Create to-do app called Indivilister with SPRING BOOT

Presented by ARMS（THAILAND）Co., Ltd.

Index

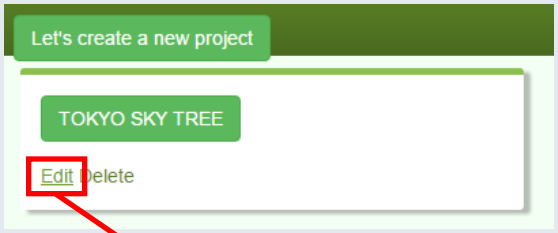
Note

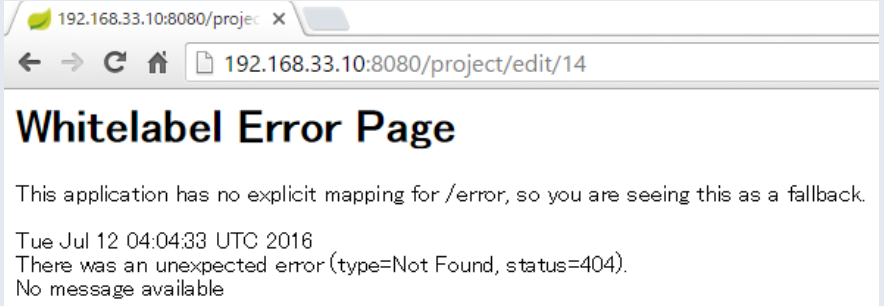
- Indivilister means "Individual + lister“

3. Create Indivilister's projects

3-2. Create edit a project

- In this section, create "edit a project"





When you click "Edit" button on your project, you see the error above.

```
<a th:href="'/project/edit/' + ${project.id}">Edit</a>
```

When you edit a project, you need its unique id.

3. Create Indivilister's projects

3-2. Create edit a project

- Add the following code into ProjectController.java

edit button <a th:href="'/project/edit/' + ${project.id}">Edit

```
@RequestMapping(value = "edit/{id}", method = RequestMethod.GET)
  public String edit(@PathVariable("id") int id, Model model) {
      model.addAttribute("projectForm", projectService.findProjectById(id));
      return "project/edit";
  }
```

@PathVariable can receive a parameter from URL and use it in the method.

We haven't created this projectService.findProjectById(id) method

3. Create Indivilister's projects

3-2. Create edit a project

- Add the following code into ProjectService.java

```
public ProjectForm findProjectById(int id) {
    Project project = projectRepository.findOne(id);
    return new ProjectForm(project.getId(), project.getName());
 }
```

No findOne() method is created in ProjectRepository, but you can already use it.

Find a project by its id, and store it "project", when this method returns ProjectForm, it calls ProjectForm(arg1, arg2) constructor.

ProjectForm.java

```
public ProjectForm(int id, String name) {
    this.id = id;
    this.name = name;
 }
```

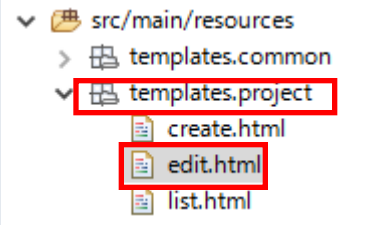
3. Create Indivilister's projects

3-2. Create edit a project

- From this code, you need "edit.html"

```
@RequestMapping(value = "edit/{id}", method = RequestMethod.GET)
  public String edit(@PathVariable("id") int id, Model model) {
      model.addAttribute("projectForm", projectService.findProjectById(id));
      return "project/edit";
  }
```

Create edit.html under templates.project



3. Create Indivilister's projects

3-2. Create edit a project

- Add the following code into edit.html

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>INDIVILISTER : Edit an Project</title>

   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css" charset="utf-8">
   <link rel="stylesheet" type="text/css" href="/css/custom.css" charset="utf-8">
</head>
<body>
<main>
   <!-- START Navigation bar -->
   <nav th:include="common/nav :: nav"></nav>
   <!-- END Navigation bar -->

   <!-- START Header -->
   <div class="header-custom">
      <div class="container">
         <h2>Edit Project</h2>
         <p>Edit your Project name</p>
      </div>
   </div>
   <!-- END Header -->
```

Continue on next page

3. Create Indivilister's projects

3-2. Create edit a project

- Add the following code into edit.html

```
<!-- START Content -->
<div class="container">
  <div class="row">
    <div class="col-lg-12">
      <div class="box effect1">
        <h4>Edit Project</h4>
        <form th:action="@{/project/edit}" method="POST" th:object="${projectForm}" accept-charset="utf-8"
enctype="application/x-www-form-urlencoded" class="form-horizontal">
          <input type="hidden" name="authenticityToken"
value="e739540dfb2b4389d499e26e8b6dc17f665de703">
          <input type="text" th:field="*{id}" hidden="">
          <div class="form-group">
            <label for="name" class="col-lg-2 control-label">Project name:</label>
            <div class="col-lg-10">
              <input class="form-control" th:field="*{name}" id="name" type="text" name="name" required="">
            </div>
          </div>
          <div class="form-group">
            <div class="col-lg-offset-2 col-lg-10">
              <button type="submit" class="btn btn-success">Edit</button>
              <button type="reset" class="btn btn-danger">Reset</button>
              <a th:href="@{/project}" class="btn btn-primary">Back to Home</a>
            </div>
          </div>
        </form>
```

Continue on next page

3. Create Indivilister's projects

3-2. Create edit a project

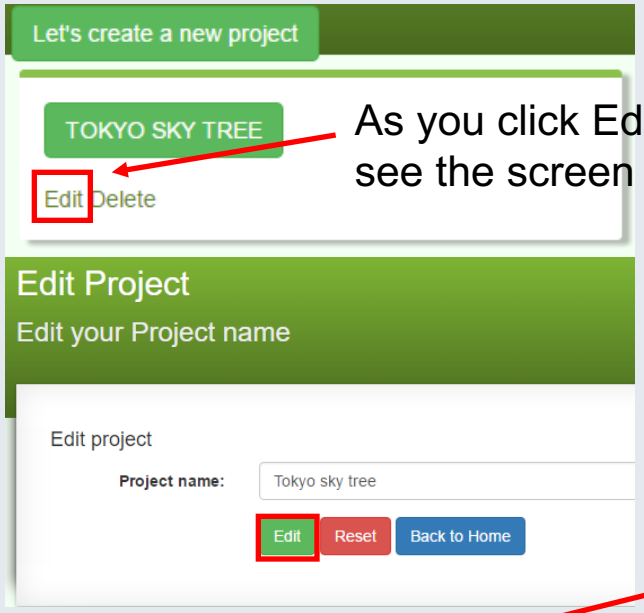
- Add the following code into edit.html

