**14.1. Form Tag Text Fields – Overview**

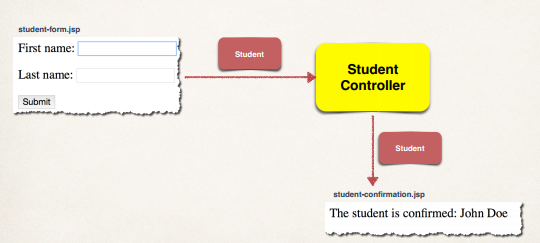
**14.1. Form Tag Text Fields (Spring MVC) – Overview**:

The Spring MVC form text field tag generates an HTML input tag using the bound value. By default, the type of the input tag is text.

**Syntax**:

**<form:input** path="name" **/>**

Here, path attribute binds the form field to the bean property.



**Showing Form**:

In our spring controller, before we show the form, we must add a model attribute. This is basically a bean that is used to hold the form data. That will give support for data binding.

* Before you show the form, you must add a model attribute
* This is a bean that will hold form data for the data binding

**Add Model Attribute**:

@RequestMapping("/showForm")public String showForm(Model theModel) { theModel.addAttribute("student", new Student()); return "student-form";}

1. Model is used to pass data between controllers and view
2. Here in "theModel.addAttribute("student", new Student());", "student" is attribute name and "new Student()" is attribute value. Effectively we are creating an empty Student object that I will pass to the form for the form to make use of with data binding.

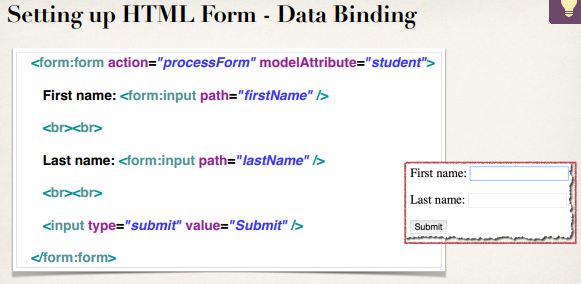
Here the important thing is the attribute name "student" is the same name that our form will use to reference this model attribute.

**Setting up HTML Form - Data Binding**:

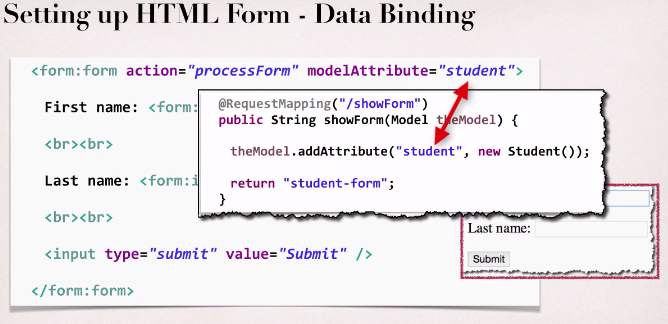
<form:form action=*"processForm"* modelAttribute=*"student"*>  
 First name: <form:input path=*"firstName"* />  
 <br><br>

Last name: <form:input path=*"lastName"* />  
 <br><br>

<input type=*"submit"* value=*"Submit"* />  
</form:form>



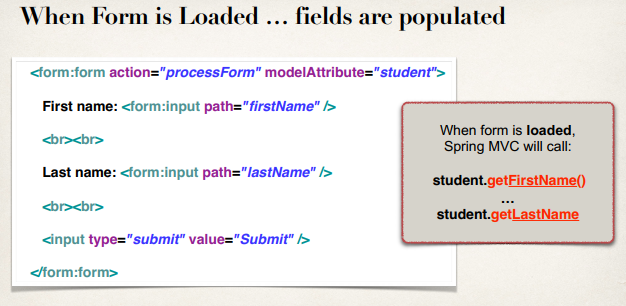
The important thing is that the modelAttribute name in form must be same with controller attribute name in controller class.



Here the **path=*"firstName"*** and **path=*"lastName"*** actually binds this form field to a property on the bean.

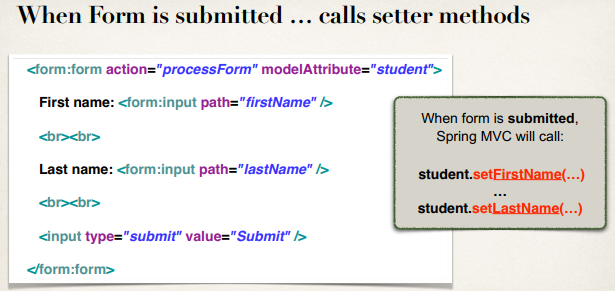
**When Form is Loaded … fields are populated**:

When the form is first loaded, Spring MVC will do behind the scenes is that they will actually use to populate the form field. When they have **path=*"firstName"*** Spring MVC will call student.getFirstName() and they use that modal attribute from at top to retrieve that data. If it’s null then that form field will simply be empty.



**When Form is submitted Spring will calls setter methods**:

When Form is submitted Spring actually call the setter methods.



**Handling Form Submission in the Controller**:

Now after submitted form data we have to read data in our controller. For this we can simply make use of a new Spring annotation called @ModelAttribute

@RequestMapping("/processForm")  
public String processForm(@ModelAttribute("student") Student

theStudent){  
 // log the input data  
 System.*out*.println("theStudent: " + theStudent.getFirstName()  
 + " " + theStudent.getLastName());  
 return "student-confirmation";  
}

Here in "processForm(@ModelAttribute("student") Student theStudent)", the attribute is "student". It actually binds that object to this parameter being passed in theStudent.

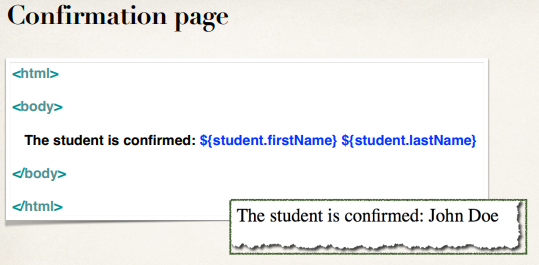
Some work behind the scenes, Spring will actually take that model attribute and bind it to this variable here, theStudent, and we can use it in our controller.

The key here is that Spring will actually take all that form data, bind it automatically to our object and then pass it into our controller so we can use it.

Finally we return the confirmation page "student-confirmation"

**Confirmation page**:

**<html>  
<body>  
 The student is confirmed: ${student.firstName} ${student.lastName}  
</body>  
</html>**



**Development Process (Step by Step)**:

1. Create Student class
2. Create Student controller class
3. Create HTML form
4. Create form processing code
5. Create confirmation page

14.1. Form Tag Text Fields - Overview