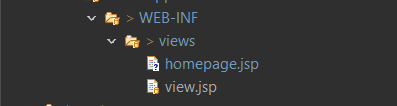
Topics Covered :

* Data binding using @RequestParam
* Data binding using DTO
* Data binding using @ModelAttribute

Before started learn about **DATA BINDING**, add the following thing in your **lovecalculator** project :

****

**homepage.jsp :**

<%@ **page** language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!**DOCTYPE** html>

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Home page</**title**>

</**head**>

<**body**>

<**div** align=*'center'*>

<**h1**>Love calculator</**h1**>

<**form**>

<**div**>

<**label** for=*'yourname'*>Your name : </**label**>

<**input** type=*'text'* name=*'yourname'* id=*'yourname'* />

</**div**>

<**div**>

<**label** for=*'crushname'*>Crush name : </**label**>

<**input** type=*'text'* name=*'crushname'* id=*'crushname'* />

</**div**>

<**div**>

<**input** type=*'submit'* value=*'calculate'* />

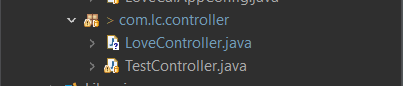
</**div**>

</**form**>

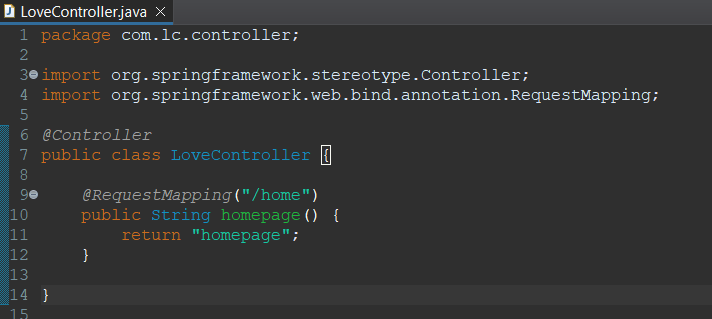
</**div**>

</**body**>

</**html**>



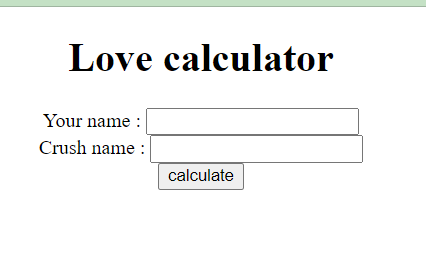
**LoveController.java**

****

Now let’s understand first what is **DATA BINDING**

For sure you have made those changes we have discussed above. Now run your project and hit a request on the given below URL :

[**http://localhost:2021/lovecalculator/myhome/home**](http://localhost:2021/lovecalculator/myhome/home)



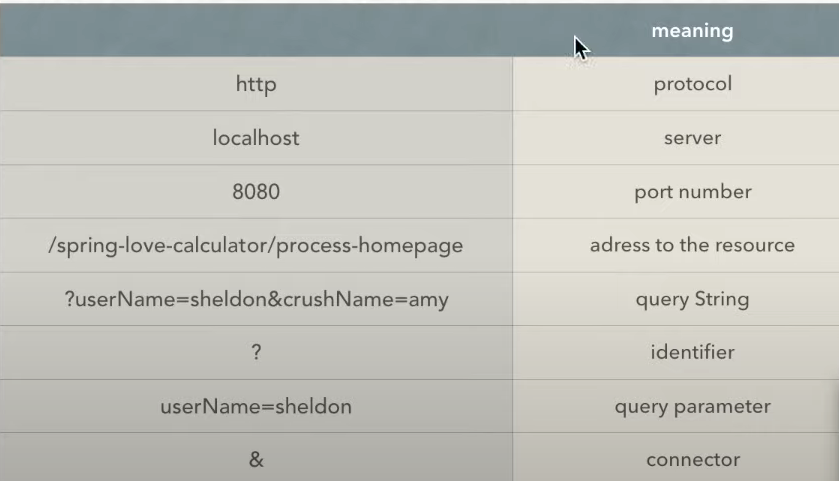
You will get a form this.

Now type your name and your crush name in the respective field and hit on calculate.

When you hit the calculate button, no action will be performed but there is change in URL.

