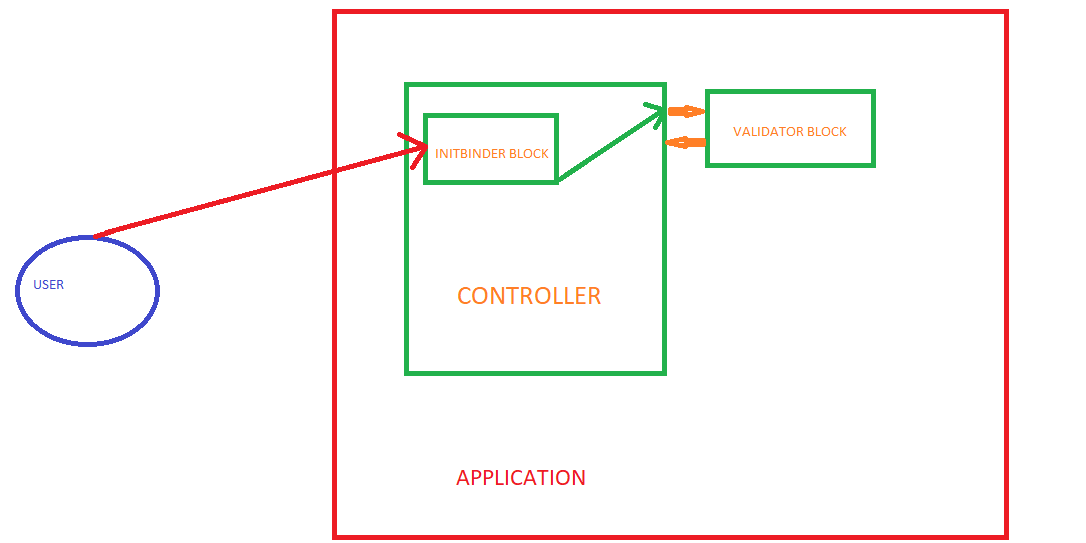
**How to Use @Initbinder in Spring MVC.**

**How to Use @Initbinder in Spring MVC**:

In Spring MVC architecture, controller is used for processing the web request and rendering the response to the View.

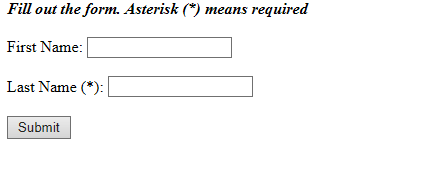
Now, @Initbinder comes into picture if we want to customize the request being sent to the controller. It is defined in the controller, helps in controlling and formatting each and every request that comes to it. This annotation is used with the methods which initializes WebDataBinder and works as a preprocessor for each request coming to the controller.



**InitBinder interaction with other Components**

Consider we have a Customer management portal for a shop, which manages the information of all the customers in the shop. It has a feature for registering a customer when he is admitted to the shop.

So the form looks somewhat like:

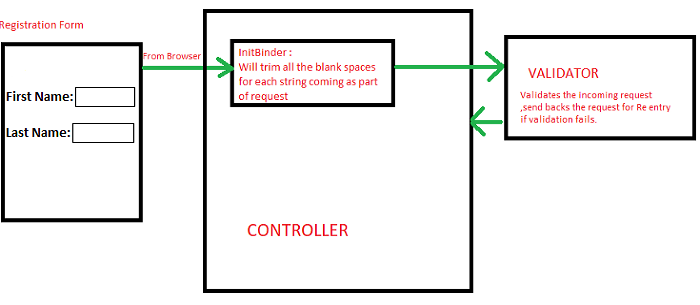


Consider the Last Name of the form, lastName **not** expected as **Null**.

**The Problem**:

If the user inserts some blank spaces for the “Last Name” and submits the form, the validator block will not treat it as Null. Hence the system will take the value as spaces and the subsequent processing will be done. But we don’t want a value with spaces to be present in our Database.

So, to have a check on this case we can consider to use a preprocessor(Initbinder) for the web request, which will trim the values coming as part of request, which means for our case it will also trim the values to Null if only blank spaces are present.



**Implementation of Initbinder**

**How to implement this**:

Consider a model Customer which has firstName ,lastName. This is the model which will carry the info of a Customer between different spring components.

**File: Customer.java**:

**package** com.ruhul.odduu.springmvc;

**import** javax.validation.constraints.NotNull;

**import** javax.validation.constraints.Size;

**public** **class** Customer {

**private** String firstName;

@NotNull(message = "is required")

@Size(min = 1, message = "is required")

**private** String lastName;

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

}

**Controller class**:

We will add @Initbinder annotated method to the controller, To add a initbinder method we have to declare a method with @initbinder annotation,this method should have WebDatabinder as parameter.

**File: CustomerController.java**:

**package** com.ruhul.odduu.springmvc;

**import** javax.validation.Valid;

**import** org.springframework.beans.propertyeditors.StringTrimmerEditor;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.web.bind.WebDataBinder;

**import** org.springframework.web.bind.annotation.InitBinder;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

@Controller

@RequestMapping("/customer")

**public** **class** CustomerController {

// add an initbinder.. to convert trim input string

// remove leading and trailing whitespace

// resolve issue for our validation

@InitBinder

**public** **void** initBinder(WebDataBinder dataBinder) {

// it is part of Spring API. true means trim to null

StringTrimmerEditor stringTrimmerEditor = **new** StringTrimmerEditor(**true**);

dataBinder.registerCustomEditor(String.**class**, stringTrimmerEditor);

}

@RequestMapping("/showForm")

**public** String showForm(Model theModel) {

theModel.addAttribute("customer", **new** Customer());

**return** "customer-form";

}

@RequestMapping("/processForm")

**public** String processForm(@Valid @ModelAttribute("customer") Customer theCustomer, BindingResult theBindingResult) {

System.***out***.println("Last name: |" + theCustomer.getLastName() + " |");

**if** (theBindingResult.hasErrors()) {

**return** "customer-form";

} **else** {

**return** "customer-confirmation";

}

}

}

We will register StringTrimmerEditor as a custom editor to the databinder with String class as target source. StringTrimmerEditor is a PropertyEditor which we are using in our demo that trims the string values. After registering the custom editor ,initbinder will trim all the String values coming as part of request.

Hence, now if the Name is sent as blank white spaces, our application will not allow the values to pass through the validator system, avoiding a leak in the system.

**Conclusion**:

This was a use case where initbinder was used as a preprocessor to trim the string values in the request before reaching the controller. Similarly many more operations like converting the String object to a Date object, adding some extra values to the input can be done.

<https://medium.com/bridgecrew/how-to-use-initbinder-in-spring-mvc-ecb974a6884>

How to Use @Initbinder in Spring MVC.