyword) {
    logger.debug("Actor + search");
    ModelAndView mv = new ModelAndView();
    mv.setViewName("Actor/index");
    if (StringUtils.isEmpty(keyword)) {
        List<Actor> list = actorRepository.findActors(keyword);
        if (CollectionUtils.isEmpty(list)) {
            String message = msg.getMessage("actor.list.empty", null, Locale.JAPAN);
            mv.addObject("emptyMessage", message);
        }
        mv.addObject("list", list);
    }
    return mv;
}

@RequestMapping(value = "/actor/create", method = RequestMethod.GET)
public String create(ActorForm form, Model model) {
    logger.debug("Actor + create");
    List<Prefecture> pref = prefectureRepository.findAll();
    model.addAttribute("pref", pref);
    modelDump(model, "create");
    return "Actor/create";
}

@RequestMapping(value = "/actor/save", method = RequestMethod.POST)
public String save(@Validated @ModelAttribute ActorForm form, BindingResult result, Model model) {
    logger.debug("Actor + save");
    if (result.hasErrors()) {
        String message = msg.getMessage("actor.validation.error", null, Locale.JAPAN);
        model.addAttribute("errorMessage", message);
        return create(form, model);
    }
    Actor actor = convert(form);
    logger.debug("actor:{0}", actor.toString());
    actor = actorRepository.saveAndFlush(actor);
    modelDump(model, "save");
    return "redirect:/actor/" + actor.getId().toString();
}

@RequestMapping(value = "/actor/delete/{id}", method = RequestMethod.GET)
public String delete(@PathVariable Integer id, RedirectAttributes attributes, Model model) {
    logger.debug("Actor + delete");
    actorRepository.delete(id);
    attributes.addFlashAttribute("deleteMessage", "delete ID:" + id);
    return "redirect:/actor";
}

/**
 * convert form to model.
 */
private Actor convert(ActorForm form) {
    Actor actor = new Actor();
    actor.setName(form.getName());

```

```

    if (StringUtils.isEmpty(form.getHeight())) {
        actor.setHeight(Integer.valueOf(form.getHeight()));
    }
    if (StringUtils.isEmpty(form.getBlood())) {
        actor.setBlood(form.getBlood());
    }
    if (StringUtils.isEmpty(form.getBirthday())) {
        DateTimeFormatter withoutZone = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
        LocalDateTime parsed = LocalDateTime.parse(form.getBirthday() + " 00:00:00", withoutZone);
        Instant instant = parsed.toInstant(ZoneOffset.ofHours(9));
        actor.setBirthday(Date.from(instant));
    }
    if (StringUtils.isEmpty(form.getBirthplaceId())) {
        actor.setBirthplaceId(Integer.valueOf(form.getBirthplaceId()));
    }
    actor.setUpdatedAt(new Date());
    return actor;
}

/**
 * for debug.
 */
private void modelDump(Model model, String m) {
    logger.debug(" ");
    logger.debug("Model:{", m);
    Map<String, Object> mm = model.asMap();
    for (Entry<String, Object> entry : mm.entrySet()) {
        logger.debug("key:{", entry.getKey());
    }
}
}

```

initBinder

initBinderで行っている下記の設定を行うと、リクエストパラメータが空文字の場合にそのパラメータの値をnullへ変換します。

これにより、不要なバリデーションが行われなくなります。

```

initBinder

    binder.registerCustomEditor(String.class, new StringTrimmerEditor(true));

```

このほかにも、任意の日付フォーマットの文字列パラメータをDateクラスへ変換するといった設定も行うことができます。

```

SimpleDateFormat sdf = new SimpleDateFormat("yyyy/MM/dd");
sdf.setLenient(false);
binder.registerCustomEditor(Date.class, new CustomDateEditor(sdf, true));

```

テンプレート

テンプレートファイルはtemplatesフォルダ以下に配置するのでsrc/main/resources/templatesフォルダを作成します。

また、ActorControllerが使用するビュー名を"Actor/****"としたので、対応するテンプレートファイルを配置するsrc/main/resources/templates/Actorフォルダも作成します。

tree

