Used project management

com.nomagic.magicdraw.core.project.ProjectsManager provides the used project management (project usage, export, import, reload, and package sharing) methods.

Related pages

- Project descriptor
- Project structure decomposition

Example #1. Exporting a used local project

The collection of project packages can be exported as a used project.

```
    ProjectsManager projectsManager = Application.getInstance().
getProjectsManager();
    File file = new File(moduleFilePath);
    ProjectDescriptor projectDescriptor = ProjectDescriptorsFactory.
createProjectDescriptor(file.toURI());
    // Export a collection of packages as a used project
    projectsManager.exportModule(project, packages, "My used local
project", projectDescriptor);
```

Example #2. Exporting a used server project

The *com.nomagic.magicdraw.teamwork.application.TeamworkUtils* class is used to export a used server project.

```
    TeamworkUtils.exportTeamworkModule(project, packages, "My used remote
project", remoteProjectQualifiedName);
```

Example #3. Using a project

The local and server project usage is similar - an appropriate project descriptor must be used, but module directories project option should be updated before using local project:

```
        ProjectsManager projectsManager = Application.getInstance().
getProjectsManager();
    File file = new File(moduleFilePath);
    // update project options only when using local project
    String moduleDir = file.getParent();
    ProjectOptions projectOptions = project.getOptions();
    List<String> directories = projectOptions.getModulesDirectories(true);
    if (!directories.contains(moduleDir))
    {
        projectOptions.addModuleDirectory(moduleDir);
    }
    ProjectDescriptor projectDescriptor = ProjectDescriptorsFactory.
createProjectDescriptor(file.toURI());
    // Use a project
    projectsManager.useModule(project, projectDescriptor);
```

Example #4. Importing a used project

The local and server project import does not differ. Just an appropriate project descriptor must be used:

```
        ProjectsManager projectsManager = Application.getInstance().
getProjectsManager();
    File file = new File(moduleFilePath);
    ProjectDescriptor projectDescriptor = ProjectDescriptorsFactory.
createProjectDescriptor(file.toURI());
    projectsManager.importModule(project, projectDescriptor);
```

Example #5. Reloading a used project

The local and server project reload does not differ. Just an appropriate project descriptor must be used:

```
    ProjectsManager projectsManager = Application.getInstance().
getProjectsManager();
    File file = new File(moduleFilePath);
    ProjectDescriptor projectDescriptor = ProjectDescriptorsFactory.
createProjectDescriptor(file.toURI());
    projectsManager.reloadModule(project, projectDescriptor);
```

Example #6. The package sharing

```
    ProjectsManager projectsManager = Application.getInstance().
getProjectsManager();
    SessionManager.getInstance().createSession("Create shared package");
    // Create a package to share
    Package aPackage = project.getElementsFactory().
createPackageInstance();
    aPackage.setOwner(project.getModel());
    aPackage.setName("myShare");
    SessionManager.getInstance().closeSession();

    // Share a package
    projectsManager.sharePackage(project, Arrays.asList(aPackage), "my
used project");


    // Save a project
    ProjectDescriptor projectDescriptor = ProjectDescriptorsFactory.
getDescriptorForProject(project);
    projectsManager.saveProject(projectDescriptor, true);
```

Example #7. Editing a used project within the project

```
    final String moduleFileName = new File(moduleFilePath).getName();
    // Find a used project directly used by the project (the primary
project)
    final IAttachedProject attachedProject = ProjectUtilities.
findAttachedProjectByName(project, moduleFileName, false);
    if (attachedProject != null)
    {
        final ModuleUsage moduleUsage = new ModuleUsage(project.
getPrimaryProject(), attachedProject);
        final Set<ModuleUsage> moduleUsages = Collections.singleton
(moduleUsage);

        final boolean readOnly = attachedProject.isReadOnly();
        if (readOnly)
        {
            // Make the used project editable (the read-write
accessibility mode)
            ModulesService.setReadOnlyOnTask(moduleUsages, false);
        }
        // Get the first shared package of the used project
        final Collection<Package> sharedPackages = ProjectUtilities.
getSharedPackages(attachedProject);
        final Package aPackage = sharedPackages.iterator().next();
        // Create a class in the used project
        SessionManager.getInstance().createSession("Create use case in
used project");
        final Class aCase = project.getElementsFactory().
createClassInstance();
        aCase.setOwner(aPackage);
        aCase.setName("myClass");
        SessionManager.getInstance().closeSession();
        // save the used project
        Application.getInstance().getProjectsManager().saveModule(project,
attachedProject, true, false);
        if (readOnly)
        {
             // Make a used project not editable (the read-only
accessibility mode)
             ModulesService.setReadOnlyOnTask(moduleUsages, true);
        }
    }
```

Advanced management

Use the *com.nomagic.magicdraw.core.ProjectUtilities* utility class for retrieving more project decomposition related information.

Use the *com.nomagic.magicdraw.core.modules.ModulesService* or *ProjectsManager* classes for managing attached projects.