Lab Exercise: Setting up Jenkins to run and report Test Coverage

In this exercise you will setup Coverage reporting on our Jenkins server, with build and test of one of your exercises.

The Jenkins Project will pull the latest version from your GitHub repository, before it builds and tests this latest version, when it is triggered manually or automatically.

You have already set up GitHub to trigger your job on the Jenkins server, when new file versions are pushed to your repository on GitHub. This should trigger all Jenkins jobs, that use this repository.

---

1. One of the team members must download the **dotCoverCoverageConfigCore.xml** file from BB to your solution folder

2. Edit to match your solution and tests. Take care you have the correct folder path and file names

3. Add it to version control and commit it. This is very important, because your Jenkins job gets it from the repo

4. Push your local repository to the remote repository

5. Create together a Jenkins New Item. Use the already defined project **TemplateCoreBuildCoverage** to copy from, and fill in your repository and other relevant information, as indicated in the template project – e.g. the correct name of the solution and of the .dll containing your test cases, including the correct folder paths

6. Use the Build Now option to verify, that the project can obtain the code from the repository, build and run the tests, **AND** at the same time calculate test coverage. Correct until this is the case. A Coverage Report menu command must appear in the left menu under your Jenkins project (refresh the page using F5).

7. All other team members pull from the remote repository and checks, that dotCoverCoverageConfig.xml is now a part of the solution.

8. Now, one from your team shall create a change in the project. Build and Test. Commit the change to the local repository. Finally do a sync: Pull – Push to your remote GitHub repository.

9. Goto the Jenkins server. After a few seconds (10-15), a new Build for your Jenkins project should start automatically, if you have set up your webhook correctly previously, and your new job. If not, check want went wrong, looking at the GitHub server and at the Jenkins server.

10. If you want to, you can disable the Jenkins job you made last week, as unit testing is now done by the coverage job.