```
resources
├── templates
│   ├── error.html
│   ├── _temp.html
│   └── Actor
│       ├── create.html
│       ├── detail.html
│       ├── index.html
│       └── save.html
```

_temp.html

_temp.html は各テンプレートファイルで共通する部分（ヘッダーやフッター、メニューなど）を管理するテンプレートです。

部品化する要素に `th:fragment="..."` という属性で名前を付けます。

呼び出し側のテンプレートでは `th:include="..."` や `th:replace="..."` という属性で部品を呼び出します。

（このように1つのファイルに複数の部品を定義することができ、呼び出し側は部品単位で利用することができます。）

テンプレートファイル: `resources/templates/_temp.html`

_temp.html

```

<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml11-strict-thymeleaf-spring4-4.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head th:fragment="header (title)">
  <title th:text="${title}">title</title>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <link href="/vendor/bootstrap-3.3.5/css/bootstrap.min.css" th:href="@{/vendor/bootstrap-3.3.5/css/bootstrap.min.css}" />
</head>
<body>

  <div class="container">

    <div th:fragment="nav" class="row">
      <div class="col-md-12">
        <ul class="nav nav-pills">
          <li role="presentation"><a href="/actor" th:href="@{/actor}" th:utext="#{actor.nav.index}">index</a></li>
          <li role="presentation"><a href="/actor/create" th:href="@{/actor/create}" th:utext="#{actor.nav.create}">create</a></li>
        </ul>
      </div>
    </div>

    <div th:fragment="footer" class="page-header">
      <div th:utext="#{footer.text}">original footer</div>
    </div>

  </div>

  <div th:fragment="script">
    <script src="/vendor/jquery/jquery-1.11.3.js" th:src="@{/vendor/jquery/jquery-1.11.3.js}"></script>
    <script src="/vendor/bootstrap-3.3.5/js/bootstrap.js" th:src="@{/vendor/bootstrap-3.3.5/js/bootstrap.js}"></script>
  </div>
</body>
</html>

```

index.html

テンプレートファイル: resources/templates/Actor/index.html

俳優一覧を表示するテンプレートファイルです。

index.html

```

<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head th:replace="_temp :: header ('ACTOR')">
</head>
<body>

<div class="container">
  <div class="page-header">
    <h1 th:utext="#{actor.index.title}">actor.index.title</h1>
    <p th:if="${emptyMessage}" th:text="${emptyMessage}">empty message</p>
    <p th:if="${errorMessage}" th:text="${errorMessage}">error message</p>
    <p th:if="${deleteMessage}" th:text="${deleteMessage}">delete message</p>
  </div>

  <div th:replace="_temp :: nav"></div>

  <div class="row">
    <div class="col-md-6">
      <form action="/actor/search" th:action="@{/actor/search}" method="get">
        <div class="input-group">
          <input type="text" name="keyword" class="form-control" placeholder="Search for..." />
          <span class="input-group-btn">
            <input class="btn btn-default" type="submit" value="Search!" />
          </span>
        </div>
      </form>
    </div>
  </div>

  <div class="row">
    <div class="col-md-12">

      <table class="table">
        <thead>
          <tr>
            <th th:utext="#{actor.id}">id</th>
            <th th:utext="#{actor.name}">name</th>
            <th th:utext="#{actor.height}">height</th>
            <th th:utext="#{actor.blood}">blood</th>
            <th th:utext="#{actor.birthday}">birthday</th>
            <th th:utext="#{actor.birthplace}">birthplace</th>
            <th th:utext="#{actor.update_at}">update_at</th>
            <th>&nbsp;</th>
          </tr>
        </thead>
        <tbody>
          <tr th:each="item,iterStat : ${list}">
            <td>
              <a class="btn btn-default" href="/actor/${item.id}" th:href="@{/actor/{id}(id=${item.id})}" th:
            </td>
            <td th:text="${item.name}">1</td>
            <td th:text="${item.height != null}? ${item.height} : '-'>1</td>
            <td th:text="${item.blood != null}? ${item.blood} : '-'>1</td>
            <td th:text="${item.birthday != null}? ${#dates.format(item.birthday,'yyyy-MM-dd')} : '-'>1</td>
            <td th:text="(${item.birthplaceId != null}? ${item.birthplaceId} + ':' : '') + (${item.pref != null}? ${item.pref} : '')>
            <td th:text="${item.updateAt}">1</td>
            <td>
              <a class="btn btn-warning" href="/actor/delete/${item.id}" th:href="@{/actor/delete/{id}(id=${item.id})}" th:
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>

