Model 객체는 Controller 에서 생성된 데이터를 담아 View 로 전달할 때 사용하는 객체이다.
※ Servlet의 request.setAttribute() 와 비슷한 역할을 함
addAttribute("key", "value") 메서드를 이용해 view에 전달할 데이터를 key, value형식으로 전달할 수 있다.

사용 예시

```
package com.mystudy.coco.controller;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
@Controller
public class TestController {
@GetMapping("/test")
public void testMethod(Model model) {
String msg = "model test";
model.addAttribute("value", msg);
}
}
```

```
<h2> Test Method : ${value} </h2>
```

@ModelAttribute는 HTTP Body 내용과 HTTP 파라미터의 값들을 Getter, Setter, 생성자를 통해 주입하기 위해 사용한다.
일반 변수의 경우 전달이 불가능하기 때문에 model 객체를 통해서 전달해야 한다.
※@ModelAttribute("파라미터명")

아래부터는 님의 글을 가져왔다.

Spring MVC에서 @ModelAttribute을 메소드의 파라미터로 사용할경우 내부적으로 돌아가는 메커니즘을 알아보자.

test메소드안에 @ModelAttribute("test") 를 파라미터로 가지고있는 형태이다. 결과는 true,false를 반환하게 해놓았다.

```
@Controller
public class TestController {
@Autowired
private TestService service;
@GetMapping("/test")
public String test(@ModelAttribute("test") Test test, Model model) {
model.addAttribute("result", service.getResult(test));
return "test";
}
}
```

아래를 보면 3개의 파라미터와 name값이 Test클래스의 인스턴스명들과 일치한다.
(자동으로 binding 시키려면 name값을 일치시켜줘야한다.)

view

@ModelAttribute 선언 후 자동으로 진행되는 작업들은 다음과 같다.

1. 파라미터로 넘겨 준 타입의 오브젝트를 자동으로 생성한다.
내 코드에서는 Test 클래스의 객체 test를 자동으로 생성한다. 이때 @ModelAttribute가 지정되는 클래스는 getter,setter가 있는 Beans 클래스여야 한다.

2.생성된 오브젝트(test) HTTP로 넘어 온 값들을 자동으로 바인딩한다.
http://localhost:3001/biz/test?testId=1&testName=1&testPwd=1
이렇게 들어오는 값들이 Test클래스의 testId, testName, testPwd 각각 해당변수의 setter를 통해서 해당 멤버 변수에로 binding된다.

3.마지막으로 @ModelAttribute 어노테이션이 붙은 객체가 (Test객체) 자동으로 Model객체에 추가되고 View단으로 전달된다.
@ModelAttribute("test") Test test 에서 어노테이션 괄호안의 test 값을 통해 view단에서 데이터들을 호출할수있다.