EGit Практическо ръководство

Ас. Пенчо Малинов

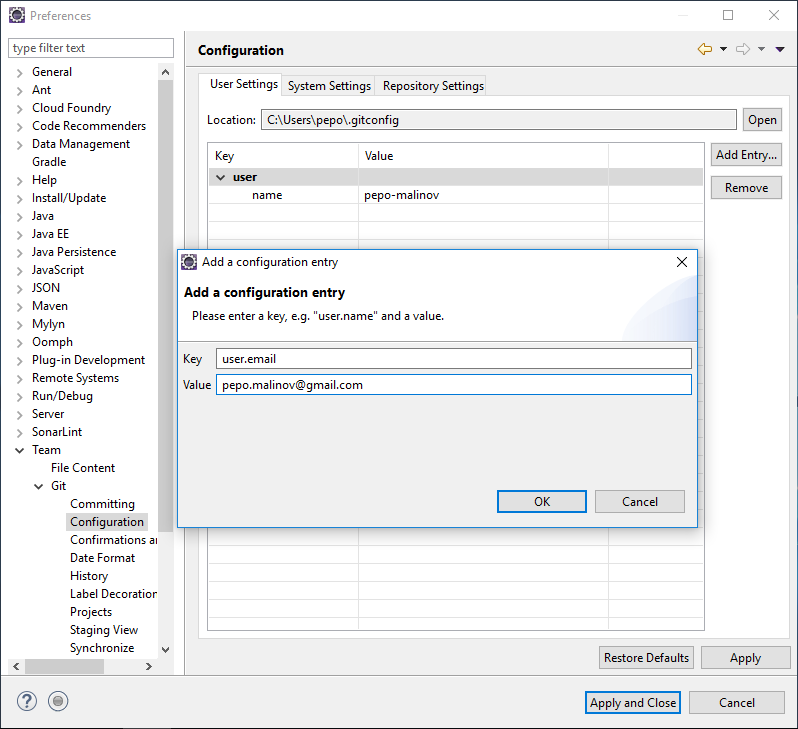
# Въведение

Това ръководство е помощен материал към дисциплината „Софтуерни инструменти“, част от курса на обучение в магистърска програма “Софтуерни технологии със специализация софтуерни архитектури и средства”. Ръководството обхваща основните случаи на използване на EGit, в допълнение към бележките как да конфигурирате, създавате и управлявате вашите първи хранилища. Наръчникът е насочен към разработчиците с основни познания за процесите на Git. Необходими изисквания:

* 1. регистрация в [GitHub](https://github.com/)
  2. инсталиран [Eclipse](https://www.eclipse.org/downloads/) – версия „Juno“, или по – нова.

# Конфигурация

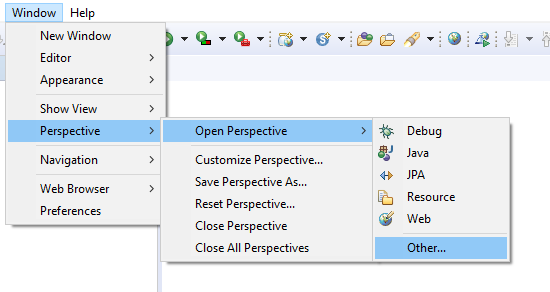
EGit официално е част от стандартната инсталация на Eclipse от версия “Juno”. Необходимо е само да се конфигурира началната информация за потребителя. За целта навигираме до Window => Preferences =>Team => Git => Configuration. Натискаме бутона „New Entry“ и добавяме в полето „Key“ “user.name”. В полето „Value“ добавяме потребителското си име в GitHub. Аналогично конфигурираме пощата с която е регистриран профила ни в GitHub.



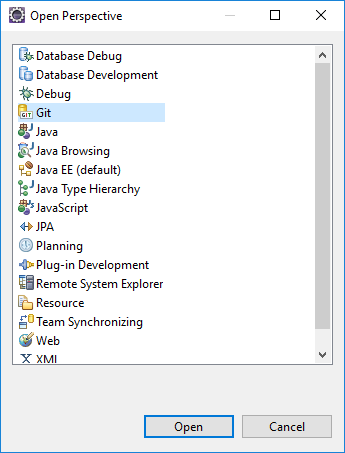
# Добавяне на Git перспектива

За да използваме EGit може да използваме съответната перспектива. Навигираме до:

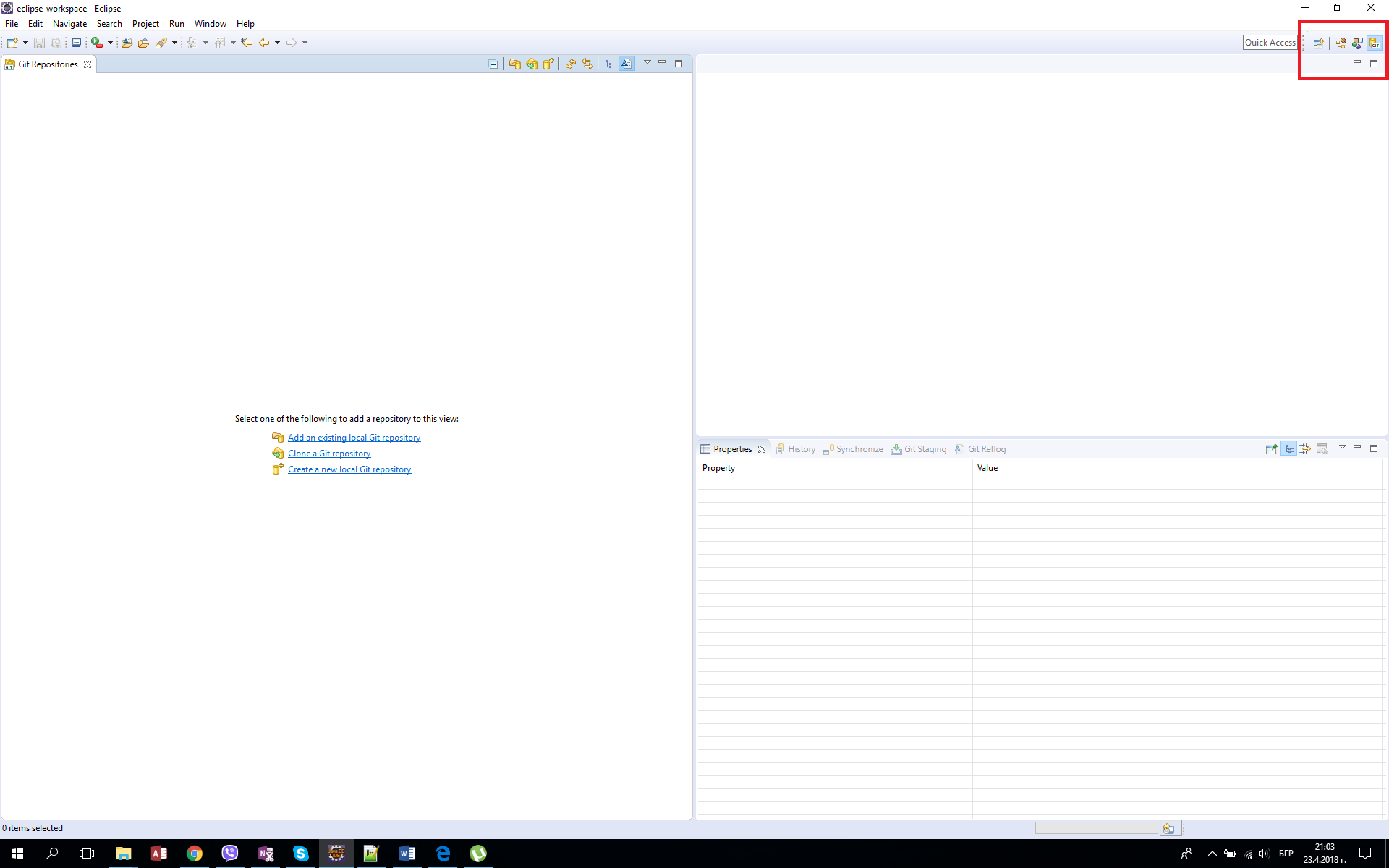
Window =>Perspective=>Open Perspective=>Other