[**http://localhost:2021/lovecalculator/myhome/home?yourname=sourabh&crushname=java**](http://localhost:2021/lovecalculator/myhome/home?yourname=sourabh&crushname=java)

where,

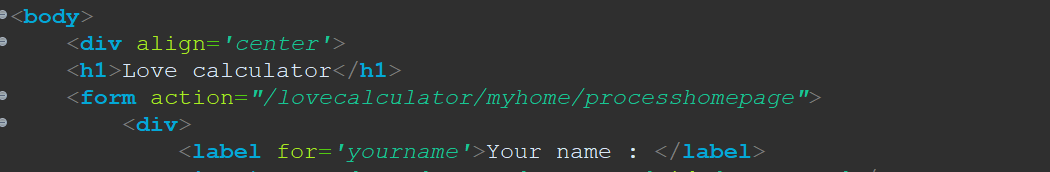


As an intelligent person, I am sure you have already guessed that to receive form data (Your name and Crush Name) at server side we need to deal with **Query String**.

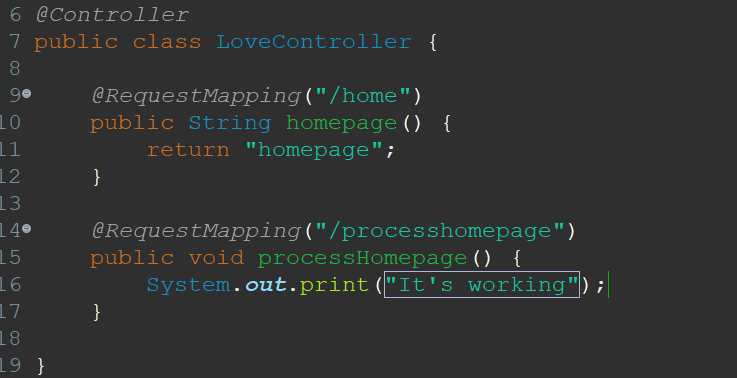
Now, the question is how we do that ?

Let’s do it together …

First we have to tell the action URL in our **homepage.jsp**

****

Now create an endpoint to handle this URL request inside your **LoveController.java**



So now when you hit the Calculate button, a message will print in your console.

Ok, now let’s move ahead !

Obviousely, that query string will also coming with this request but we don’t know yet that how we scrap the form data from the query string. Let’s learn it now

yourname=sourabh&crushname=java

In this specific query string, we have two query parameter

yourname=Sourabh

crushname=java

Now as you can observe, query parameter is a key value pair.

We have an annotation **@RequestParam( KEY )** which takes the key of query parameter and have ability to inject it’s value into a method parameter.

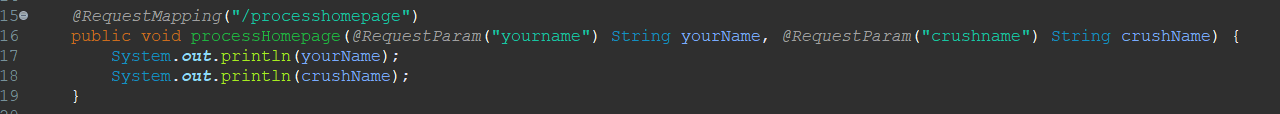
Let’s take an example,

Suppose we have a method,

public void func(@RequestParam(“yourname”) String yourName) { }

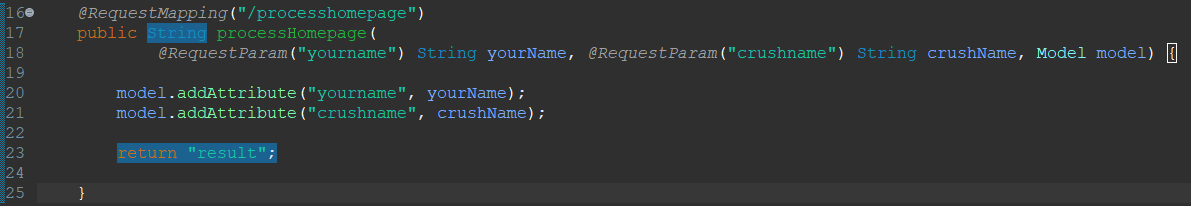
Now whatever value linked with yourname will be injected into yourName parameter

Make some changes in your processHomepage() method like below :



Till now, we have access of data passing through the form, now let’s render it using a new view.

