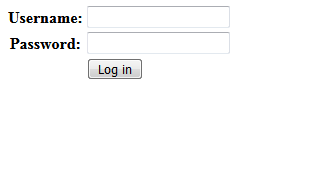
**Spring MVC Demo 4 User Guide**

1. Import ddl.sql to the MySQL named demodb4.

2. Run SpringMVCDemo4 project on Tomcat.

3. Run file Login.html in SpringMVCDemo4FrontEnd folder.

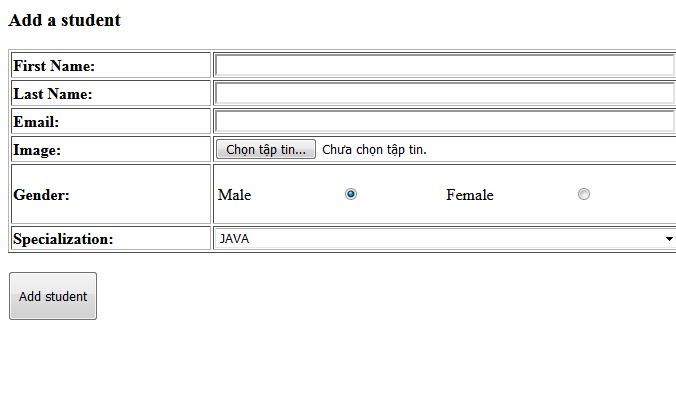


4. Login with username: admin and password: 123abc

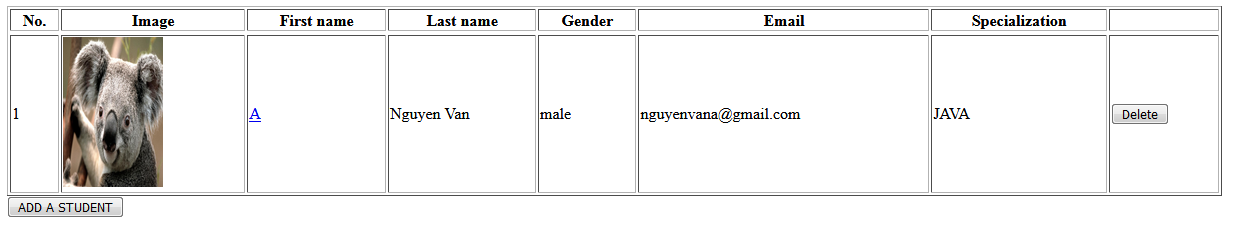
5. We will see the Student List Page like this.



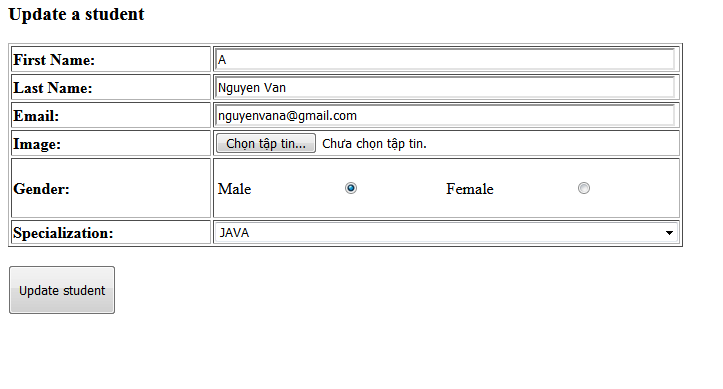
6. Click 'ADD A STUDENT' to go to the Student Adding Page.



7. Fill all necessary field in the form and click 'Add student'. (remember to choose an image file for Image). It will relocate to the Student List Page with the new student, which has been added before.



8. We can edit information of this student click on the First Name of this student. Then it will relocate to the Update Student Page with their current information except Image.