От появилия се диалогов прозорец избираме Git и натискаме бутона „Open“



В резултат ни се отваря съответната перспектива. В горния десен ъгъл(оградения в червено) се намират различните перспективи за използване. Там е и бутона за Git перспективата. Разполагаме с няколко изгледа, които ще ни помогнат в работата ни с Git.



Разполагаме с няколко изгледа, които ще ни помогнат в работата ни с Git.

В дясно се намира изгледа „Git Repositories”. В него има 3 основни бутона:



Първият е за добавяне на съществуващо локално хранилище. Вторият е за клониране на външно хранилище. Третият е за създаване на ново локално Git хранилище.

В десния долен край на екрана се намират други помощни изгледи.

# Създаване на локално хранилище(Repository)

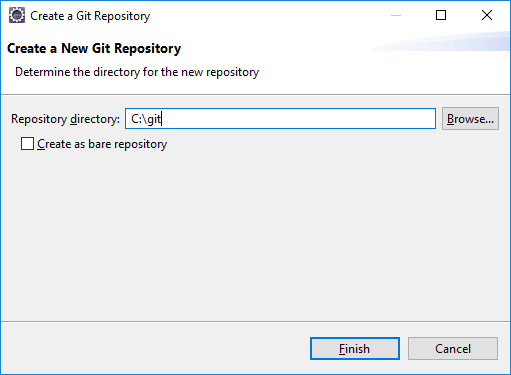
Един от големите плюсове на Git пред SVN и CVS е лесното създаване на локални хранилища, дори преди да ги споделите с други хора. По този начин може да управлявате версиите на работата ви по различни проекти локано(на вашия компютър). Добавянето на ново локално хранилище може да стане по два начина. Чрез бутона за създаване на хранилище в Git перспективата, или директно чрез добавяне на контрол над даден проект. По- долу ще разгледаме и двата подхода

1. Добавяне на локално хранилище през Git перспективата на Eclipse

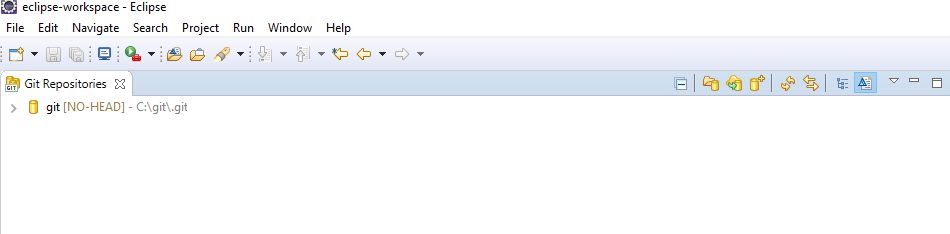
В Git перспективата натискаме върху бутона:



Отваря се диалогов прозорец в който трябва да посочим директорията за новото локално хранилище. Каквото име сложим на папката, така ще бъде именувано и хранилището.

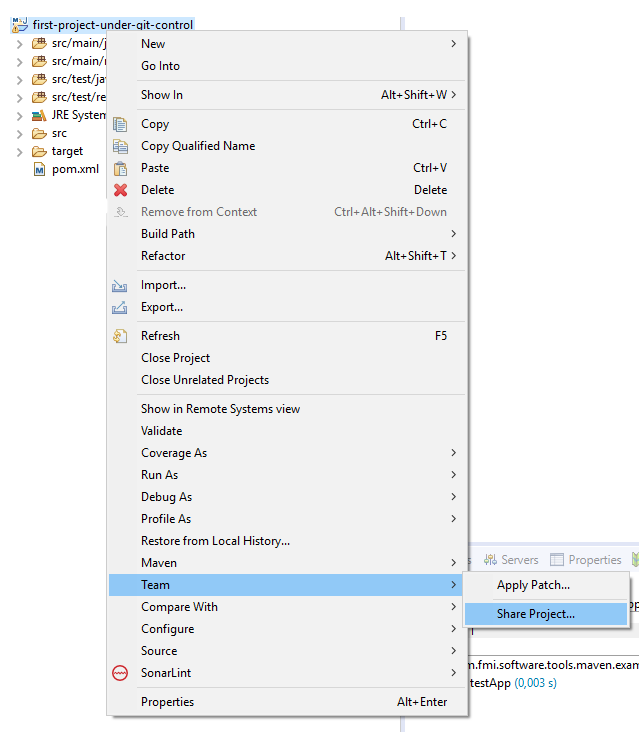


След натискане на бутона „Finish“ на екрана ще се почви новото хранилище

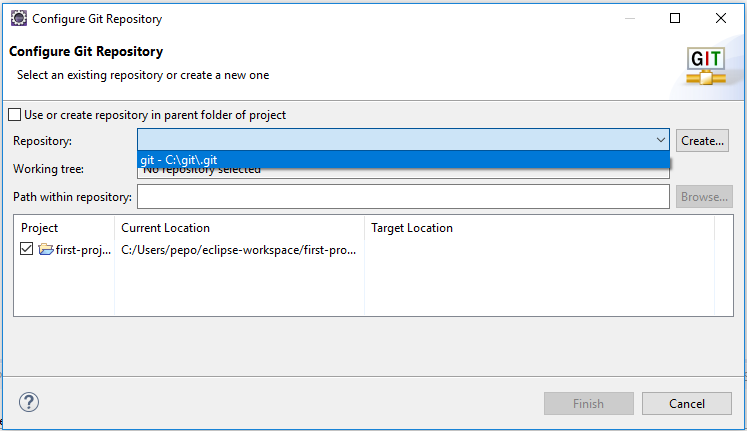


1. Добавяне на локално хранилище чрез вкарването на проект под контрол на версиите

