```java

package acme.features.client.contract;

import java.util.Collection;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import acme.client.data.models.Dataset;
import acme.client.services.AbstractService;
import acme.client.views.SelectChoices;
import acme.entities.contract.Contract;
import acme.entities.project.Project;
import acme.roles.Client;

@Service
public class ClientContractDeleteService extends AbstractService<Client, Contract> {

    @Autowired
    private ClientContractRepository repository;


    @Override
    public void authorise() {
        boolean status;
        Contract contract;
        int id = super.getRequest().getData("id", int.class);
        contract = this.repository.findContractById(id);
        status = contract != null && contract.isDraft() && super.getRequest().getPrincipal().hasRole(Client.class) && contract.getClient().getId() == super.getRequest().getPrincipal().getActiveRoleId();
        super.getResponse().setAuthorised(status);
    }

    @Override
    public void load() {
        Contract contract;
        int id;

        id = super.getRequest().getData("id", int.class);
        contract = this.repository.findContractById(id);

        super.getBuffer().addData(contract);
    }

    @Override
    public void bind(final Contract contract) {
        assert contract != null;

```

```
48          super.bind(contract, "code", "moment", "providerName",
"customerName", "goals", "budget", "isDraft");
49      }
50
51      @Override
52      public void validate(final Contract contract) {
53          assert contract != null;
54      }
55
56      @Override
57      public void perform(final Contract contract) {
58          assert contract != null;
59          this.repository.deleteAll
(this.repository.findAllProgressLogsByContractId(contract.getId()));
60          this.repository.delete(contract);
61
62      }
63      @Override
64      public void unbind(final Contract contract) {
65          assert contract != null;
66          boolean isDraft;
67          Collection<Project> projects;
68          projects = this.repository.findAllPublishedProjects();
69          SelectChoices choices;
70          choices = SelectChoices.from(projects, "title", contract.getProject
());
71          isDraft = contract.isDraft() == true;
72
73          Dataset dataset;
74
75          dataset = super.unbind(contract, "code", "moment", "providerName",
"customerName", "goals", "budget", "isDraft", "project");
76          dataset.put("contractId", contract.getId());
77          dataset.put("isDraft", isDraft);
78          dataset.put("project", choices.getSelected().getKey());
79          dataset.put("projects", choices);
80
81          super.getResponse().addData(dataset);
82      }
83
84 }
85
```