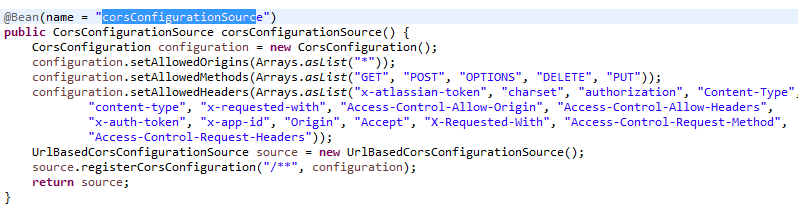


Because Image is not a primitive data type (It is a multipart-file). Therefore, if you do not want to update Student Image, we can skip that field for keeping this image. After finish editing, click 'Update student' to complete.

9. If we want to delete student, we can click button 'Delete' at each row.

**CORS and Uploading Files**

1. Because Spring MVC is a strict framework, it always has many types of firewall that protect developer's source code from hackers and so on. Therefore, we must set up some exceptions for those firewalls (that will allow requests from Front-end) by setting up the bean named corsConfigurationSource like this :



For example: this bean will command our application do something like:

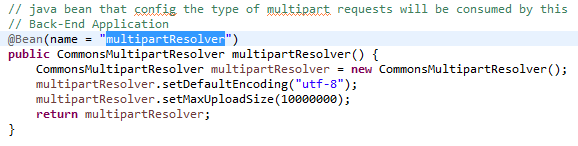
* Allow request methods like GET, POST, PUT, DELETE.
* Allow request headers like Authorization, Content-type, and so on.
* Allow some formats of request URL (in this example, I allowed all format of request).

By doing this technique, Spring MVC Application will allow to communicate with other native framework like Angular, React, or Jquery Ajax.

1. About upload files to Spring MVC Application, we also have some solutions:

* Serialize those files to blob values and send it to Spring MVC. This method is very old and low performance.
* Upload those files by using MultipartFile Object and send them via HttpRequests. This is the best solution for projects because it's performance.

To do this technique, we must declare the bean that make our application consumes MultipartFile named multipartResolver, this bean will declare the type of MultipartFile that will be consumes. For example, encoding type, maximum weight of files, etc.



In SpringMVCDemo4 Project, I made a util class called fileUploadUtils, which is stored in vn.edu.saigontech.SpringMVCDemo.utils, that has some methods can help us to upload file to server:

* saveUploadedFiles(List<MultipartFile> files, String saveLocationPath)

this method will upload a list of MultipartFile to the direction named saveLocationPath in server.

* saveUploadedFile(MultipartFile file, String saveLocationPath)

this method will upload a MultipartFile to the direction named saveLocationPath in server.

* deleteUploadFile(String fileName, String saveLocationPath)

this method will delete a file named fileName in the direction named saveLocationPath in server.

* formatFileNameToServer(String target)

this method will change the value of target String, that will be suitable when they are uploaded to server. This method will be use inside saveUploadFile and saveUploadFiles method above.

In controllers, we will use those utility methods like this:

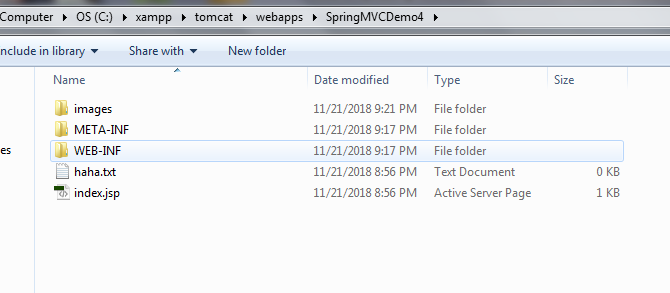


In this example, we will see that method saveUploadedFiles save all file in uploadfiles array into the path named req.getServletContext().getRealPath("/images/")

\* For more information about req.getServletContext().getRealPath("/images/"), this is a static folder stored in bin folder of each eclipse project (in our project, it will be stored in

C:\Eclipse-WorkSpace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\SpringMVCDemo4\images\

). But when we deploy our project on Tomcat, it will be located under project folder like this:



About download files from Spring MVC Application, we can use some buffered objects that will receive file from server hard drive and parse them to our familiar files like this example:



End.