```
            </div>
          </div>
        </div>
    </div>
    <!-- END Content -->

    <div class="push"></div>
</main>

<!-- START Footer -->
<span th:include="common/footer :: footer"></span>
<!-- START Footer -->

<script type="text/javascript" language="javascript" charset="_charset" src="/js/jquery-1.12.2.min.js"></script>
<script type="text/javascript" language="javascript" charset="_charset" src="/js/bootstrap.min.js"></script>
</body>
</html>
```

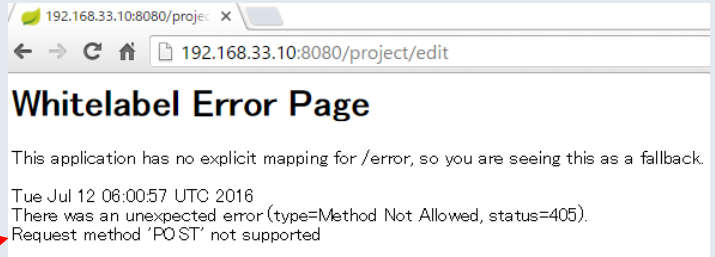
3. Create Indivilister's projects

3-2. Create edit a project

- Restart your project and access server:8080/project



As you click Edit button, you will see the screen like below.

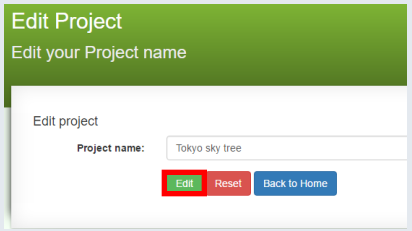


But if you click other "Edit" button, you will see the error.

3. Create Indivilister's projects

3-2. Create edit a project

- Add the following code into ProjectController.java




```
<form th:action="@{/project/edit}" method="POST" th:object="${projectForm}"
```

```
@RequestMapping(value = "edit", method = RequestMethod.POST)
  public Object edit(@ModelAttribute ProjectForm projectForm) {
      projectService.update(projectForm);
      return "redirect:/project";
  }
```

But we haven't created projectService.update(projectForm) method

3. Create Indivilister's projects

3-2. Create edit a project

- Add the following code into ProjectService.java

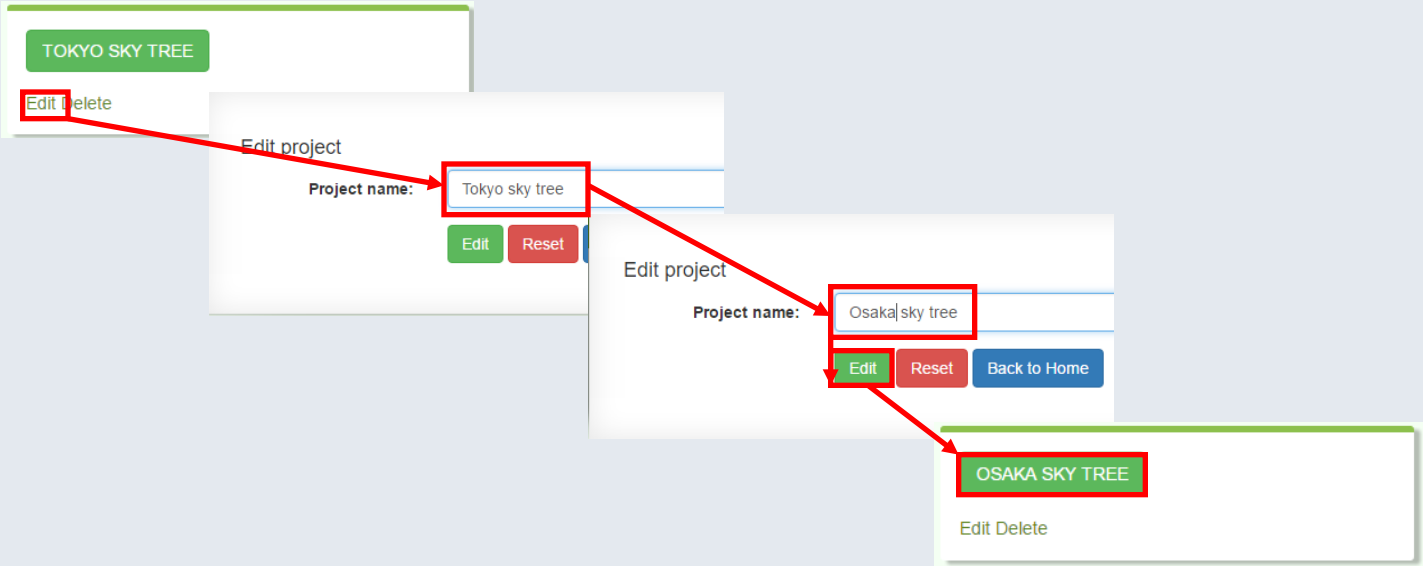
```
public void update(ProjectForm projectForm) {
    Project project = projectRepository.findOne(projectForm.getId());
    project.setName(projectForm.getName());
    project.setUpdatedDate(Calendar.getInstance().getTime());
    projectRepository.save(project);
}
```

Repository.save() method is a dual purposed method. Spring decides if this is inserted or updated. Spring checks @id(id property) of the entity to find out if the entity is new one or not. If the id property is null, the entity will be treated as new(insert), else not new(update)

3. Create Indivilister's projects

3-2. Create edit a project

- Restart your project and access server:8080/project

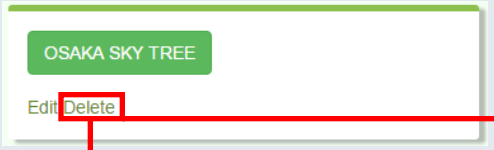


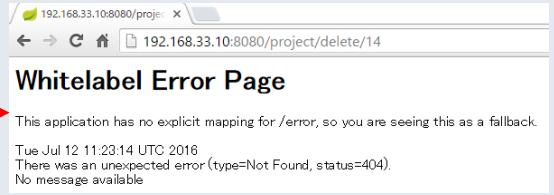
Edit and update your project to see if it's properly updated.

3. Create Indivilister's projects

3-3. Create delete a project

- In this section, Create delete a project





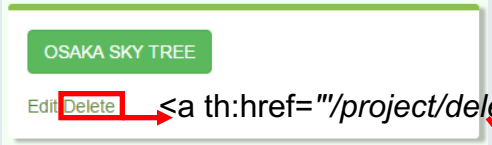
As you delete your project, you will see the error like the above.

<a th:href="'/project/delete/' + ${project.id}"

3. Create Indivilister's projects

3-3. Create delete a project

- Add the following code into ProjectController.java



<a th:href="'/project/delete/' + ${project.id}"

