```java

package acme.features.client.progresslog;

import java.util.Collection;
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.client.views.SelectChoices;
import acme.entities.contract.Contract;
import acme.entities.contract.ProgressLog;
import acme.roles.Client;

@Service
public class ClientProgressLogUpdateService 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);
    }

    @Override
    public void bind(final ProgressLog object) {
        assert object != null;

        super.bind(object, "recordId", "completeness", "comment", "registrationMoment", "isDraft", "reponsiblePerson");
    }

    @Override
    public void validate(final ProgressLog object) {
        assert object != null;

        if (!super.getBuffer().getErrors().hasErrors("recordId")) {
            ProgressLog existing;
            existing = this.repository.findOneProgressLogByCode(object.getRecordId());
            super.state(existing == null || existing.equals(object), "recordId", "client.progresslog.form.error.duplicated");
        }
        if (!super.getBuffer().getErrors().hasErrors("completeness")) {

            Double objectCompleteness = object.getCompleteness();
            List<ProgressLog> pls = this.repository.findBefore(object.getContract().getId());
            ProgressLog lastVersion = this.repository.findOneProgressLogById(object.getId());
            Double lastCompleteness = pls.get(0).getCompleteness();
            boolean condition = objectCompleteness.equals(lastVersion.getCompleteness()) || objectCompleteness > lastCompleteness;
            super.state(condition && objectCompleteness < 100, "completeness", "client.progresslog.form.error.completeness");
        }

    }

    @Override
    public void perform(final ProgressLog progresslog) {
        assert progresslog != null;

        this.repository.save(progresslog);
    }

    @Override
    public void unbind(final ProgressLog object) {
        assert object != null;
        Dataset dataset;
        boolean isDraft;
        SelectChoices choices;
        Collection<Contract> contracts = this.repository.findAllContractsByClientId(super.getRequest().getPrincipal().getActiveRoleId());

        choices = SelectChoices.from(contracts, "code", object.getContract());

        isDraft = object.isDraft() == true;

        dataset = super.unbind(object, "recordId", "contract", "completeness", "comment", "registrationMoment", "isDraft", "reponsiblePerson");
        dataset.put("contract", choices.getSelected().getKey());
        dataset.put("contracts", choices);
        dataset.put("isDraft", isDraft);
        super.getResponse().addData(dataset);
    }
}

```