오늘은 폼(Form) 데이터를

처리하고 검증하는 법을 학습하겠습니다

1. HomeController.java

```
package com.example.pracmvc;

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.example.pracmvc.member.Member;

/**
 * Handles requests for the application home page.
 */
@Controller
public class HomeController {

	private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

	/**
	 * Simply selects the home view to render by returning its name.
	 */
	@RequestMapping(value = "/", method = RequestMethod.GET)
	public String home(Locale locale, Model model) {
		logger.info("Welcome home! The client locale is {}.", locale);

		Date date = new Date();
		DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);

		String formattedDate = dateFormat.format(date);

		model.addAttribute("serverTime", formattedDate );

		return "home";
	}

	@RequestMapping(value = "board/confirmId")
	public String confirmId(HttpServletRequest httpServletRequest, Model model) {
		String id = httpServletRequest.getParameter("id");
		String pw = httpServletRequest.getParameter("pw");
		model.addAttribute("id", id);
		model.addAttribute("pw", pw);
		return "board/confirmId";
	}

	@RequestMapping("board/checkId")
	public String checkId(@RequestParam("id") String id, @RequestParam("pw") int pw, Model model) {
		model.addAttribute("identify", id);
		model.addAttribute("password", pw);
		return "board/checkId";
	}
```

```
//	@RequestMapping("/member/join")
//	public String joinData(@RequestParam("name") String name, @RequestParam("id") String id,
//			@RequestParam("pw") String pw, @RequestParam("email") String email, Model model) {
//
//		Member member = new Member();
//		member.setName(name);
//		member.setId(id);
//		member.setPw(pw);
//		member.setEmail(email);
//
//		model.addAttribute("memberInfo", member);
//
//		return "member/join";
//	}

	@RequestMapping("/member/join")
	public String joinData(@ModelAttribute("member") Member member, BindingResult result) {
		MemberValidator validator = new MemberValidator();
		validator.validate(member, result);
		if(result.hasErrors()) {
			return "member/error";
		}
		return "member/join";
	}

}
```

2. confirmId.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>
 ID : ${id} <br />
 PW : ${pw}
</body>
</html>
```

3. checkId.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>
 ID : ${identify} <br />
 PW : ${password}
</body>
</html>
```

먼저 HttpServletRequest 클래스를

이용해서 폼 데이터를 처리할 수 있습니다

위의 코드에서 confirmId() 메소드는

클라이언트로부터 id와 pw를 받아

Model에 값을 세팅하는 것을 볼 수 있네요~

두번째로 @RequestParam 어노테이션을

이용해서 폼 데이터를 처리할 수도 있는데요

위의 코드에서 checkId() 메소드가 해당됩니다

마지막으로 커맨드 객체를 이용한 경우를 보면

위의 코드에서 주석 처리한 부분을

깔끔하게 처리하고 있는 joinData() 메소드입니다

주석 처리한 부분에 비해 코드 양이 현저히 줄었네요

이것은 아래와 같은 데이터 클래스가 필요합니다

4. Member.java

```java
package com.example.pracmvc.member;

public class Member {

	private String name;
	private String id;
	private String pw;
	private String email;

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getId() {
		return id;
	}
	public void setId(String id) {
		this.id = id;
	}
	public String getPw() {
		return pw;
	}
	public void setPw(String pw) {
		this.pw = pw;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}

}
```

위처럼 폼 데이터를 커맨드 객체로

컨트롤러에 전달되면 Validator를 이용해서

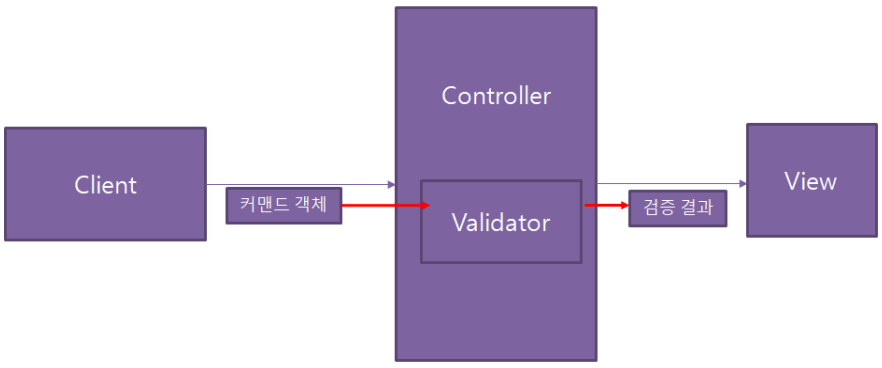
객체의 유효성을 검사할 수 있습니다

참고로 javascript로도 검사할 수 있는데

이것은 클라이언트에서 검사하는 방식이고

Validator를 이용한 방법은

서버에서 검사하는 방식입니다



5. MemberValidator.java

```java
package com.example.pracmvc;

import org.springframework.validation.Errors;
import org.springframework.validation.ValidationUtils;
import org.springframework.validation.Validator;

import com.example.pracmvc.member.Member;

public class MemberValidator implements Validator {

	@Override
	public boolean supports(Class<?> clazz) {
		// TODO Auto-generated method stub
		return Member.class.isAssignableFrom(clazz);
	}

	@Override
	public void validate(Object target, Errors errors) {
		// TODO Auto-generated method stub
		System.out.println("validate()");
		Member member = (Member)target;

		String memberName = member.getName();
		if(memberName == null || memberName.trim().isEmpty()) {
			System.out.println("memberName is null or empty");
			//errors.rejectValue("name", "trouble");
			ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "trouble");
		}

	}

}
```

Validator 클래스는 Validator 인터페이스를

구현하고 있고 validate() 메소드 사용합니다

ValidationUtils 클래스는 validate() 메소드를

좀 더 편리하게 사용할 수 있도록

고안된 클래스입니다

Validator는 이 방법 외에도

@Valid와 @InitBinder를 이용할 수 있는데요

여기서는 생략하겠습니다

6. RedirectController.java

```java
package com.example.pracmvc;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class RedirectController {

	@RequestMapping("/studentConfirm")
	public String studentRedirect(HttpServletRequest httpServletRequest, Model model){

		String id = httpServletRequest.getParameter("id");
		if(id.equals("abc")) {
			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";
	}

}
```

7. studentOk.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Insert title here</title>
</head>
<body>
studentOk.jsp
</body>
</html>
```

8. studentNg.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
```

```
<title>Insert title here</title>
</head>
<body>
studentNg.jsp
</body>
</html>
```