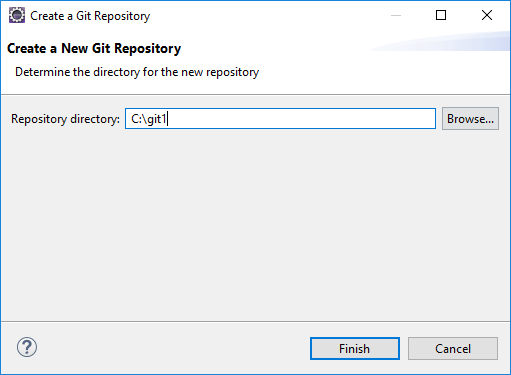
Избираме проект, на който желаем да управляваме версиите с Git. С десен бутон на мишката натискаме върху проекта и от контекстното меню навигираме до Team=>Share Project…



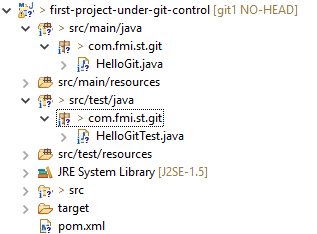
От диалоговия прозорец може да изберем съществуващо хранилище, или да създадем ново.



Стъпките за създаване на ново хранилище са аналогични на предходния пример



След като сме добавили избрания проект към хранилището са настъпили някои визуални промени с него. Появил се е знака „?“ върху него и върху част от папките на проекта. Всичко отбелязано с този знак е ново за Git още не се намира в хранилището.



# 5 “Cammit” на промените

За да съхраним промените(новите файлове от проекта) в хранилището е необходимо да направим “Commit”. За целта ще ползваме „Git Staging“ изгледа, намиращ се в долния ляв прозорец на Eclipse в Git перспективата. (Този изглед може да се добави към всяка перспектива). В него имаме 4 основни секции:

- Unstaged Changes – тук се намират всички нови или променени за Git файлове.

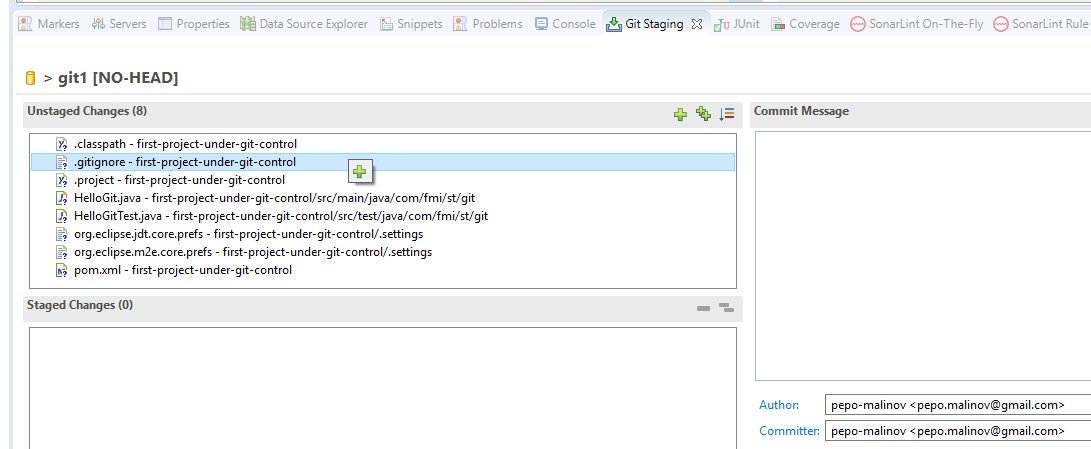
- Staged Changes – тук ще поставим всички променени или нови файлове които искаме да запазим в хранилището. Става чрез преместване на файловете в тази секция.

- Commit Message – В този прозорец трябва да се опишат какви са промените които сме направили. Това съобщение трябва да е достатъчно говоримо и обяснително, за да може без да се проверяват какви са промените да имаме представа за съдържанието на камита. Примерно номер на задача с нейното заглавие.

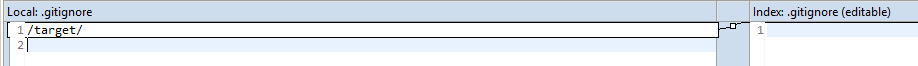
- Потребителски данни. В полетата “Author” и „Committer“ е необходимо да добавим информация за потребителя, който прави камита. Те по подрабиране са попълнини с данните, които сме попълнили в конфигурацията на Git.

Има и два бутона:

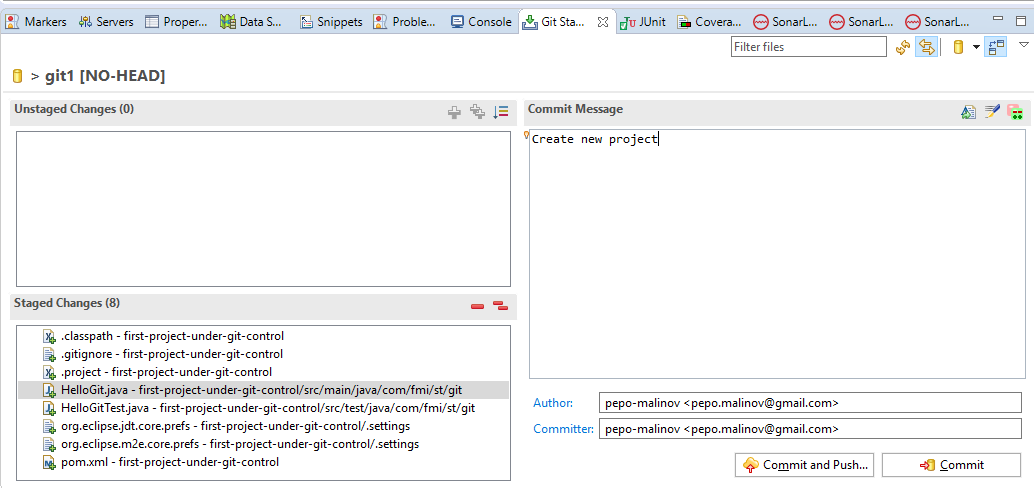
* Commit – С натискането на този бутон промените които сме решили да съхраним ще се запишат в локалното хранилище.
* Commit and Push… - С този бутон желаните за съхранение промени ще се запишат първо в локалното хранилище и веднага след това ще се запишат и в отдалеченото хранилище.



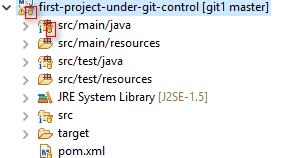
Ще обърне внимание на един файл – “gitignore”. В този файл декларираме всички файлове и папки, които не желаем да са под контрола на Git. Може да видите че в момента папката “target” е упомената в този файл => тя не е маркирана с „?“.



След като преместим желаните нови за Git файлове в прозореца са съхранение виждаме че въпросителният знак е заменен със зелен знак „+“.Гит ни информира че тези нови файлове ще се добавят в хранилището. Добавяме съобщение и натискаме бутона „Commit“. Нямаме настроено още отдалечено хранилище, затова не натискаме другия бутон.

