

```
1
2 package acme.features.manager.project;
3
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Service;
6
7 import acme.client.data.models.Dataset;
8 import acme.client.services.AbstractService;
9 import acme.entities.project.Project;
10 import acme.roles.Manager;
11
12 @Service
13 public class ManagerProjectDeleteService extends AbstractService<Manager,
14     Project> {
15     // Internal state
16     -----
17     @Autowired
18     private ManagerProjectRepository repository;
19
20     // AbstractService interface
21     -----
22
23     @Override
24     public void authorise() {
25         boolean status;
26         int masterId;
27         Project project;
28
29         masterId = super.getRequest().getData("id", int.class);
30         project = this.repository.findProjectById(masterId);
31         status = project != null && project.isDraft() && super.getRequest
32             ().getPrincipal().hasRole(Manager.class) && project.getManager().getId() ==
33             super.getRequest().getPrincipal().getActiveRoleId();
34         super.getResponse().setAuthorised(status);
35     }
36
37     @Override
38     public void load() {
39         Project project;
40         int id;
41
42         id = super.getRequest().getData("id", int.class);
43         project = this.repository.findProjectById(id);
44
45         super.getBuffer().addData(project);
46     }
```

```
46
47     @Override
48     public void bind(final Project project) {
49         assert project != null;
50
51         super.bind(project, "code", "title", "abstractText",
52             "hasFatalErrors", "cost", "link", "isDraft");
53
54     @Override
55     public void validate(final Project project) {
56         assert project != null;
57     }
58
59     @Override
60     public void perform(final Project project) {
61         assert project != null;
62
63         this.repository.deleteAll
64             (this.repository.findAllAssignmentsOfAProjectById(project.getId()));
65         this.repository.deleteAll
66             (this.repository.findAllProgressLogsByProjectId(project.getId()));
67         this.repository.deleteAll
68             (this.repository.findAllContractOfAProjectById(project.getId()));
69
70         this.repository.deleteAll
71             (this.repository.findAllAuditRecordsOfAProjectById(project.getId()));
72         this.repository.deleteAll
73             (this.repository.findAllCodeAuditsOfAProjectById(project.getId()));
74
75         this.repository.deleteAll
76             (this.repository.findAllSponsorShipOfAProjectById(project.getId()));
77         this.repository.deleteAll
78             (this.repository.findAllObjectivesOfAProjectById(project.getId()));
79
80         this.repository.deleteAll
81             (this.repository.findAllTrainingSessionsOfAProjectById(project.getId()));
82         this.repository.deleteAll
83             (this.repository.findAllTrainingModuleOfAProjectById(project.getId()));
84         this.repository.delete(project);
85     }
86
87     @Override
88     public void unbind(final Project project) {
89         assert project != null;
90         boolean userStoriesPublishables;
91         boolean isDraft;
92
93         userStoriesPublishables =
94             this.repository.findAllUserStoriesOfAProjectById(project.getId()).stream
```

```
(()).allMatch(x -> x.isDraft() == false) &&
this.repository.findAllUserStoriesOfAProjectById(project.getId()).size() > 0
85     && project.isHasFatalErrors() == false;
86     isDraft = project.isDraft() == true;
87
88     Dataset dataset;
89
90     dataset = super.unbind(project, "code", "title", "abstractText",
"hasFatalErrors", "cost", "link", "isDraft");
91     dataset.put("masterId", project.getId());
92     dataset.put("publishable", userStoriesPublishables);
93     dataset.put("isDraft", isDraft);
94
95     super.getResponse().addData(dataset);
96 }
97
98 }
99
```