```java

package acme.features.client.progresslog;

import java.util.List;

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.entities.contract.Contract;
import acme.entities.contract.ProgressLog;
import acme.roles.Client;

@Service
public class ClientProgressLogPublishService extends AbstractService<Client, ProgressLog> {

    @Autowired
    private ClientProgressLogRepository repository;


    @Override
    public void authorise() {
        boolean status;
        int progressLogId;
        Contract contract;
        ProgressLog progressLog;
        progressLogId = super.getRequest().getData("id", int.class);
        contract = this.repository.findOneContractByProgressLogId(progressLogId);
        progressLog = this.repository.findOneProgressLogById(progressLogId);
        status = super.getRequest().getPrincipal().getActiveRoleId() == contract.getClient().getId() && progressLog != null && progressLog.isDraft() && contract != null && super.getRequest().getPrincipal().hasRole(Client.class);

        super.getResponse().setAuthorised(status);
    }

    @Override
    public void load() {
        ProgressLog object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findOneProgressLogById(id);
        super.getBuffer().addData(object);
    }

```

```
46      @Override
47      public void bind(final ProgressLog object) {
48          assert object != null;
49
50          super.bind(object, "recordId", "completeness", "comment",
    "registrationMoment", "reponsiblePerson", "isDraft");
51      }
52
53      @Override
54      public void validate(final ProgressLog object) {
55          assert object != null;
56
57          if (!super.getBuffer().getErrors().hasErrors("recordId")) {
58              ProgressLog existing;
59              existing = this.repository.findOneProgressLogByCode
    (object.getRecordId());
60              super.state(existing == null || existing.equals(object),
    "recordId", "client.progresslog.form.error.duplicated");
61          }
62          if (!super.getBuffer().getErrors().hasErrors("completeness")) {
63
64              Double objectCompleteness = object.getCompleteness();
65              List<ProgressLog> pls = this.repository.findBefore
    (object.getContract().getId());
66              ProgressLog lastVersion = this.repository.findOneProgressLogById
    (object.getId());
67              Double lastCompleteness = pls.get(0).getCompleteness();
68              boolean condition = objectCompleteness.equals
    (lastVersion.getCompleteness()) || objectCompleteness > lastCompleteness;
69              super.state(condition && objectCompleteness < 100,
    "completeness", "client.progresslog.form.error.completeness");
70          }
71
72      }
73
74      @Override
75      public void perform(final ProgressLog progresslog) {
76          assert progresslog != null;
77
78          progresslog.setDraft(false);
79
80          this.repository.save(progresslog);
81      }
82
83      @Override
84      public void unbind(final ProgressLog object) {
85          assert object != null;
86          Dataset dataset;
87
88          dataset = super.unbind(object, "recordId", "completeness",
```

```
  "comment", "registrationMoment", "reponsiblePerson", "isDraft");
89
90          dataset.put("isDraft", object.isDraft());
91
92          super.getResponse().addData(dataset);
93      }
94
95  }
96
```