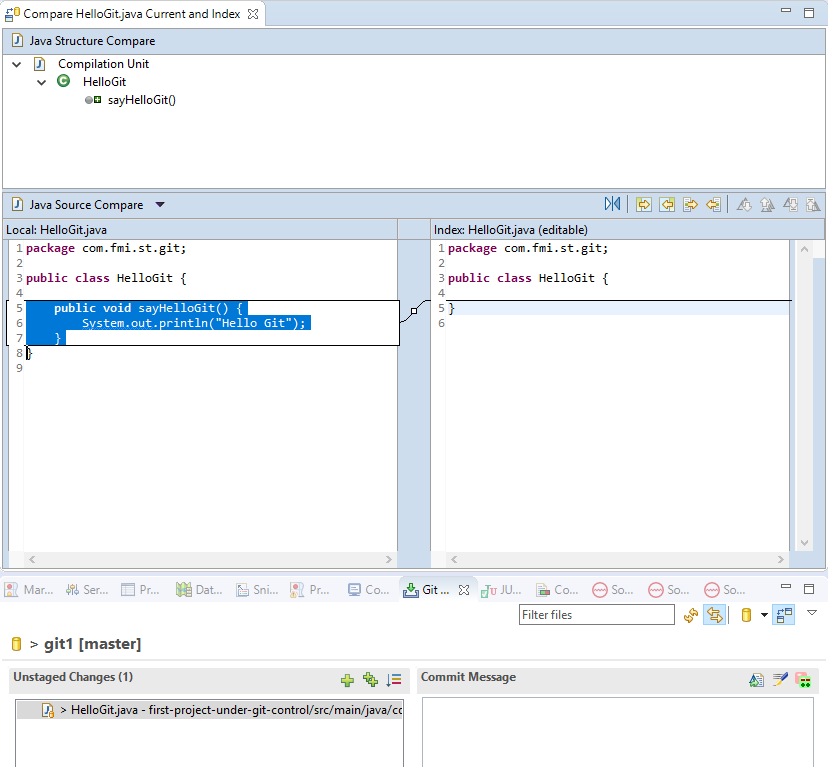


След като сме запазили промените в хранилището се забелязва отново визуално промяна в проекта, папките и файловете които са вече добавени към хранилището.



# 6 Сравнение на текущото състояние спрямо последните промени в хранилището

След като направим промени във файлове под контрола на Git, можем да ги прегледаме преди да ги камитнем. Това става като кликнем 2 пъти върху файла, намиращ се в „Unstaged Changes“. Файла се отваря в изглед за сравнение с версията налична в хранилището.



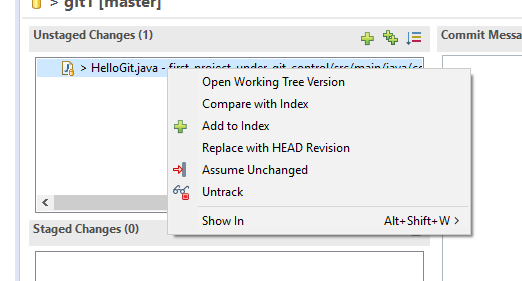
С Натискане на десет бутон на мишката върху файла може да видим няколко възможни действия които можем да предприемем:

- „Open Working Tree Version“ – отваря файла в текущия вариант (основния работен прозорец)

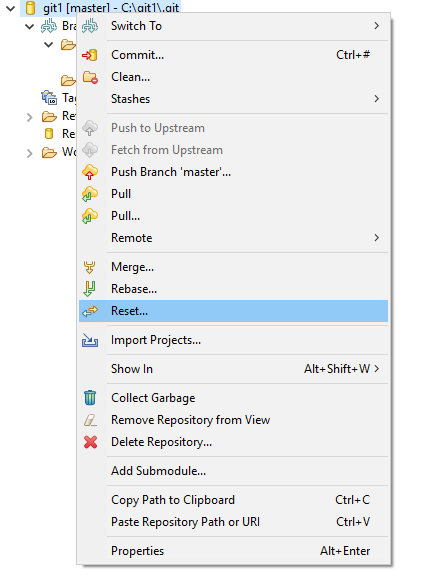
- „Compare with Index“ – файла се отваря в специален изглед за сравнение на текущата версия на файла с последната версия в хранилището.

- „Add to Index“ - премества файла в прозореца “Staged Changes“ от където да се камитне.

- „Replace wit HEAD Revision“ – премахва всички направени промени спрямо последната версия в хранилището

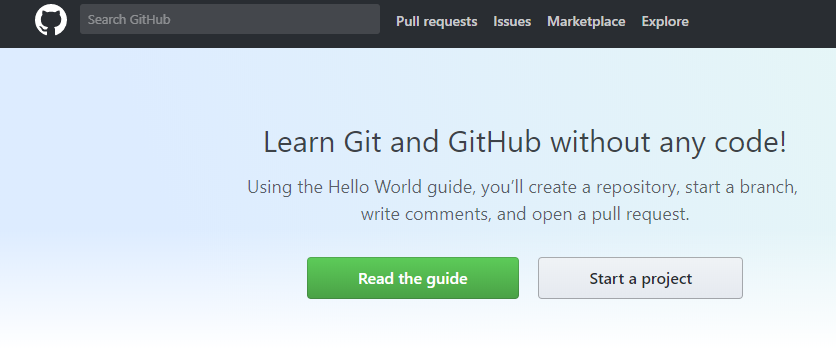


За да премахнем всички промени и да се върнем към последната версия на проекта от хранилището можем да ползваме “Reset” функционалността.