  <div th:replace="_temp :: footer"></div>

```

```
</div>
```

```
<div th:include="_temp :: script"></div>
```

```
</body>
```

```
</html>
```

detail.html

テンプレートファイル: resources/templates/Actor/detail.html

一人の俳優の情報を表示するテンプレートファイルです。

detail.html


```

<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head th:replace="_temp :: header ('ACTOR DETAIL')">
</head>
<body>

<div class="container">
<div class="page-header">
<h1 th:utext="#{actor.detail.title}">actor.detail.title</h1>
</div>

<div th:replace="_temp :: nav"></div>

<div class="row">
<div class="col-md-12">
<table class="table" th:object="${actor}">
<tr>
<th th:utext="#{actor.id}">id</th>
<td th:text="*{id}"></td>
</tr>
<tr>
<th th:utext="#{actor.name}">name</th>
<td th:text="*{name}"></td>
</tr>
<tr>
<th th:utext="#{actor.height}">height</th>
<td th:text="*{height}"></td>
</tr>
<tr>
<th th:utext="#{actor.blood}">blood</th>
<td th:text="*{blood}"></td>
</tr>
<tr>
<th th:utext="#{actor.birthday}">birthday</th>
<td th:text="*{birthday != null}? *{#dates.format(birthday,'yyyy-MM-dd')} : '-'></td>
</tr>
<tr>
<th th:utext="#{actor.birthplace}">birthplace</th>
<td th:text="*{birthplaceId} + ':' + (*{pref != null}? *{pref.name} : '(unknown)')></td>
</tr>
<tr>
<th th:utext="#{actor.update_at}">update_at</th>
<td th:text="*{updateAt}"></td>
</tr>
</table>
</div>
</div>

<div th:replace="_temp :: footer"></div>

<div th:include="_temp :: script"></div>
</body>
</html>

```

create.html

テンプレートファイル: resources/templates/Actor/create.html

俳優の情報を登録するフォームを表示するテンプレートファイルです。

create.html

```

<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head th:replace="_temp :: header ('ACTOR CREATE')">
</head>
<body>

<div class="container">
<div class="page-header">
<h1 th:utext="#{actor.create.title}">actor.create.title</h1>
</div>

