```java

package acme.features.client.progresslog;

import java.util.Date;
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.helpers.MomentHelper;
import acme.client.services.AbstractService;
import acme.entities.contract.Contract;
import acme.entities.contract.ProgressLog;
import acme.roles.Client;

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

    @Autowired
    private ClientProgressLogRepository repository;


    @Override
    public void authorise() {
        boolean status;
        int contractId = super.getRequest().getData("masterId", int.class);
        Contract contract;
        contract = this.repository.findOneContractById(contractId);
        status = super.getRequest().getPrincipal().getActiveRoleId() == 
contract.getClient().getId() && contract != null && contract.isDraft() && 
super.getRequest().getPrincipal().hasRole(super.getRequest().getPrincipal
().getActiveRole());

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

    @Override
    public void load() {
        final int masterId = super.getRequest().getData("masterId", 
int.class);
        Contract contract;

        contract = this.repository.findOneContractById(masterId);

        ProgressLog progressLog = new ProgressLog();
        progressLog.setDraft(true);
        progressLog.setContract(contract);
        super.getBuffer().addData(progressLog);
    }

    @Override
    public void bind(final ProgressLog object) {
        assert object != null;
        Date currentMoment = MomentHelper.getCurrentMoment();
        Date registrationMoment = new Date(currentMoment.getTime());
        super.bind(object, "recordId", "completeness", "comment", "registrationMoment", "reponsiblePerson", "isDraft");
        int masterId = super.getRequest().getData("masterId", int.class);
        Contract contract = this.repository.findOneContractById(masterId);
        object.setContract(contract);
        object.setRegistrationMoment(registrationMoment);
    }

    @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, "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());
            if (!pls.isEmpty()) {
                Double lastCompleteness = pls.get(0).getCompleteness();
                super.state(objectCompleteness > lastCompleteness && objectCompleteness < 100, "completeness", "client.progresslog.form.error.completeness");
            } else
                super.state(objectCompleteness <= 100, "completeness", "client.progresslog.form.error.completeness");
        }

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

        this.repository.save(object);
    }

    @Override
    public void unbind(final ProgressLog object) {
        assert object != null;
        Dataset dataset;

        //SelectChoices contractChoices;
        //Collection<Contract> contracts = 
this.repository.findAllContractsByClientId(super.getRequest().getPrincipal
().getActiveRoleId());

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

        dataset = super.unbind(object, "contract", "recordId", 
"completeness", "comment", "registrationMoment", "isDraft", 
"reponsiblePerson");
        dataset.put("masterId", super.getRequest().getData("masterId", 
int.class));
        dataset.put("isDraft", object.getContract().isDraft());
        super.getResponse().addData(dataset);
    }
}

```