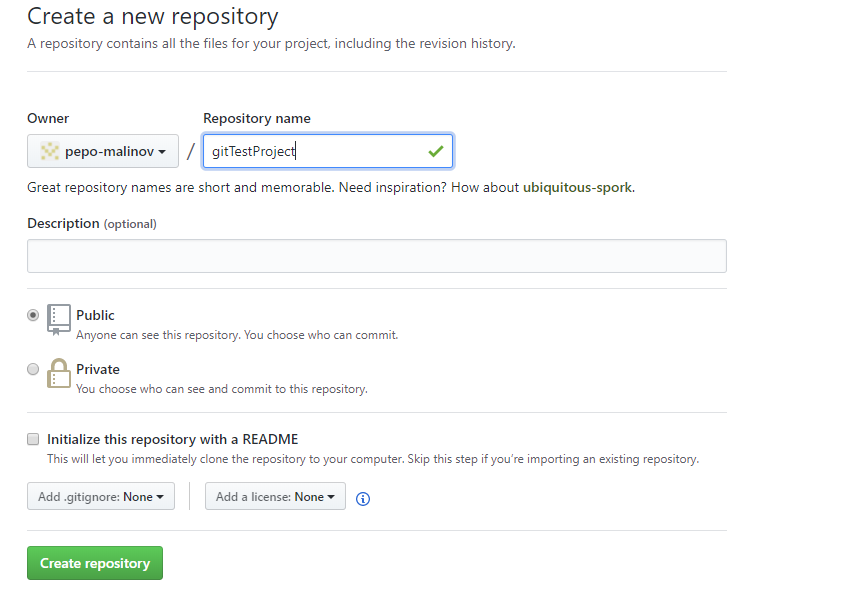


# 7 Създаване на отдалечено хранилище

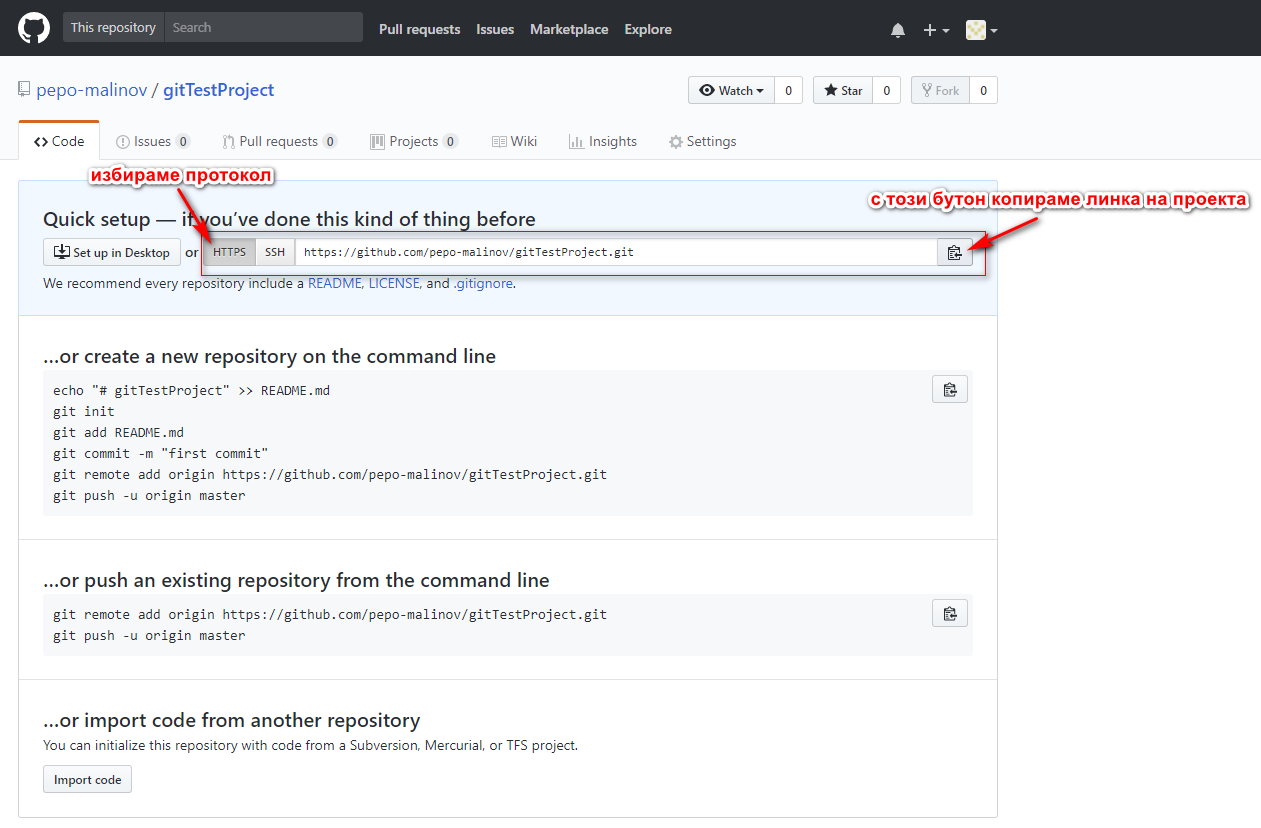
За да създадем отдалечено хранилище ще използваме регистрация в GitHub. Натискаме върху бутона „Start a project“



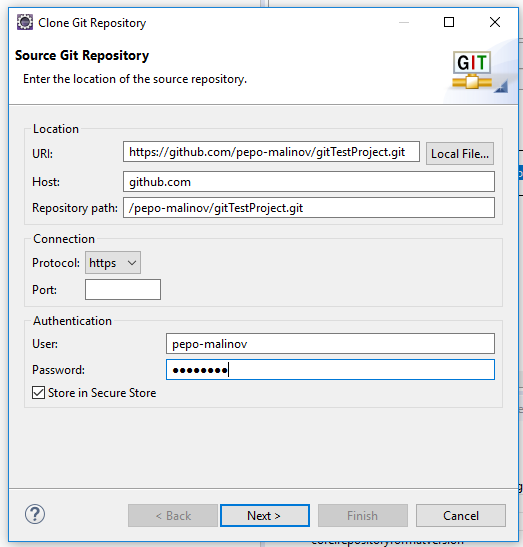
След като конфигурираме основните компоненти на хранилището натискаме върху бутона „Create repository“



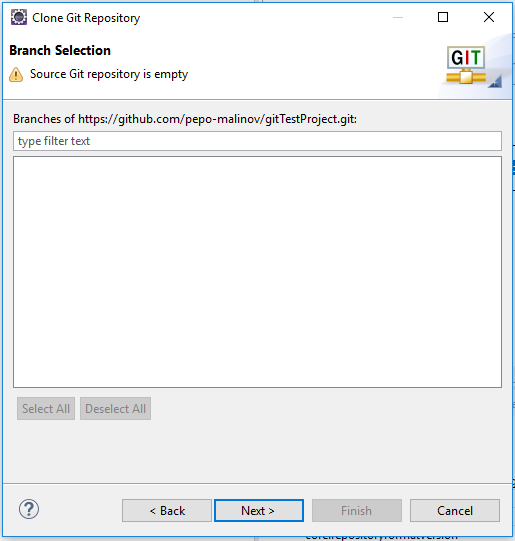
За да работим с новото хранилище на локалната ни машина е необходимо да ко клонираме. Копираме линка който виждаме по долу.