<div th:replace="_temp :: nav"></div>

<div class="row">
<div class="col-md-12">

<form class="form-horizontal" role="form" action="/actor/save" th:action="@{/actor/save}" th:object="${
  <!-- name -->
  <div class="form-group">
    <label for="id_name" class="col-sm-2 control-label" th:utext="#{actor.name}">name</label>
    <div class="col-sm-10">
      <input id="id_name" class="form-control" type="text" name="name" th:field="*{name}" />
      <span th:if="${#fields.hasErrors('name')}" th:errors="*{name}" class="help-block">error!</span>
    </div>
  </div>
  <!-- height -->
  <div class="form-group">
    <label for="id_height" class="col-sm-2 control-label" th:utext="#{actor.height}">height</label>
    <div class="col-sm-10">
      <input id="id_height" class="form-control" type="text" name="height" th:field="*{height}" />
      <span th:if="${#fields.hasErrors('height')}" th:errors="*{height}" class="help-block">error!</span>
    </div>
  </div>
  <!-- blood -->
  <div class="form-group">
    <label for="id_blood" class="col-sm-2 control-label" th:utext="#{actor.blood}">blood</label>
    <div class="col-sm-10">
      <input id="id_blood" class="form-control" type="text" name="blood" th:field="*{blood}" />
      <span th:if="${#fields.hasErrors('blood')}" th:errors="*{blood}" class="help-block">error!</span>
    </div>
  </div>
  <!-- birthday -->
  <div class="form-group">
    <label for="id_birthday" class="col-sm-2 control-label" th:utext="#{actor.birthday}">birthday</label>
    <div class="col-sm-10">
      <input id="id_birthday" class="form-control" type="text" name="birthday" th:field="*{birthday}" />
      <span th:if="${#fields.hasErrors('birthday')}" th:errors="*{birthday}" class="help-block">error!</span>
    </div>
  </div>
  <!-- birthplaceId -->
  <div class="form-group">
    <label for="id_birthplaceId" class="col-sm-2 control-label" th:utext="#{actor.birthplace}">birthplace</label>
    <div class="col-sm-10">
      <select id="id_birthplaceId" class="form-control" name="birthplaceId">
        <option value="">---</option>
        <option th:each="item : ${pref}" th:value="${item.id}" th:text="${item.name}" th:selected="${item.id == pref}" />
      </select>
    </div>
  </div>

  <div class="form-group">
    <input class="btn btn-default" type="submit" value="save" />
  </div>
</form>

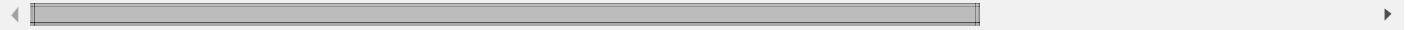
</div>

```

```
</div>

<div th:replace="_temp :: footer"></div>
</div>

<div th:include="_temp :: script"></div>
</body>
</html>
```



save.html

テンプレートファイル: resources/templates/Actor/save.html

登録結果を表示するテンプレートファイルです。

save.html

```

<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head th:replace="_temp :: header ('ACTOR SAVE')">
</head>
<body>

<div class="container">
<div class="page-header">
<h1 th:utext="#{actor.save.title}">actor.save.title</h1>
</div>

<div th:replace="_temp :: nav"></div>

<div class="row">
<div class="col-md-12">
<table class="table" th:object="${actor}">
<tr>
<th th:utext="#{actor.id}">id</th>
<td th:text="*{id}"></td>
</tr>
<tr>
<th th:utext="#{actor.name}">name</th>
<td th:text="*{name}"></td>
</tr>
<tr>
<th th:utext="#{actor.height}">height</th>
<td th:text="*{height}"></td>
</tr>
<tr>
<th th:utext="#{actor.blood}">blood</th>
<td th:text="*{blood}"></td>
</tr>
<tr>
<th th:utext="#{actor.birthday}">birthday</th>
<td th:text="*{birthday != null}? *{#dates.format(birthday,'yyyy-MM-dd')} : '-'"></td>
</tr>
<tr>
<th th:utext="#{actor.birthplace}">birthplace</th>
<td th:text="*{birthplaceId} + ':' + (*{pref != null}? *{pref.name} : '(unknown)')"></td>
</tr>
<tr>
<th th:utext="#{actor.update_at}">update_at</th>
<td th:text="*{updateAt}"></td>
</tr>
</table>
</div>
</div>

<div th:replace="_temp :: footer"></div>

<div th:include="_temp :: script"></div>
</body>
</html>

```

エラーページ

デフォルトではThymeleafのエラーページが表示されます。
 下図は存在しないURLにアクセスしたときに表示されるエラーページです。

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

(https://qiita-

Mon Jul 27 20:51:34 JST 2015

There was an unexpected error (type=Not Found, status=404).

No message available

image-store.s3.amazonaws.com/0/22772/6eb53659-a5b5-7963-f0a6-f3e85bfa479d.png)

error.html

任意のエラーページを表示したい場合は、templatesフォルダ直下に error.html という名前のテンプレートファイルでエラーページを用意します。

今回は下記のような簡単なエラーページを用意しました。

存在しないURLにアクセスするとこの error.html が表示されます。

error.html

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head th:replace="_temp :: header ('ERROR')">
</head>
<body>

    <div class="container">
        <div class="page-header">
            <h1>default error page</h1>
        </div>
        <div th:replace="_temp :: footer"></div>
    </div>

    <div th:include="_temp :: script"></div>
</body>
</html>
```

