```java

package acme.features.manager.assignment;

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.project.Assignment;
import acme.entities.project.Project;
import acme.entities.project.UserStory;
import acme.features.manager.project.ManagerProjectRepository;
import acme.roles.Manager;

@Service
public class ManagerAssignmentCreateService extends AbstractService<Manager, Assignment> {

    @Autowired
    private ManagerProjectRepository repository;


    @Override
    public void authorise() {
        boolean status;
        int projectId = super.getRequest().getData("masterId", Integer.class);
        Manager manager = this.repository.findManagerByProjectId(projectId);
        status = manager != null && super.getRequest().getPrincipal().hasRole
(Manager.class) && manager.getId() == super.getRequest().getPrincipal().getActiveRoleId();

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

    @Override
    public void load() {

        Assignment assigment = new Assignment();
        Integer id;

        id = super.getRequest().getData("masterId", int.class);
        Project p = this.repository.findProjectById(id);

        assigment.setProject(p);

        super.getBuffer().addData(assigment);

    }

    @Override
    public void bind(final Assignment assigment) {
        assert assigment != null;

        super.bind(assigment, "project.title", "userStory");
    }

    @Override
    public void validate(final Assignment assigment) {
        assert assigment != null;

        if (!super.getBuffer().getErrors().hasErrors("userStory")) {
```

```
62              boolean condition = assigment.getProject().getManager().getId() == assigment.getUserStory().getManager().getId();
63              super.state(condition, "*", "manager.project.form.error.ownership");
64          }
65
66          if (!super.getBuffer().getErrors().hasErrors("project"))
67              super.state(assigment.getProject().isDraft() == true, "*", "manager.project.form.create-denied");
68
69          if (!super.getBuffer().getErrors().hasErrors("userStory")) {
70              int masterId = assigment.getProject().getId();
71              Collection<UserStory> us = this.repository.findAllUserStoriesOfAProjectById(masterId);
72              super.state(!us.contains(assigment.getUserStory()), "userStory", "manager.project.form.UsDuplicated");
73          }
74
75      }
76
77      @Override
78      public void perform(final Assignment assigment) {
79          assert assigment != null;
80
81          this.repository.save(assigment);
82      }
83
84      @Override
85      public void unbind(final Assignment assigment) {
86          assert assigment != null;
87
88          Dataset dataset;
89
90          int id = super.getRequest().getPrincipal().getActiveRoleId();
91          SelectChoices userStoriesChoices;
92
93          Collection<UserStory> userStories = this.repository.findAllUserStoriesOfAManagerById(id);
94
95          userStoriesChoices = SelectChoices.from(userStories, "title", assigment.getUserStory());
96
97          dataset = super.unbind(assigment, "project.title", "userStory");
98
99          int masterId = super.getRequest().getData("masterId", int.class);
100
101         dataset.put("masterId", masterId);
102         dataset.put("userStories", userStoriesChoices);
103
104         super.getResponse().addData(dataset);
105     }
106
107 }
108
```