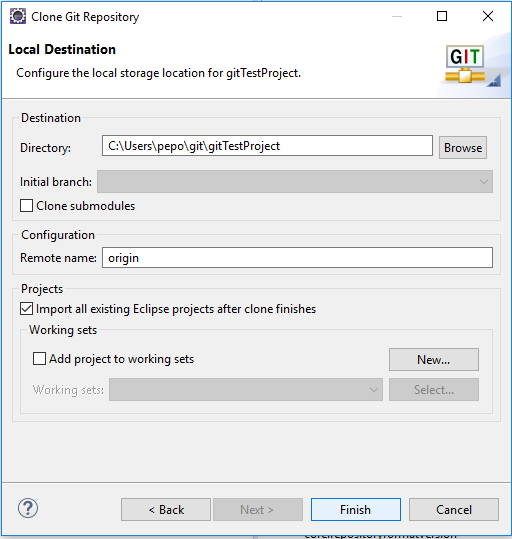
След като сме копирали линка отваряме Git перспективата в Eclipse и натискаме Бутона за клониране на хранилище. Отваря ни се диалогов прозорец, като по голямата част от полетата са попълнени, благодарение на копирания линк. Попълваме името и паролата си от GitHub и натискаме на бутона„Next“



Отваря се диалогов прозорец с информация за наличните клонове. Тъй като няма още нищо в хранилището виждаме и съответното съобщение. Отново натискаме „Next“



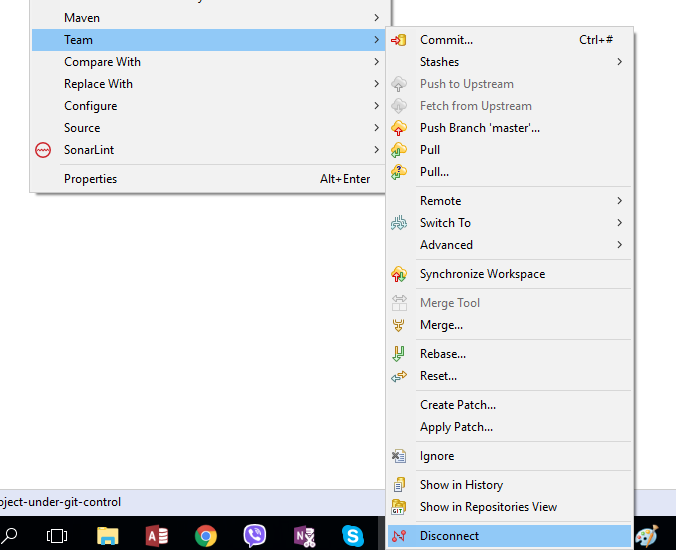
В последния екран посочваме папката в която да се намира новото локално хранилище, което е свързано с външно. Можем да кликне на отметката да се добавят автоматично проектите намиращи се в хранилището, ако има такива.



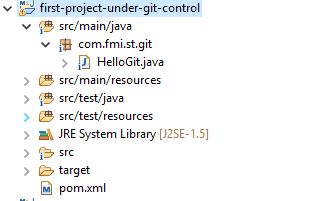
След като приключим с тази процедура резултата е ново празно локално хранилище.



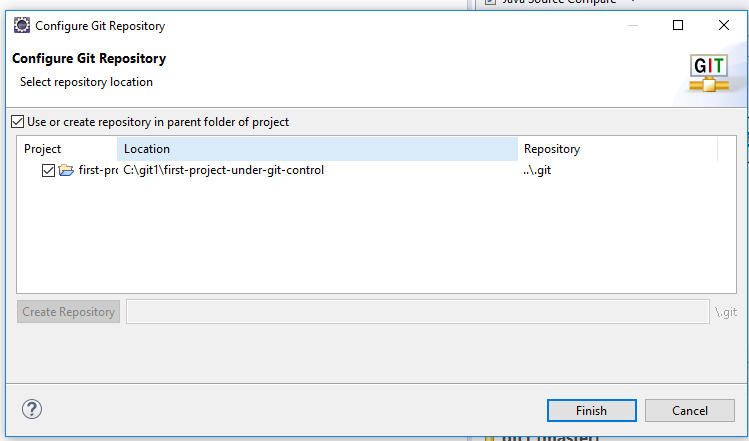
Ако искаме да добавим проекта с който работихме до сега, първо трябва да го откачим от хранилището за което е настроен. Това става с през контекстното меню на проекта Team=>Disconnect.

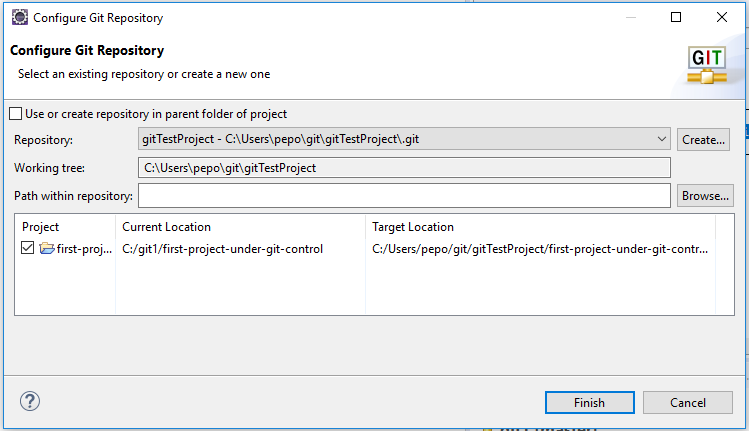


Изчезват всички визуални индикатори сочещи че проекта е под контрол на Git.

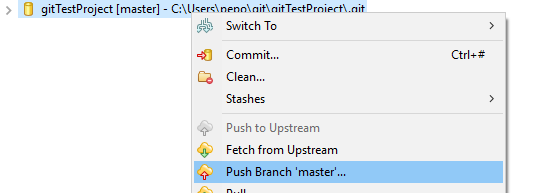


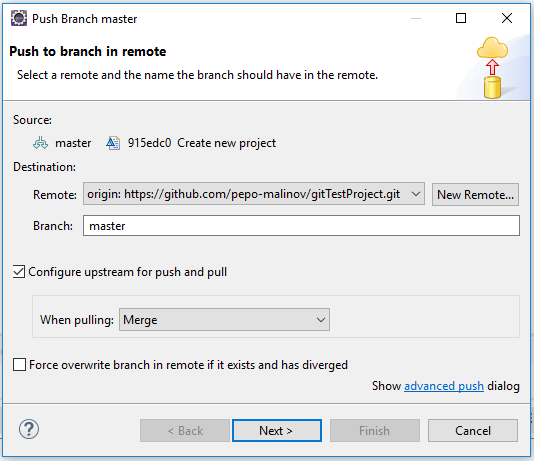
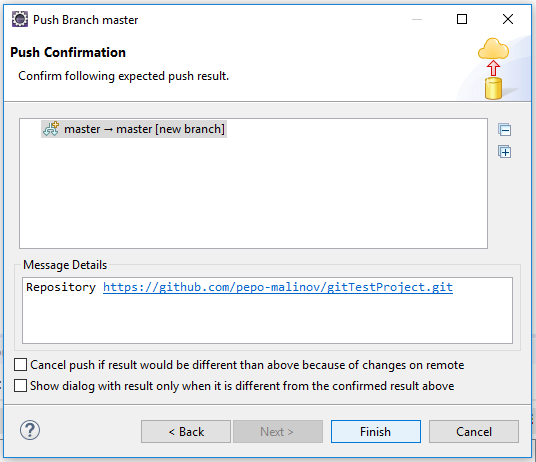
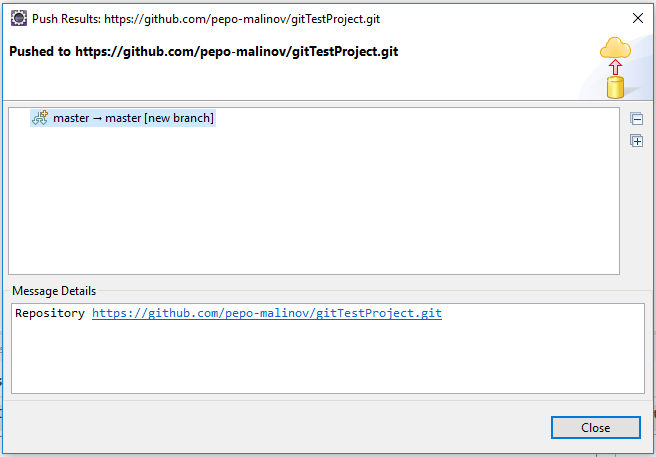
За да го добавим в новото хранилище минаваме през познатите стъпки: Team=>Share Project…