HTTPステータスコード別にエラーページを用意したい場合

今回のアプリケーションでは使用していませんが、下記のようなコントローラーを作成してhttpステータスコード毎に任意のエラーページを表示することができます。

この例ではhttpステータスが404と500のときに指定するエラーページを表示します。

ErrorController.java

```

package com.example.actor.web;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.context.embedded.ConfigurableEmbeddedServletContainer;
import org.springframework.boot.context.embedded.EmbeddedServletContainerCustomizer;
import org.springframework.boot.context.embedded.ErrorPage;
import org.springframework.context.annotation.Bean;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class ErrorController {
    final static Logger logger = LoggerFactory.getLogger(ErrorController.class);

    @Bean
    public EmbeddedServletContainerCustomizer containerCustomizer() {
        return new EmbeddedServletContainerCustomizer() {
            @Override
            public void customize(ConfigurableEmbeddedServletContainer container) {
                ErrorPage error404 = new ErrorPage(HttpStatus.NOT_FOUND, "/error/404");
                ErrorPage error500 = new ErrorPage(HttpStatus.INTERNAL_SERVER_ERROR, "/error/500");
                container.addErrorPages(error404, error500);
            }
        };
    }

    @RequestMapping("/error/404")
    public String error404() {
        return "Error/404";
    }

    @RequestMapping("/error/500")
    public String error500() {
        return "Error/500";
    }
}

```

エラーページのテンプレートファイルを作成します。

(エラーページを配置するフォルダやファイル名は任意です。)

- resources/templates/Error/404.html
- resources/templates/Error/500.html

エラーページの内容を確認する場合は、"http://localhost:9000/error/404"にアクセスすることで表示内容を確認することができます。(http://localhost:9000/error/404%22%E3%81%AB%E3%82%A2%E3%82%AF%E3%82%BB%E3%82%B9%E3%81%99%E3%82%8B%E3%81%93%E3%81%A8%E3%81%A7%E8%A1%A8%E7%A4%BA%E5%86%85%E5%AE%B9%E3%82%92%E7%A2%BA%E8%AA%8D%E3%81%99%E3%82%8B%E3%81%93%E3%81%A8%E3%81%8C%E3%81%A7%E3%81%8D%E3%81%BE%E3%81%99%E3%80%82)

メッセージリソース

メッセージリソースはresourcesフォルダ以下に配置します。

ファイル名のベースはapplication.ymlで指定できます。

今回はベース名を messages としていますので、日本語のメッセージリソースファイルは messages_ja.properties となります。

src/main/resources/messages_ja.propertiesを作成します。

```
messages_ja.properties

# page title
actor.index.title = \u4FF3\u512A
actor.detail.title = \u8A73\u7D30
actor.create.title = \u65B0\u898F
actor.save.title = \u767B\u9332

# field label
actor.id = id
actor.name = \u540D\u524D
actor.height = \u8EAB\u9577
actor.blood = \u8840\u6DB2\u578B
actor.birthday = \u8A95\u751F\u65E5
actor.birthplace = \u51FA\u8EAB\u5730
actor.update_at = \u66F4\u65B0\u65E5\u6642

# navi menu label
actor.nav.index = \u4E00\u89A7
actor.nav.create = \u65B0\u898F

# footer message
footer.text = spring boot sample application

# action message
actor.list.empty = list empty
actor.validation.error = validation error
```

静的リソース

静的リソース(css,js,img等)はプロジェクトフォルダ(/actor)の直下のstaticフォルダ以下に配置します。
このアプリケーションではbootstrapとjQueryを使用しますので、それらのファイルをstatic以下にコピーします。