```
@RequestMapping(value = "delete/{id}",  method = RequestMethod.GET)
  public String delete(@PathVariable("id") int projectId) {
      projectService.delete(projectId);
      return "redirect:/project";
  }
```

@PathVariable receives {id} from URL, and the id goes to an argument for the delete() method.

But we haven't created projectService.delete(projectId)

3. Create Indivilister's projects

3-3. Create delete a project

- Add the following code into ProjectService.java

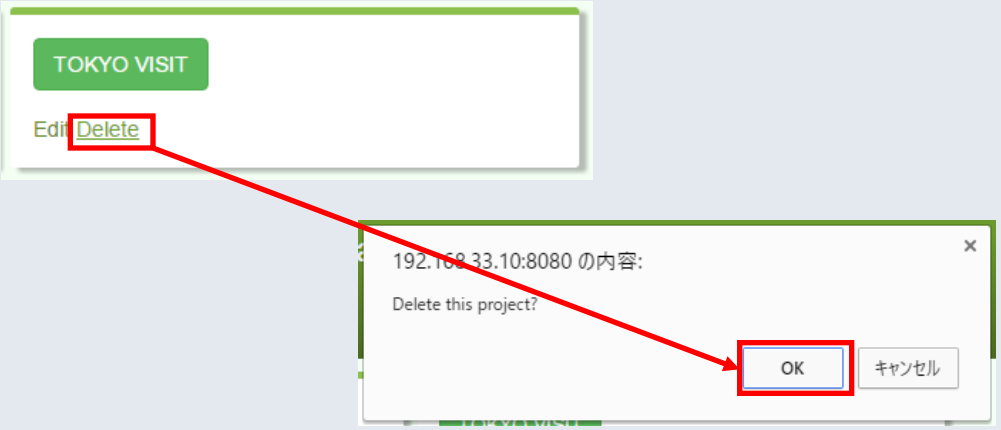
```
public void delete(int projectId) {
        projectRepository.delete(projectId);
  }
```

No delete method is defined in ProjectRepository…, but you can also use this method.

3. Create Indivilister's projects

3-3. Create delete a project

- Restart your project and access server:8080/project

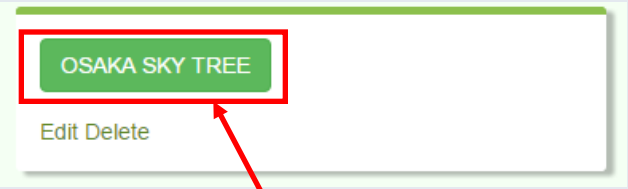


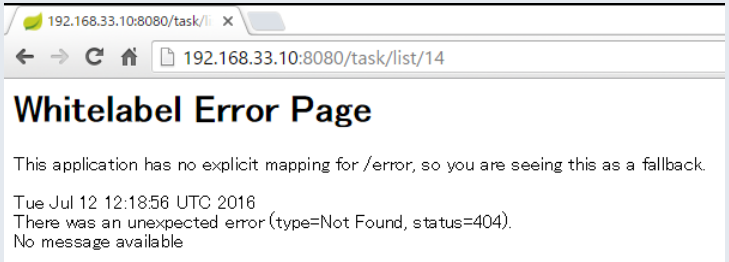
Check if delete function is properly working.

4. Create Indivilister's tasks

4-1. Create a task

- In this chapter, we create tasks in a project.





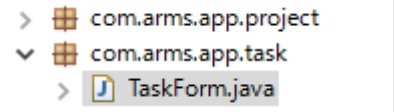
As you click "project name" button to create tasks, you will see the error like the above.

```
<a th:href="'/task/list/' + ${project.id}" class="btn btn-success">
<span th:text="${project.name}">project0001</span>
</a>
```

4. Create Indivilister's tasks

4-1. Create a task

- Create a form object to receive data from text field. TaskForm.java under com.arms.app.task



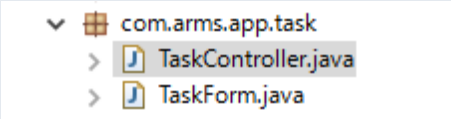
Add the following code into TaskForm.java

```
package com.arms.app.task;

import lombok.Data;

@Data
public class TaskForm {

    private int id;

    private String name;

    private int projectId;
}
```

4. Create Indivilister's tasks

4-1. Create a task

- Create TaskController.java under com.arms.app.task



Add the following code into TaskController.java

```
package com.arms.app.task;

import com.arms.domain.service.TaskService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("task")
public class TaskController {
```

Continue on next page

4. Create Indivilister's tasks

4-1. Create a task

- Add the following code into TaskController.java

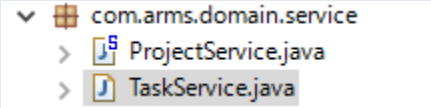
```
@Autowired
    TaskService taskService;

    @RequestMapping(value = "list/{project_id}", method = RequestMethod.GET)
    public String list(@PathVariable("project_id") int projectId, Model model) {
        TaskForm taskForm = new TaskForm();
        taskForm.setProjectId(projectId);
        model.addAttribute("taskForm", taskForm);
        model.addAttribute("project", taskService.findProjectByProjectId(projectId));
        return "task/list";
    }
}
```

We haven't created TaskService yet…..

4. Create Indivilister's tasks

4-1. Create a task

- Create TaskService.java under com.arms.domain.service



Add the following code into TaskService.java so that you can use it from the controller.

