5. MVC-3

1. Get 방식과 Post 방식 처리
2. @ModelAttribut
3. 리다이렉트(redirect) 키워드
4. JSON 데이터를 생성
5. 폼 데이터 값 검증

1. Get 방식과 Post 방식 처리

- **@RequestMapping에서 요청을 받을 때 Get방식과 Post 방식을 구분 할 수 있음**

```
@RequestMapping(method=RequestMethod.GET, value="/studet")
public String goStudent(HttpServletRequest request, Model model){
    String id=request.getParameter("id");
    model.addAttribute("id",id);
    return "student/studentId";
}
```

```
<form action="student" method="get">
student id : <input type="text" name="id"><br>
<input type="submit" value="전송">
</form>
```

```
<form action="student" method="post">
student id : <input type="text" name="id"><br>
<input type="submit" value="전송">
</form>
```

2. @ModelAttribute

- 커맨드 객체의 이름을 개발자가 변경할 수 있음

```
@RequestMapping("/studentView")
public String studentView(StudentInformation studentInformation){
    return "studentView";
}
```

```
<body>
이름 : ${studentInformation.name} <br />
나이 : ${studentInformation.age} <br />
학년 : ${studentInformation.classNum} <br />
반 : ${studentInformation.gradeNum}
</body>
```

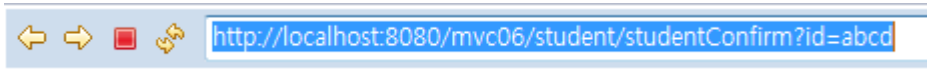
```java
@RequestMapping("/studentView")
public String studentView(@ModelAttribute("studentInfo") StudentInformation studentInformation){
    return "studentView";
}
```

```jsp
<body>
이름 : ${studentInfo.name} <br />
나이 : ${studentInfo.age} <br />
학년 : ${studentInfo.classNum} <br />
반 : ${studentInfo.gradeNum}
</body>
```

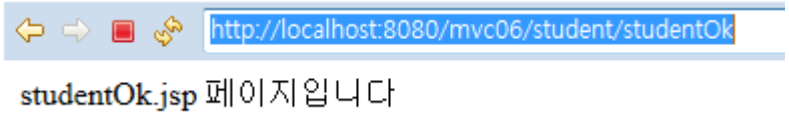
3. 리다이렉트(redirect 키워드)

- 다른 페이지로 이동할 때 사용

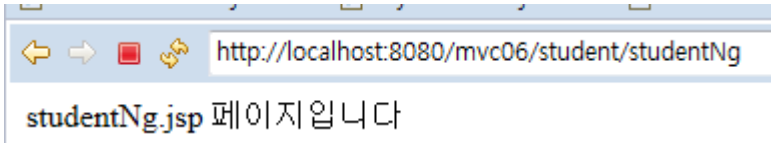
```
@RequestMapping("/studentConfirm")
public String StudentRedirect(HttpServletRequest
httpServletResource,
        Model model){
    String id=httpServletResource.getParameter("id");
    if(id.equals("abcd")){
        return "redirect:studentOk";
    }
    return "redirect:studentNg";
}
```




```
@RequestMapping("/studentOk")
public String studentOk(Model model){
    return "student/studentOk";
}
```




```
@RequestMapping("/studentNg")
public String studentNg(Model model){
    return "student/studentNg";
}
```



4. JSON 데이터를 생성하는 경우

- **JSON(JavaScript Object Notation)**
  - pom.xml 에 Jackson-databind 라이브러리 추가

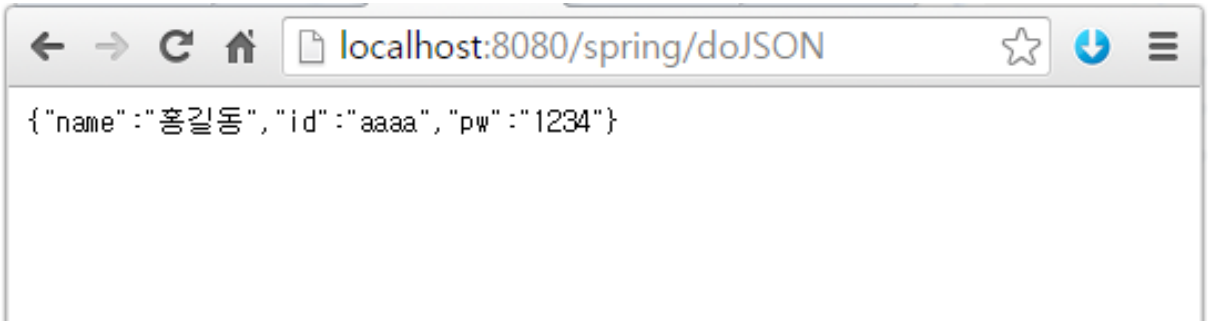
```
<dependency>
	<groupId>com.fasterxml.jackson.core</groupId>
	<artifactId>jackson-databind</artifactId>
	<version>2.7.0</version>
</dependency>
```

  - 컨트롤로 작성

```
@RequestMapping("/doJSON")
public @ResponseBody Member doJSON(){
	Member member=new Member("홍길동","aaaa","1234");
	return member;
}
```