- bootstrapは、"static/vendor/bootstrap-3.3.5"に配置しました。
- jQueryは、"static/vendor/jquery"に配置しました。

```
tree
Actor
├─src
├─static
│  └─vendor
│     ├──bootstrap-3.3.5
│     │  ├──css
│     │  ├──fonts
│     │  └─js
│     └─jquery
└─target
```

実行する

```
> mvn spring-boot:run
```

アプリケーションが起動したら下記のURLにアクセスします。
俳優の一覧が表示されれば成功です。

`http://localhost:9000/actor` (`http://localhost:9000/actor`)

Actuator

Actuatorでspring bootの状態をブラウザから確認することができます。
機能を有効にするには`application.yml`で `endpoints.enabled` を`true`に設定します。

ID	endpoint
autoconfig	<code>http://localhost:9000/autoconfig</code> (<code>http://localhost:9000/autoconfig</code>)
beans	<code>http://localhost:9000/beans</code> (<code>http://localhost:9000/beans</code>)
configprops	<code>http://localhost:9000/configprops</code> (<code>http://localhost:9000/configprops</code>)
dump	<code>http://localhost:9000/dump</code> (<code>http://localhost:9000/dump</code>)
env	<code>http://localhost:9000/env</code> (<code>http://localhost:9000/env</code>)
health	<code>http://localhost:9000/health</code> (<code>http://localhost:9000/health</code>)
metrics	<code>http://localhost:9000/metrics</code> (<code>http://localhost:9000/metrics</code>)
mappings	<code>http://localhost:9000/mappings</code> (<code>http://localhost:9000/mappings</code>)
trace	<code>http://localhost:9000/trace</code> (<code>http://localhost:9000/trace</code>)

Part V. Spring Boot Actuator: Production-ready features (<http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#production-ready>)

メモ

Spring loaded

reloadが効かない点が幾つかあったので列挙します。（もしかすると設定が足りないなどの原因があるかもしれません。）

- Eclipse内からmavenコマンドでアプリケーションを実行した場合、`terminated`(赤い四角)ボタンを押しもjavaプロセスが終了せずに残留する。この為コマンドプロンプトから`mvn`コマンドで実行し、終了は`Ctrl+C`キーを押す。
- `Message`リソースファイルの変更が反映されない。
- `@ModelAttribute` アノテーションの`value`の変更が反映されない。

Thymeleaf

コレクションを扱う方法



コントローラー側で下記のコレクションをセットした場合

```
Iterable<Category> result = categoryService.findAll();
model.addAttribute("result", result);
```

```
<table class="table" th:if="${result}">
  <caption th:text="${result.size()}">result</caption>
  <thead>
    ...省略...
  </thead>
  <tbody>
    <tr th:each="category,status : ${result}">
      ...省略...
    </tr>
  </tbody>
</table>
```

- コレクションが持つメソッドを呼び出すことができます。この例ではsize()メソッドでコレクションの件数を取得しています。

🔗この記事は以下の記事からリンクされています

-  **Thymeleafを使用した入力フォームのサンプルコード** ([/rubytomato@github/items/387d46ea34eb92071065](https://github.com/rubytomato/items/387d46ea34eb92071065)) からリンク 5ヶ月前
-  **Play Framework (java)で簡単な検索アプリケーションを開発する** ([/rubytomato@github/items/17aed34b6294b4f205ea](https://github.com/rubytomato/items/17aed34b6294b4f205ea)) からリンク 4ヶ月前

PR 社内メールでの情報共有をなくそう - Qiita:Team (https://teams.qiita.com/?utm_content=48&utm_medium=text&utm_source=qiita)

あなたもコメントしてみませんか :)

[ユーザー登録\(無料\) \(/signup?redirect_to=%2Frubytomato%40github%2Fitems%2Fe4fda26faddbcfd84d16%23comments\)](/signup?redirect_to=%2Frubytomato%40github%2Fitems%2Fe4fda26faddbcfd84d16%23comments)

すでにアカウントを持っている方はログイン (/login?redirect_to=%2Frubytomato%40github%2Fitems%2Fe4fda26faddbcfd84d16%23comments)