```
package com.arms.domain.service;

import com.arms.app.task.TaskForm;
import com.arms.domain.entity.Project;
import com.arms.domain.entity.Task;
import com.arms.domain.repository.ProjectRepository;
import com.arms.domain.repository.TaskRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Calendar;
import java.util.Date;

@Service
public class TaskService {
```

4. Create Indivilister's tasks

4-1. Create a task

- Add the following code into TaskService.java

```
    @Autowired
    ProjectRepository projectRepository;

    @Autowired
    TaskRepository taskRepository;

    public Project findProjectByProjectId(int projectId) {
        return projectRepository.findOne(projectId);
    }
}
```

Tasks should be found by project id since many tasks belong to one project. You have to be able to access Project field as well as Task field. So both Task and ProjectRepository should be annotated with @Autowired.

4. Create Indivilister's tasks

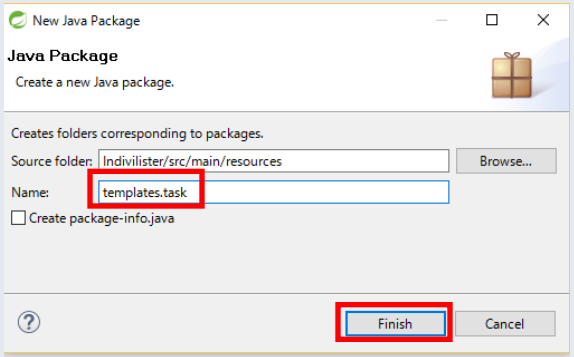
4-1. Create a task

- Create task/list.html under templates.task

```
@RequestMapping(value = "list/{project_id}", method = RequestMethod.GET)
   public String list(@PathVariable("project_id") int projectId, Model model) {
...............................
        return "task/list";
   }
```

As you see list() method, this returns the view task/list.html

Right-click on "src/main/resources" New – Package
Type templates.task in the Name field.
and press Finish button.



4. Create Indivilister's tasks

4-1. Create a task

- Right-click on templates.task New – File

and create list.html. Add the following code into list.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>INDIVILISTER : project0001</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css" charset="utf-8">
   <link rel="stylesheet" type="text/css" href="/css/custom.css" charset="utf-8">
</head>
<body>
<main>
   <!-- START Navigation bar -->
   <nav th:include="common/nav :: nav"></nav>
   <!-- END Navigation bar -->

   <!-- START Header -->
   <div class="header-custom">
      <div class="container">
         <h2>project0001</h2>
         <p></p>
      </div>
   </div>
   <!-- END Header -->
```

Continue on next page

4. Create Indivilister's tasks

4-1. Create a task

- Add the following code into list.html

```
<!-- START Content -->
  <div class="container">
    <div class="row">
      <div class="col-xs-12">
        <div class="box effect1">
          <h4 th:text="'All tasks of ' + ${project.name}">All tasks of xxxx</h4>
          <div class="table-responsive">
            <table class="table">
              <tbody>
              <tr th:each="task, stat : ${project.taskList}">
                <td class="col-lg-2" style="text-align: right">
                  <input type="checkbox" class="task-status" th:data="${task.id}" th:checked="${task.status}? 'true' : 'false'">
                </td>
                <td class="col-lg-8" th:text="${task.name}">task001</td>
                <td class="col-lg-2">
                  <a th:href="'/task/delete/' + ${project.id} + '/' + ${task.id}" onclick="return confirm('Delete this task?');">Delete</a>
                </td>
              </tr>
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
```

Continue on next page

4. Create Indivilister's tasks

4-1. Create a task

- Add the following code into list.html

```
<div class="row">
        <div class="col-xs-12">
          <div class="box effect1">
            <h4 th:text="'Add a new task for ' + ${project.name}">Add a new task for project0001</h4>
            <form th:action="@{/task/create}" th:object="${taskForm}" method="POST" accept-charset="utf-8" enctype="application/x-
www-form-urlencoded" class="form-horizontal">
              <input type="hidden" name="authenticityToken" value="e739540dfb2b4389d499e26e8b6dc17f665de703">
              <input type="text" id="project-id" th:field="*{projectId}" type="hidden">
              <div class="form-group">
                <label for="newTask" class="col-lg-2 control-label">New task:</label>
                <div class="col-lg-10">
                  <input class="form-control" th:field="*{name}" id="newTask" type="text" name="newTask">
                </div>
              </div>
              <div class="form-group">
                <div class="col-lg-offset-2 col-lg-10">
                  <button type="submit" class="btn btn-success">Create a new task</button>
                  <button type="reset" class="btn btn-danger">Reset</button>
                  <a href="/project/" class="btn btn-primary">Back to Home</a>
                </div>
              </div>
            </form>
```