JSON(JavaScript Object Notation)-2

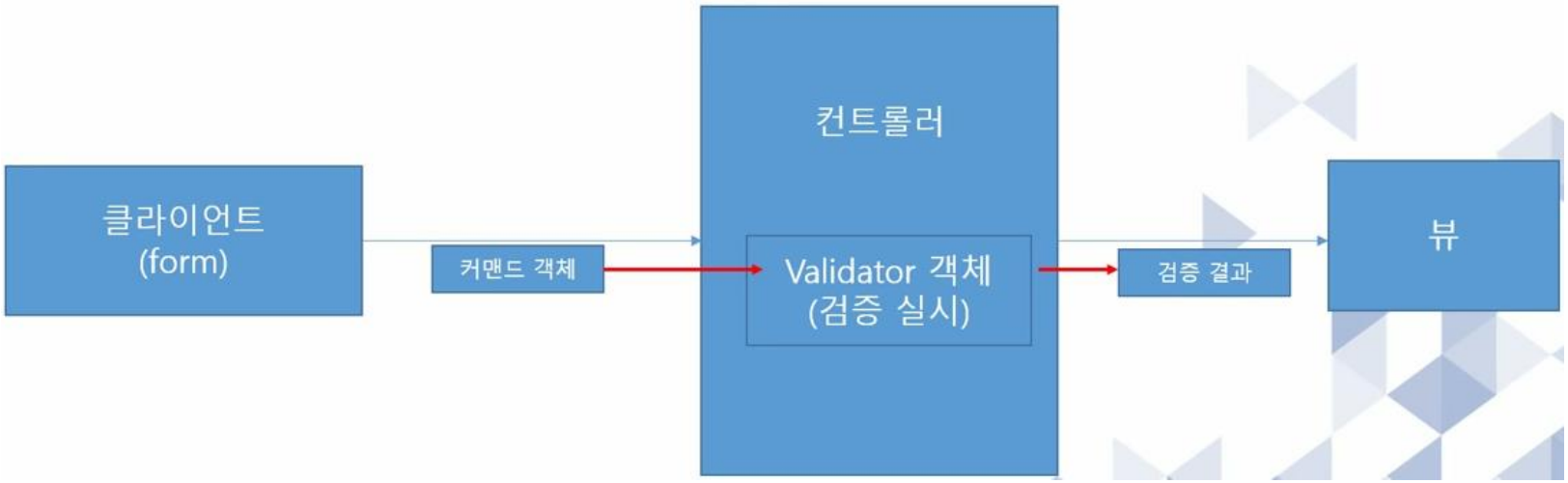
- @ResponseBody 어노테이션
  - 컨트롤러 메소드 리턴 값이 View에 출력되지 않고 HTTP Response Body에 직접 쓰여지게 됨
- 실행
  - 반드시 브라우저(Chrom에서 실행, 인터넷 익스플로러 버전에 따라 실행)에서 실행



4. 폼 데이터값 검증

Validator를 이용한 검증

- 폼에서 전달되는 데이터를 커맨드 객체에 담아 컨트롤 객체에 전달할 때 커맨드 객체의 유효성 검사
- 유효성 검사 방법
  - client 검사 방법 :Javascript
  - 서버 검사 방법: Validator 인터페이스 이용



Validator를 이용한 검증(2)

```
@RequestMapping("/student/create")
public String studentCreat(@ModelAttribute("student")
        Student student,BindingResult result){
        String page="createDonePage";
        StudentValidator validator=new StudentValidator();
        validator.validate(student, result);
        if(result.hasErrors()){
                page="createPage";
        }
        return page;
}
```

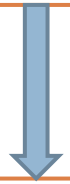
```
public class StudentValidator implements Validator {
	@Override
	public boolean supports(Class<?> arg0) {
		return Student.class.isAssignableFrom(arg0);
	}

	@Override
	public void validate(Object obj, Errors errors) {
	System.out.println("Validate()");
		Student student=(Student)obj;
		String studentName=student.getName();
		if(studentName==null||studentName.trim().isEmpty()){
			System.out.println("studentName is null or empty");
			errors.rejectValue("name", "troble");
		}
		int studentId=student.getId();
		if(studentId==0){
			System.out.println("studentId is 0");
			errors.reject("id", "troble");
		}
	}
}
```

ValidationUtils 클래스

- validate()메소드를 좀 더 편리하게 사용할 수 있도록 고안

```
String studentName=student.getName();
if(studentName==null||studentName.trim().isEmpty()){
        System.out.println("studentName is null or empty");
        errors.rejectValue("name", "troble");
}
```



```
ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "troble");
```

@Valid와 @InitBinder

- 사용자가 validate() 메소드를 호출하지 않고 스프링 프레임워크가 호출하는 방법

의존 추가

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>4.2.0.Final</version>
</dependency>
```

@InitBinder 추가

```
@InitBinder
protected void initBinder(WebDataBinder binder){
    binder.setValidator(new StudentValidator());
}
```

@Valid와 @InitBinder

```
@RequestMapping("/student/create")
public String studentCreat(@ModelAttribute("student") Student student,BindingResult result){
        String page="createDonePage";
        StudentValidator validator=new StudentValidator();
        validator.validate(student, result);
        if(result.hasErrors()){
                page="createPage";
        }
        return page;
}
```

```
@RequestMapping("/student/create")
public String studentCreat(@ModelAttribute("student") @Valid Student
                student,BindingResult result){
        String page="createDonePage";
        if(result.hasErrors()){
                page="createPage";
        }
        return page;
}
@InitBinder
protected void initBinder(WebDataBinder binder){
        binder.setValidator(new StudentValidator());
}
```