As we know, to access data from controller to view we need of **Model** object and add into all the data whatever we want to access inside our view.



Create a new JSP page called **result.jsp,** render the model data accordingly.

<%@ **page** language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!**DOCTYPE** html>

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>Love Result</**title**>

</**head**>

<**body**>

<**h1**>User name : ${ yourname }</**h1**>

<**h1**>User name : ${ crushname }</**h1**>

</**body**>

</**html**>

Now run the project you should view the result page.

You have used @RequestParam and the program was working perfectly. But did you observe a shortcoming in this approach ?

Assume, a form sending 10 Query parameters. In this case, you have to use @RequestParam with 10 method parameters. So your method structure will become complex and less readable.

To solve this problem, spring provides the concept of **DTO :**

Create a class **UserDto.java** inside **com.lc.dto** package

package com.lc.dto;

public class UserDto {

private String yourName;

private String crushName;

public String getYourName() {

return yourName;

}

public void setYourName(String yourName) {

this. yourName = yourName;

}

public String getCrushName() {

return crushName;

}

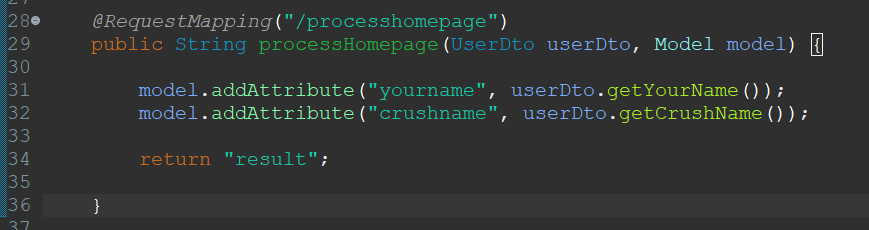
public void setCrushName(String crushName) {

this.crushName = crushName;

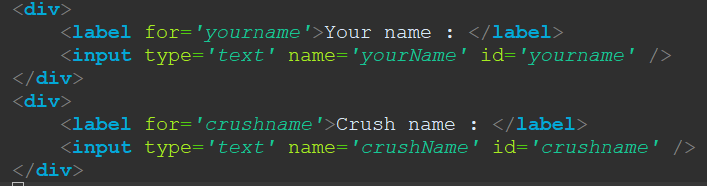
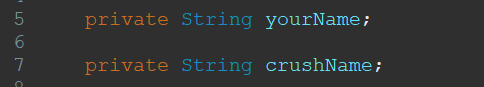
}

}

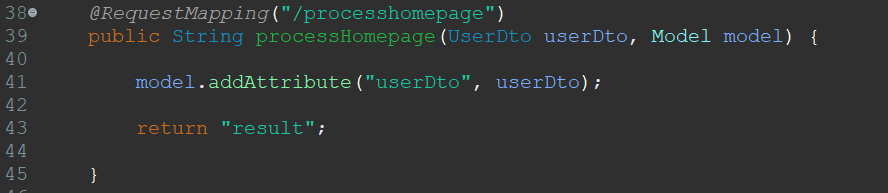
Now instead of using @RequestParam, Just follow the bellow image

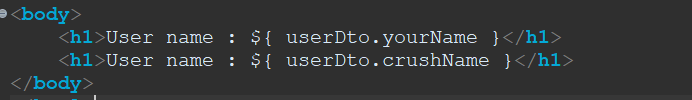


The only thing you need to care about, the name of field you are setting inside your **homepage.jsp** should be match the field defined in the **UserDto.java**

**** ****

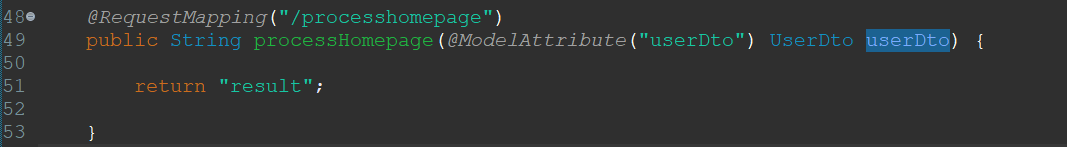
Let’s compact our code little bit more,





Can we compact it more ?

Yes, by using **@ModelAttribute**



What’s happening here ?



It’s is doing two things,

1. Creating object of **Model**
2. Adding an attribute having key userDto and value **UserDto object**