1. 개요

이 예제에서는 Spring에서 경로 변수를 선택적으로 만드는 방법을 배웁니다. 먼저 Spring 이 핸들러 메서드에서 *@PathVariable* 매개 변수를 바인딩하는 방법을 설명 합니다 . 그런 다음 다른 Spring 버전에서 경로 변수를 선택적으로 만드는 다른 방법을 보여줄 것입니다.

경로 변수에 대한 간략한 개요는 Spring MVC 기사를 참조하십시오 .

2. Spring이 *@PathVariable* 매개 변수를 바인딩하는 방법

기본적으로 Spring은 핸들러 메소드에서 *@PathVariable* 로 어노테이션이 달린 모든 매개 변수를 URI 템플릿의 해당 변수와 결합하려고 시도합니다 . Spring이 실패하면 해당 핸들러 메서드에 요청을 전달하지 않습니다.

예를 들어, *id* 경로 변수를 선택적 으로 만들려고 시도하는 (실패한) 다음 *getArticle* 메소드를 고려하십시오 .

```
@RequestMapping(value = {"/article", "/article/{id}"})
public Article getArticle(@PathVariable(name = "id") Integer articleId) {
    if (articleId != null) {
        //...
    } else {
        //...
    }
}
```

여기서 *getArticle* 메소드는 */ article* 및 */ article / {id}* 모두에 대한 요청을 처리해야합니다 . Spring은 *articleId* 매개 변수가있는 경우 *id* 경로 변수 에 바인딩하려고 시도합니다 .

예를 들어 */ article / 123에* 요청을 보내면 *articleId* 값이 *123으로* 설정 됩니다.

반면에 */ article에* 요청을 보내면 Spring은 다음 예외로 인해 상태 코드 500을 반환합니다.

```
org.springframework.web.bind.MissingPathVariableException:
  Missing URI template variable 'id' for method parameter of type Integer
```

이는 Spring이 *id* 가 누락 되어 *articleId* 매개 변수에 대한 값을 설정할 수 없기 때문 입니다.

따라서 다음 섹션에서 볼 수 있듯이 해당 경로 변수가없는 경우 특정 *@PathVariable* 매개 변수의 바인딩을 무시하도록 Spring에 알리는 방법이 필요 합니다.

3. 경로 변수를 선택적으로 만들기

3.1. *@PathVariable* 의 *필수* 속성 사용

Spring 4.3.3부터 *@PathVariable* 어노테이션은 경로 변수가 핸들러 메소드에 필수인지 여부를 표시하는 *데 필요한* 부울 속성을 정의합니다 .

예를 들어 다음 버전의 *getArticle* 은 *필수* 속성을 사용 합니다.

```
@RequestMapping(value = {"/article", "/article/{id}"})
public Article getArticle(@PathVariable(required = false) Integer articleId) {
   if (articleId != null) {
       //...
   } else {
       //...
   }
}
```

때문에 *필요한* 속성이 *잘못된* 경우 생성, Spring 불평하지 않습니다 *아이디* 경로 변수가 요청에 전송되지 않습니다. 즉, Spring은 *articleId* 가 전송되면 *id로* 설정하고 그렇지 않으면 *null을 설정* 합니다.

반면에 *required* 가 *true* 이면 *ID* 가 누락 된 경우 Spring에서 예외가 발생 합니다.

3.2. 사용하여 *선택적* 매개 변수 유형을

다음 구현은 JDK 8의 *Optional* 클래스 와 함께 Spring 4.1 이 *articleId* 를 선택적 으로 만드는 또 다른 방법을 제공 하는 방법을 보여줍니다 .

```
@RequestMapping(value = {"/article", "/article/{id}"})
public Article getArticle(@PathVariable Optional<Integer> optionalArticleId) {
    if (optionalArticleId.isPresent()) {
        Integer articleId = optionalArticleId.get();
        //...
    } else {
        //...
    }
}
```

여기서 Spring은 *id* 값을 보유하기 위해 *Optional <Integer>* 인스턴스 인 *optionalArticleId* 를 생성합니다 . 경우 *ID* 가 존재, *optionalArticleId* 는 달리, 그 값을 바꿈됩니다 *optionalArticleId* 이 랩합니다 *널* 값입니다. 그런 다음 *Optional* 의 *isPresent (), get ()* 또는 *orElse ()* 메서드를 사용하여 값을 처리 할 수 있습니다.

3.3. 사용 *Map* 매개 변수 유형을

Spring 3.2 이후로 사용 가능한 선택적 경로 변수를 정의하는 또 다른 방법 은 @ *PathVariable* 매개 변수에 대한 *M ap* 를 사용하는 것입니다 .

```
@RequestMapping(value = {"/article", "/article/{id}"})
public Article getArticle(@PathVariable Map<String, String> pathVarsMap) {
    String articleId = pathVarsMap.get("id");
    if (articleId != null) {
        Integer articleIdAsInt = Integer.valueOf(articleId);
        //...
    } else {
        //...
    }
}
```

이 예제에서 *Map <String, String> pathVarsMap* 매개 변수는 URI에있는 모든 경로 변수를 키 / 값 쌍으로 수집합니다. 그런 다음 *get ()* 메서드를 사용하여 특정 경로 변수를 *얻을* 수 있습니다.

Spring은 경로 변수의 값을 *String* 으로 추출하기 때문에 *Integer.valueOf ()* 메서드를 사용하여 *Integer* 로 변환했습니다 .

3.4. 두 개의 처리기 방법 사용

레거시 Spring 버전을 사용하는 경우 *getArticle* 핸들러 메서드를 두 개의 메서드로 분할 할 수 있습니다 .

첫 번째 메소드는 */ article / {id}* 에 대한 요청을 처리합니다 .

```
@RequestMapping(value = "/article/{id}")
public Article getArticle(@PathVariable(name = "id") Integer articleId) {
    //...
}
```

두 번째 방법은 */ article* 에 대한 요청을 처리합니다 .

```
@RequestMapping(value = "/article")
public Article getDefaultArticle() {
    //...
}
```