Continue on next page

4. Create Indivilister's tasks

4-1. Create a task

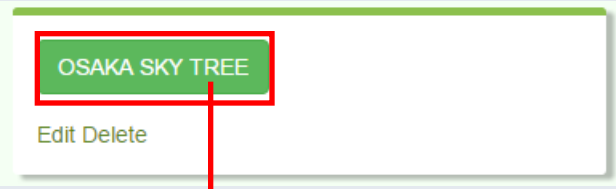
- Add the following code into list.html

```
            </div>
          </div>
        </div>
    </div>
    <!-- END Content -->
    <div class="push"></div>
</main>

<!-- START Footer -->
<span th:include="common/footer :: footer"></span>
<!-- START Footer -->

<script type="text/javascript" language="javascript" charset="_charset" src="/js/jquery-1.12.2.min.js"></script>
<script type="text/javascript" language="javascript" charset="_charset" src="/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function(){
        $('.task-status').on('click',function(){
            var status = $(this).prop('checked');
            window.location.href = '/task/edit/' + $(this).attr('data') + '/' + status
        });
    });
</script>
</body>
</html>
```

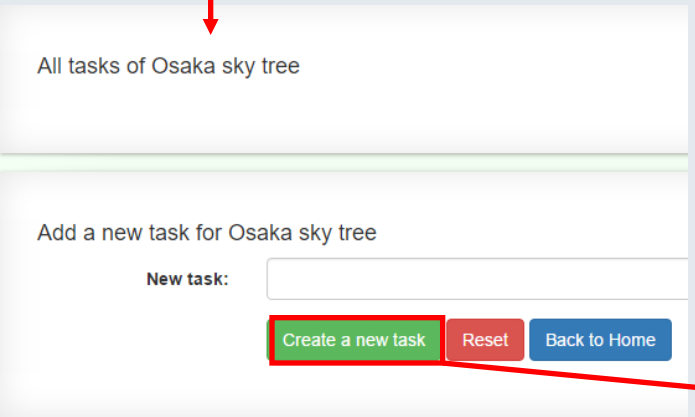
4. Create Indivilister's tasks

4-1. Create a task

- Restart your project and access server:8080/project



Click "project name" button to move to the task creation screen

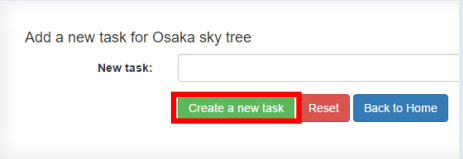


But…if you create a new task, you will see the error.

4. Create Indivilister's tasks

4-1. Create a task

- Create a method corresponding to this action



<form th:action="*@{/task/create}*" th:object="*${taskForm}*" method="*POST*"

Add the following code into TaskController.java

```
@RequestMapping(value = "create", method = RequestMethod.POST)
    public String create(@ModelAttribute TaskForm taskForm) {
        taskService.save(taskForm);
        return "redirect:/task/list/" + taskForm.getProjectId();
    }
```

We haven't created save() method in TaskService.java

4. Create Indivilister's tasks

4-1. Create a task

- Add the following code into TaskService.java

```
public void save(TaskForm taskForm) {
    Date date = Calendar.getInstance().getTime();
    Task task = new Task();
    task.setName(taskForm.getName());
    task.setProject(projectRepository.findOne(taskForm.getProjectId()));
    task.setStatus(false);
    task.setCreatedDate(date);
    task.setUpdatedDate(date);
    taskRepository.save(task);
  }
```

This process is almost the same as save(project).

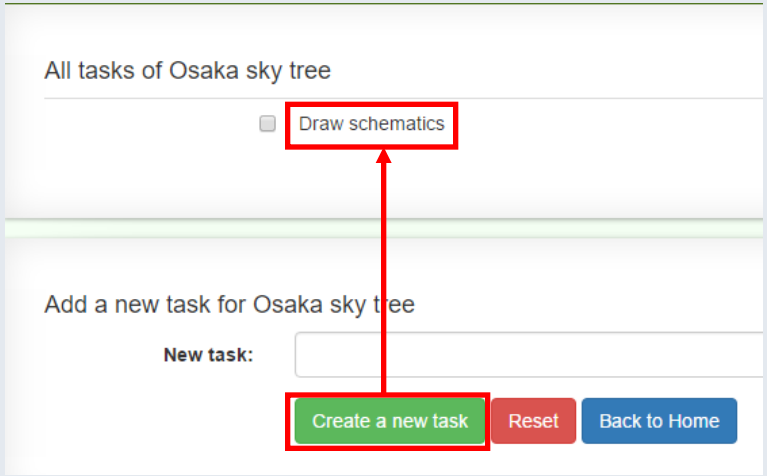
private boolean status (Task.java)
task.setStatus(**false**); (task.status = **false**;)

status field is boolean (true/false) . This field is linked to the checkbox of each task.

4. Create Indivilister's tasks

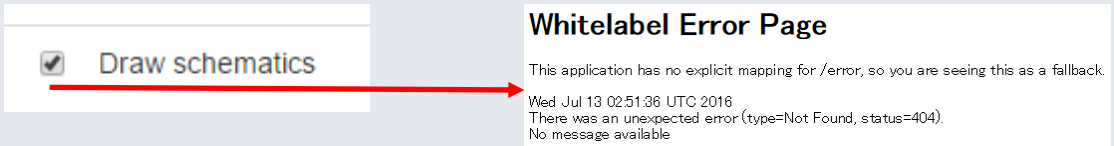
4-1. Create a task

- Restart your project and access server:8080/project



Now you can create a new task.

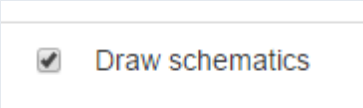
But… as you make the checkbox checked, you will see the error.



4. Create Indivilister's tasks

4-2. Create edit a task

- Let us see what' happening in the checkbox.




```
<tr th:each="task, stat : ${project.taskList}">
    <td class="col-lg-2" style="text-align: right">
        <input type="checkbox" class="task-status" th:data="${task.id}"
th:checked="${task.status}? 'true' : 'false'">
    </td>
    <td class="col-lg-8" th:text="${task.name}">task001</td>
```

${project.taskList}

```
@OneToMany(mappedBy = "project", cascade = CascadeType.ALL)
   private List<Task> taskList;

@ManyToOne
   @JoinColumn(name = "project_id")
   private Project project;
```

Thanks to this relation, you can access Task from Project such as id, status and name.

4. Create Indivilister's tasks

4-2. Create edit a task

- Let us see what' happening in the checkbox.

