X-editable plugin and Spring MVC - best way to update Entity

Stack searched but did not find the answer. I have subpages with several textfield's and when I editing the data field a plugin sends a reply to my controller:

```
@RequestMapping(value = "/myAcc", method = RequestMethod.POST)
    public String getValues(@ModelAttribute XEditableForm form){
        userService.update(form.getPk(), form.getValue());
        return "myAcc";
    }
```

where userService make a user update in db

```
@Transactional
public void update(long id, String firstName){
    User user= userRepository.findOne(id);
    user.setFirstName(firstName);
    userRepository.save(user);
}
```

The problem is that every time in this method I would check what was returned from xeditable plugin and update specific user field i.e. surname etc. In my opinion this is not the best solution.

XEditableForm returns:

```
pk - primary key of record to be updated (ID in db)
name - name of field to be updated (column in db)
value - new value
```

Question for you. How can I do this better?

java database spring x-editable

edited Feb 18 '16 at 18:52 asked Feb 18 '16 at 17:44
Adamo
159 1 17

It depends. If it just a submit button that's fine, there is no overhead. If you want reflect form changes to database immediately for every key press I suggest send request only after few seconds past since last key press or find out some event which tells that you are done and time to send data. If immediate changes not required to be persisted in database you can update you model attribute/detached entity/smth else and persist or merge it later. – Georgy Gobozov Feb 18 '16 at 18:58

I think my explanation is incompleted. The idea is that the user can change the values in several fields. After each change the user object is updated in database. So now I would have to write some methods to detect change in every field. If user change only surname I must check other fields and update only not null fields. I asked if there is any way to generalize this example by creating an object and then update it. You mention here about the merge. You could show an example of how it looks in JPARepository Interface ? – Adamo Feb 18 '16 at 19:28

1 Answer

I will answer for himself. In this situation we could use

```
org.springframework.utilClass ReflectionUtils
```

Exmaple of use:

```
User user = userRepository.findOne(id);
Field name = ReflectionUtils.findField(User.class, "name");
ReflectionUtils.makeAccessible(true);
name.set(user, "Admin");
```

In this way, you can edit the fields in the class knowing their names . And that's what I mean.

edited Mar 7 '16 at 7:11

answered Feb 22 '16 at 10:33

Adamo
159 1 17

@georgy gobozov could you look at it? – Adamo Feb 22 '16 at 10:36