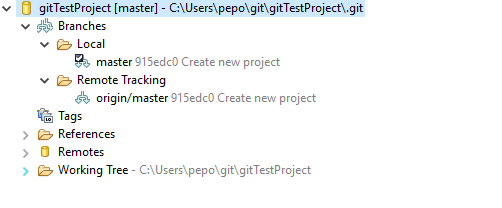


Също така трябва да камитнем новия за хранилището проект. Отново Проекта е под контрола на Git, но в ново хранилище. Промените които сме направили са все още само локално на нашата машина. За да ги споделим с отдалеченото хранилище отиваме в Git перспективата и с десен бутон на мишката върху новото хранилище натискаме “Push Branch ‘master’…”. По този начин добавяме новия клон(преди да камитнем java проекта нямахме налични клонове) към отдалеченото хранилище. **Всеки локален клон може да стане и отдалечен чрез същите стъпки.**



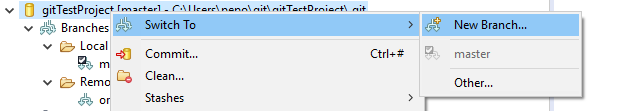


Можем да видим наличието на отдалечен и локален клон. Вече може когато камитваме промени да ползваме бутона 

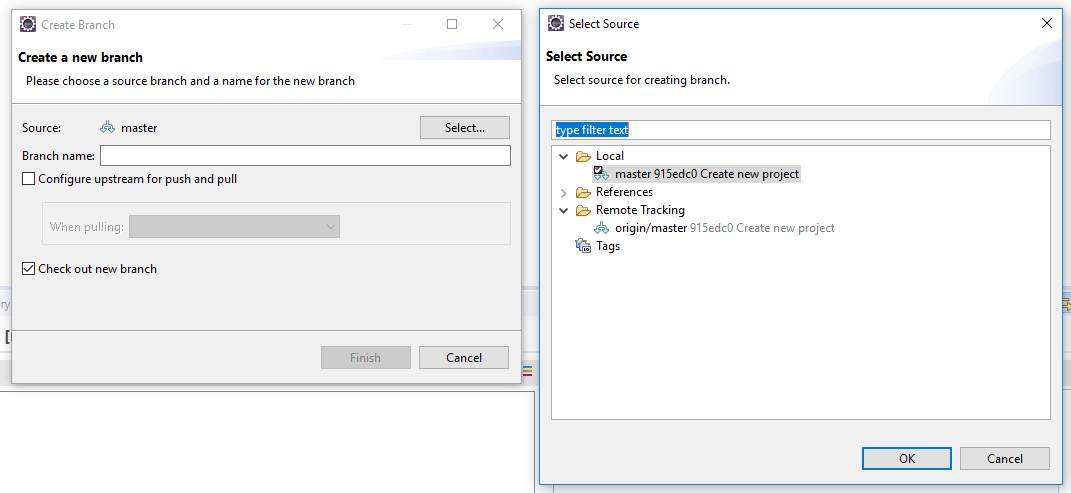


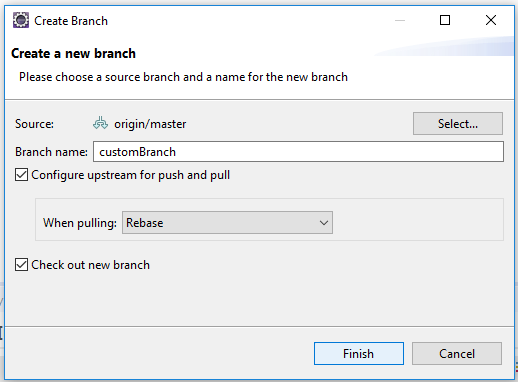
# 8 Създаване на локален клон(branch)

За да създадем нов локален клон, в Git перспективата от контекстното меню на хранилището избираме „Switch To=>New Branch…“

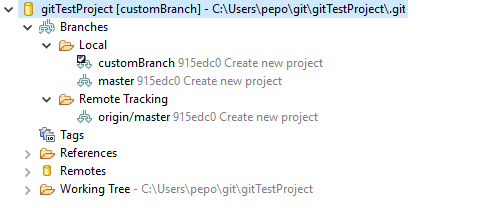


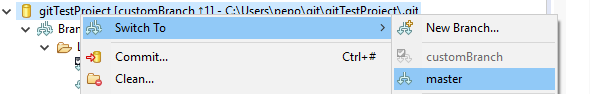
Можем да конфигурираме името на новия клон и източника му на данни(Може да е всеки друг клон – локален или отдалечен)





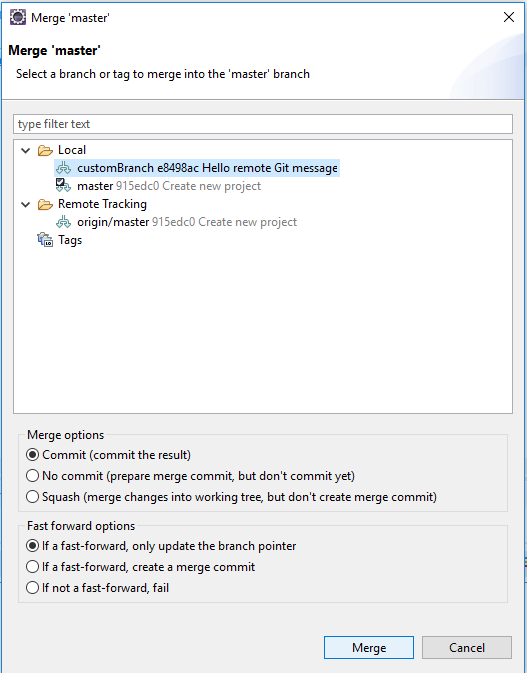
Преминаването от един клон към друг става по аналогичен начин. Разликата е че вместо да създаваме нов клон избираме някой от наличните



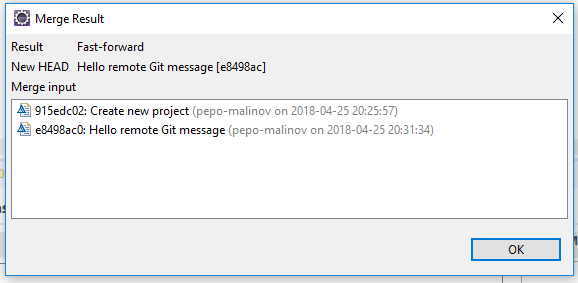


# 9 Сливане на промените от различни клонове

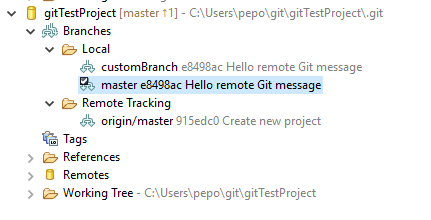
За да пренесем промени от един клон към друг използваме бутона „Merge“ от контекстното меню на хранилището.