☑ Draw schematics

```
<script type="text/javascript">
  $(document).ready(function(){
    $('.task-status').on('click',function(){
      var status = $(this).prop('checked');
      window.location.href = '/task/edit/' + $(this).attr('data') + '/' + status
    });
  });
</script>
```

```
$(document).ready(function(){
                    When finished
loading html
        Execute ……————————
});
```

```
$('.task-status').on('click',function(){
      var status = $(this).prop('checked');
      window.location.href = '/task/edit/' + $(this).attr('data') + '/' + status
    });
```

When task-status is clicked
status = $(.task-status).prop(true); →checked=true unchecked=false
window.location.href = /task/edit/task.id/status"

window.location.href = PATH
Move to the URL PATH

So…we need a controller method to respond to this URL request.

4. Create Indivilister's tasks

4-2. Create edit a task

- Add the following code into TaskController.java

```
@RequestMapping(value = "edit/{task_id}/{status}", method = RequestMethod.GET)
  public String edit(@PathVariable("task_id") int taskId, @PathVariable("status") boolean status) {
     taskService.update4Status(taskId, status);
     return "redirect:/task/list/" + taskService.findProjectByTaskId(taskId).getId();
  }
```

Receive taskId and status from URL, update Task DB, and then redirected to the same page with the same project.

But… we haven't created updated4Status and findProjectByTaskId method yet.

4. Create Indivilister's tasks

4-2. Create edit a task

- Add the following code into TaskService.java

```
public void update4Status(int taskId, boolean status) {
    Task task = taskRepository.findOne(taskId);
    task.setStatus(status);
    taskRepository.save(task);
 }
```

Find one task by taskId and update status field(true/false) ---- checkbox

Continue on next page

4. Create Indivilister's tasks

4-2. Create edit a task

- Add the following code into TaskService.java

```
public Project findProjectByTaskId(int taskId) {
    Task task = taskRepository.findOne(taskId);
    return task.getProject();
}
```

Find one task by taskId and update status field(true/false) ---- checkbox

Again, thanks to this relation, you can access Project from Task, and return project.

```
@ManyToOne
  @JoinColumn(name = "project_id")
  private Project project;
```

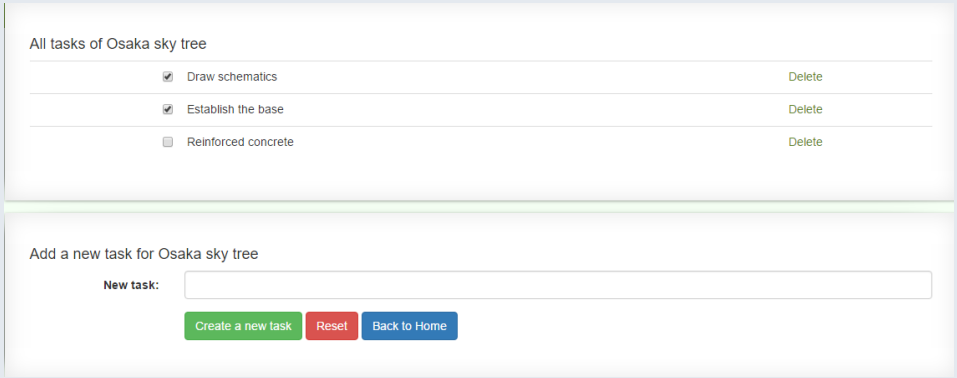
```
@RequestMapping(value = "edit/{task_id}/{status}", method = RequestMethod.GET)
  public String edit(@PathVariable("task_id") int taskId, @PathVariable("status") boolean status) {
    taskService.update4Status(taskId, status);
    return "redirect:/task/list/" + taskService.findProjectByTaskId(taskId).getId();
  }
```

Finally this part turns project Id --- /task/list/projectId

4. Create Indivilister's tasks

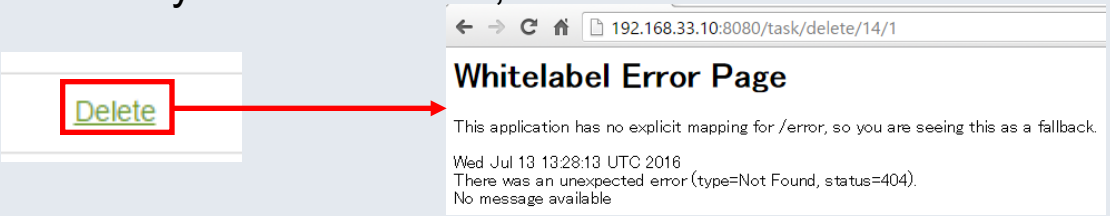
4-2. Create edit a task

- Restart your project and access server:8080/project



Create projects and make the checkbox checked.

But… if you delete a task, the error occurs.



4. Create Indivilister's tasks

4-3. Create delete a task

- Let's see "Delete" button code.

Delete

```
<a th:href="'/task/delete/' + ${project.id} + '/' + ${task.id}" onclick="return confirm('Delete this task?');">Delete</a>
```

A task has to be deleted by project.id and task.id

Add the following code into TaskController.java

```
@RequestMapping(value = "delete/{project_id}/{task_id}", method = RequestMethod.GET)
    public String delete(@PathVariable("project_id") int projectId, @PathVariable("task_id") int taskId) {
        taskService.delete(taskId);
        return "redirect:/task/list/" + projectId;
    }
```

But… we haven't created taskService.delete() method

4. Create Indivilister's tasks

4-3. Create delete a task

- Add the following code into TaskService.java

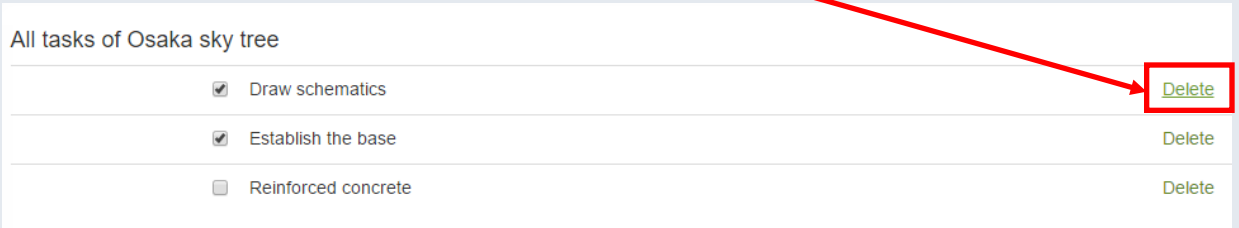
```
public void delete(int taskId) {
    taskRepository.delete(taskId);
  }
```

No delete() method is created in TaskRepository.java, this is also JpaRepository extended result.

