@RequestParam

@RequestParam 어노테이션을 이용하면 Servlet 요청 파라미터(ex. 쿼리 파라미터, form 데이터)를 컨트롤러 메소드의 인자로 바인딩할 수 있습니다. 스프링 프레임워크를 사용할 때 가장 편리한 기능 중 하나가 파라미터를 자동으로 수집하는 것입니다. 매번 request.getParameter로 데이터를 파싱하던 노동에서 벗어날 수 있게 됩니다.

@RequestParam(name = "name")

파라미터 name을 지정하여 파라미터로 전달된 데이터들 중 지정한 이름의 데이터를 파싱합니다. 매개변수로 지정한 변수의 타입이 String이 아닌 경우 자동으로 형변환이 이루어집니다.

```
@RequestMapping(value = "/login1")
public String login1(@RequestParam(name = "userid") String id, @RequestParam("passwd") String pw) {
	return "userid: "+id+", passwd: "+pw;
}
```

```
http://localhost:8092/testmapping/login1?userid=aaa&passwd=1234
----
userid: aaa, passwd: 1234
```

@RequestParam

파라미터로 전달되는 name과 매개변수의 이름을 일치시키면 name 속성을 지정하지 않아도 자동으로 데이터를 파싱해서 동일한 이름의 변수에 저장합니다.

```
@RequestMapping(value = "/login2")
public String login2(@RequestParam String userid, @RequestParam String passwd) {
	return "userid: "+userid+", passwd: "+passwd;
}
```

```
http://localhost:8092/testmapping/login2?userid=aaa&passwd=1234
----
userid: aaa, passwd: 1234
```

@RequestParam 생략

위의 예제와 같이 파라미터로 전달되는 name과 매개변수의 이름이 일치하면 @RequestParam을 적어주지 않아도 자동으로 데이터를 파싱해서 동일한 이름의 변수에 저장합니다.

```
@RequestMapping(value = "/login3")
public String login3(String userid, String passwd) {
	return "userid: "+userid+", passwd: "+passwd;
}
```

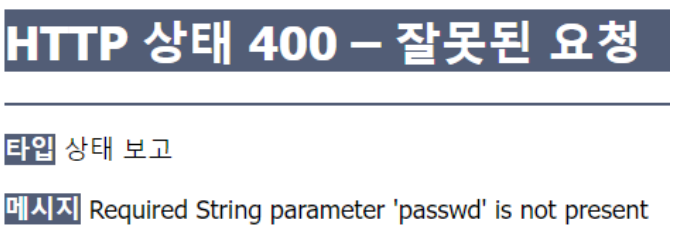
```
http://localhost:8092/testmapping/login3?userid=aaa&passwd=1234
----
userid: aaa, passwd: 1234
```

@RequestParam 속성

required 속성 - 필수 파라미터 해제

@RequestParam을 명시한 메소드 파라미터는 기본적으로 필수 파라미터가 되어 해당 이름으로 넘어오는 파라미터가 존재하지 않는 경우 예외가 발생합니다.

```
http://localhost:8092/testmapping/login2?userid=aaa
```



이 경우 다음과 같이 required 속성으로 false를 지정해 주면 해당 파라미터를 필수가 아닌 옵션이 되도록 설정 가능합니다.

```
@RequestMapping(value = "/login4")
public String login4(@RequestParam String userid, @RequestParam(required = false) String passwd) {
	return "userid: "+userid+", passwd: "+passwd;
}
```

```
http://localhost:8092/testmapping/login4?userid=aaa
----
userid: aaa, passwd: null
```

defaultValue 속성 - 디폴트 값 설정

defaultValue 속성을 지정하여 해당 파라미터에 대한 디폴트 값을 설정할 수 있습니다. 이 경우 해당 파라미터를 반드시 전달하지 않아도 디폴트 값으로 설정되기 때문에, 실제로 required = false 로 설정하는 것과 같습니다.

```
@RequestMapping(value = "/login5")
public String login5(@RequestParam String userid, @RequestParam(defaultValue = "0000") String passwd) {
	return "userid: "+userid+", passwd: "+passwd;
}
```

```
http://localhost:8092/testmapping/login5?userid=aaa
----
```

userid: aaa, passwd: 0000

여러 파라미터 매핑하기

List, 배열 - 다중 값 매핑하기

동일한 파라미터 name으로 여러 데이터가 전달되는 경우 List나 배열 등으로 매핑할 수 있습니다.

```
@RequestMapping(value = "/login6")
public String login6(@RequestParam List<String> ids) {
	return "ids: "+ids;
}
```

http://localhost:8092/testmapping/login6?ids=1,2,3
---
ids: [1, 2, 3]

```
@RequestMapping(value = "/login7")
public String login7(@RequestParam String[] ids) {
	String str = "";
	for (String id : ids) {
		str += id+" ";
	}
	return "ids: "+str;
}
```

http://localhost:8092/testmapping/login7?ids=1&ids=2&ids=3
---
ids: 1 2 3

Map - 모든 파라미터 매핑하기

파라미터 name이나 개수에 대한 정의 없이 Map으로 모든 파라미터를 매핑할 수 있습니다. key는 파라미터 name으로, 해당 파라미터 value는 value로 저장됩니다.

```
@RequestMapping(value = "/login8")
public String login8(@RequestParam Map<String, String> allParams) {
	return "allParams: "+allParams;
}
```

http://localhost:8092/testmapping/login8?userid=aaa&passwd=1234&age=10
---
allParams: {userid=aaa, passwd=1234, age=10}