4. Create Indivilister's tasks

4-3. Create delete a task

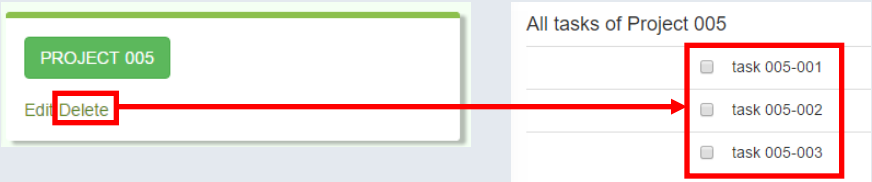
- Restart your project and access server:8080/project, and check if you can delete tasks.



4. Create Indivilister's tasks

4-3. Create delete a task

- Delete a project and its task at the same time. When a project is deleted, tasks under the project should also be deleted.



Add the following code into ProjectService.java

```
@Autowired
   ProjectRepository projectRepository;
@Autowired
   TaskRepository taskRepository;
```

Now you have a reference with @Autowired to TaskRepository in ProjectServce.

Continue on next page

4. Create Indivilister's tasks

4-3. Create delete a task

- Add the following code into delete method() in ProjectService.java

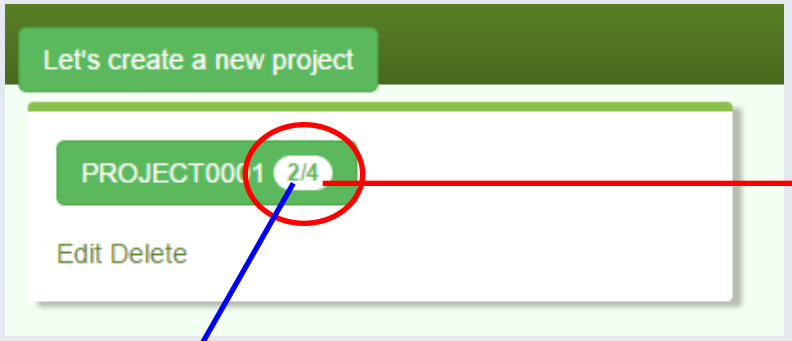
```
public void delete(int projectId) {
    taskRepository.deleteByProjectId(projectId);
    projectRepository.delete(projectId);
}
```

When this method is called from the delete button on a project, it will delete both the project and its tasks.

4. Create Indivilister's tasks

4-4. Count task

- We will create "count tasks" as below



The number of all tasks in the PROJECT001

The number of remaining tasks in the PROJECT001

project/list.html

```
<span class="badge" th:text="${projectRemainingTaskMap.get(project.id)} + '/' + ${project.taskList.size()}">1/2</span>
```

4. Create Indivilister's tasks

4-4. Count task

- Add the following code into ProjectService.java

```
public Map<Integer, Integer> calcRemainingTaskNumber(List<Project>
projectList) {
        Map<Integer, Integer> remainingTaskNumberMap = new HashMap();
        for(Project project : projectList) {
            int taskCount = 0;
            for(Task task : project.getTaskList()) {
                if(!task.isStatus()) {
                    taskCount++;
                }
            }
            remainingTaskNumberMap.put(project.getId(), taskCount);
        }
        return remainingTaskNumberMap;
    }
```

4. Create Indivilister's tasks

4-4. Count task

- Let's understand what's in this code

java.util.Map → public Map<Integer, Integer>
To store an element in Map(HashMap), put()method in HashMap
put(K key, V value)

Initialize Map object with HashMap and store it in remainingTaskNumberMap
Repeat 1-3 for projectList
1.Get TaskList() and repeat the number of task(s)
2.if(!task.isStatus()) → getter for boolean status field
3. status = false → taskCount = taskCount + 1

remainingTaskNumberMap.put(project.getId(), taskCount);
Store ("project.id as the key", taskCount as value)

So if you tell the key, you get the value.
In this case, you tell project.id, you will get the number of tasks(taskCount)

```
public Map<Integer, Integer> calcRemainingTaskNumber(List<Project> projectList) {
    Map<Integer, Integer> remainingTaskNumberMap = new HashMap();
    for(Project project : projectList) {
        int taskCount = 0;
        for(Task task : project.getTaskList()) {
            if(!task.isStatus()) {
                taskCount++;
            }
        }
        remainingTaskNumberMap.put(project.getId(), taskCount);
    }
    return remainingTaskNumberMap;
}
```

4. Create Indivilister's tasks

4-4. Count task

- Add the following code into index() method in ProjectController.java

```
@RequestMapping(value = "", method = RequestMethod.GET)
  public String index(Model model) {
      List<Project> projectList = projectService.findAllProject();
      model.addAttribute("projectList", projectList);
      model.addAttribute("projectRemainingTaskMap", projectService.calcRemainingTaskNumber(projectList));
    return "project/list";
  }
```

Remove the <!-- --> from the following code in project/list.html

```
<span class="badge" th:text="${projectRemainingTaskMap.get(project.id)} + '/' + ${project.taskList.size()}">1/2</span>
```

4. Create Indivilister's tasks

4-4. Count task

- Understanding the following code

```
<span class="badge" th:text="${projectRemainingTaskMap.get(project.id)} + '/' + ${project.taskList.size()}">1/2</span>
```

```
remainingTaskNumberMap.put(project.getId(), taskCount);
return remainingTaskNumberMap;
```

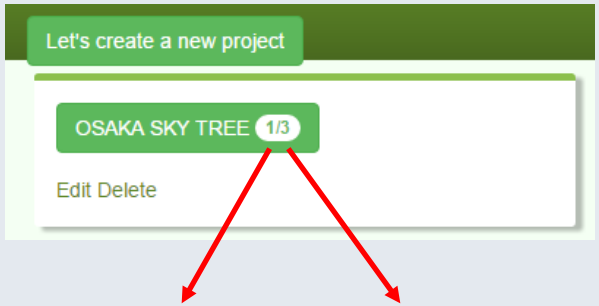
If you tell the key, you will get the value. If you tell project.id, you will get taskCount;

${project.taskList.size()}
size() is the number of elements in collection(List)
Ex. [aaaa, bbbb, cccc, dddd] 4 elements in collection.

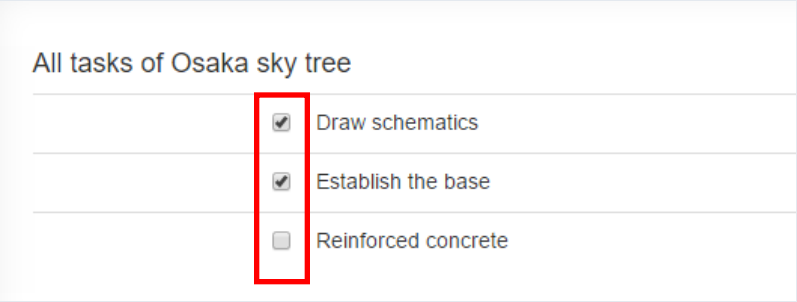
4. Create Indivilister's tasks

4-4. Count task

- Restart your project and access server:8080/project



Unfinished task:1 / All tasks:3



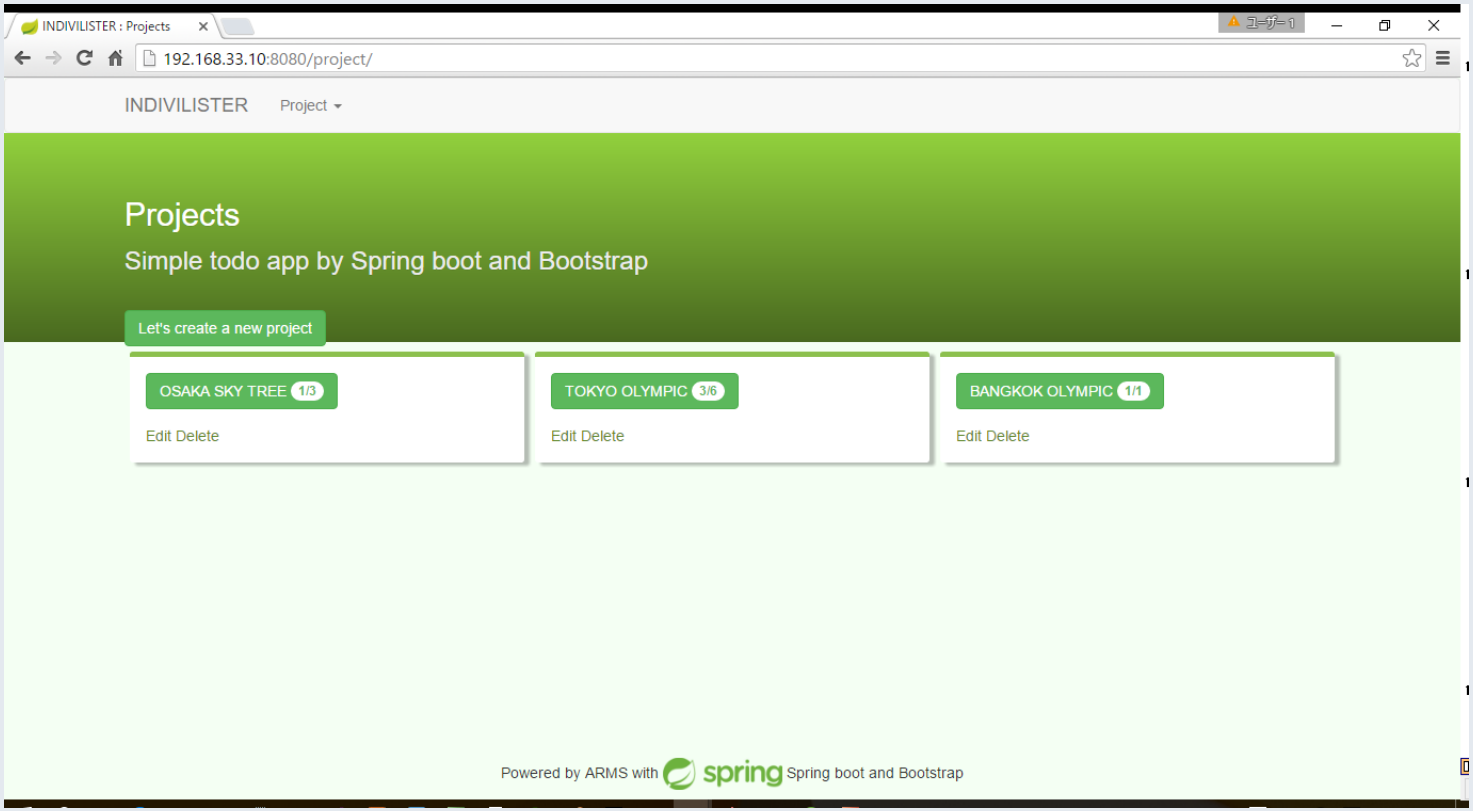
In this example:
Check finished task:2
Uncheck unfinished task:1
All tasks:3

**Unfinished task = remaining task

4. Create Indivilister's tasks

4-4. Count task

- Complete!



ARMS

Your Idea Leads Your Ideals

homepage: http://arms-asia.com/
facebook: https://www.facebook.com/arms.asia?fref=ts