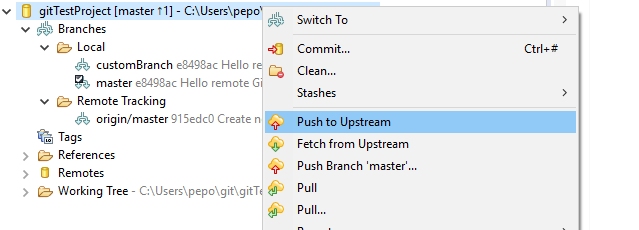


Избираме клона от който ще взимаме промени и натискаме бутона “Merge”

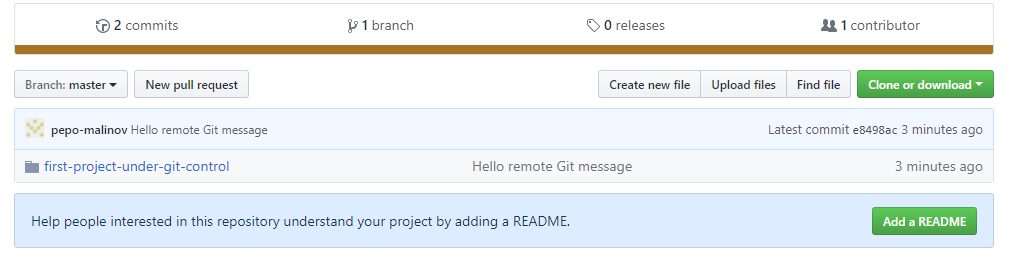


В резултат имаме камит. В името на хранилището се вижда към кой клон сме се закачили. Също така има и една стрелка нагоре с числото 1. Това означава, че има един камит, който е само локален. За да пренесем промените към отдалеченото хранилище използваме Бутона „Push to Upstream“



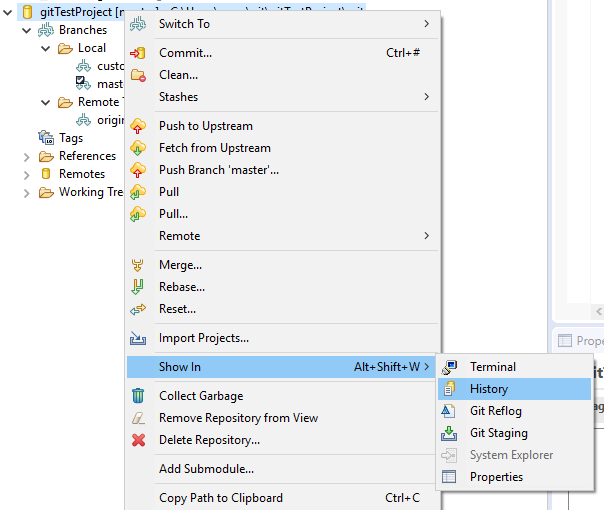


Промените може да видим в GitHub.

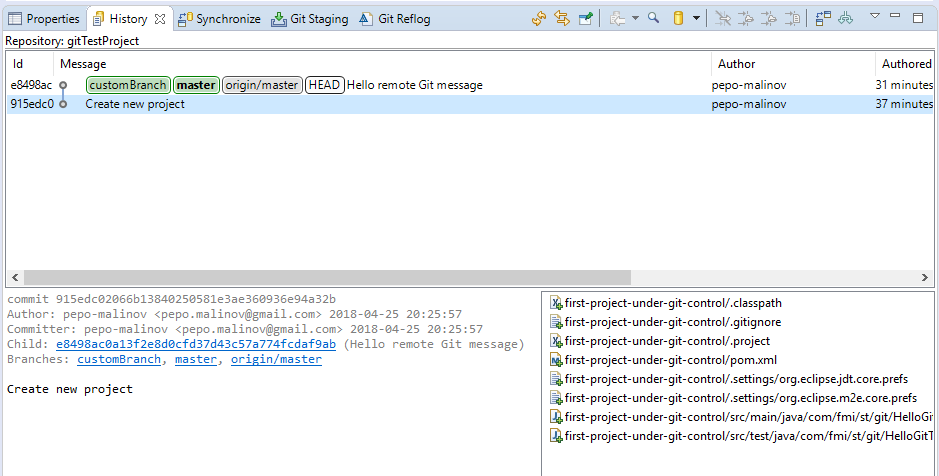


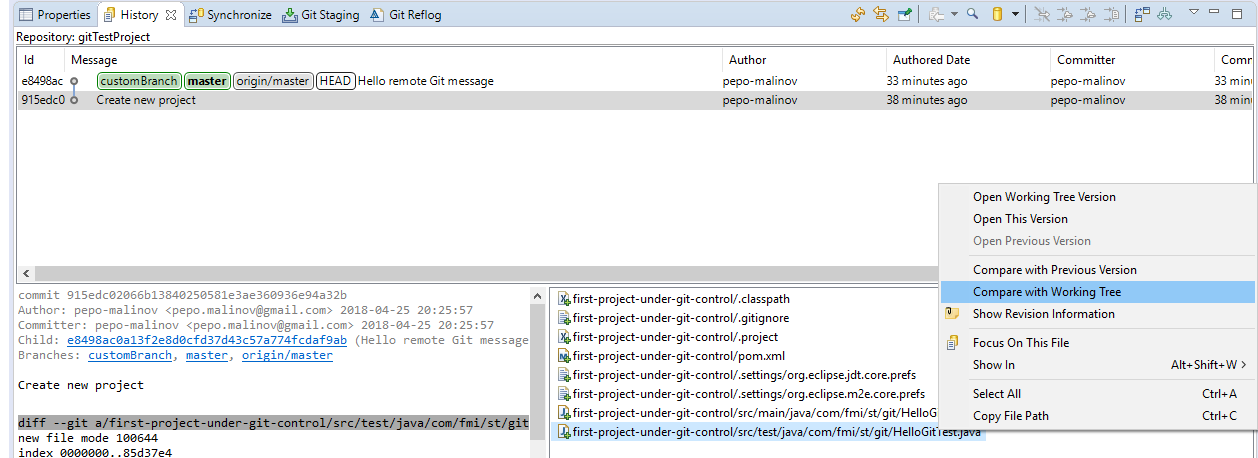
# 10 Използване на изгледа „History“

EGit идва с комплект различни помощи изгледи. Един от тях е „History“. Навигираме от контекстното меню на хранилището до “Show In=>History”

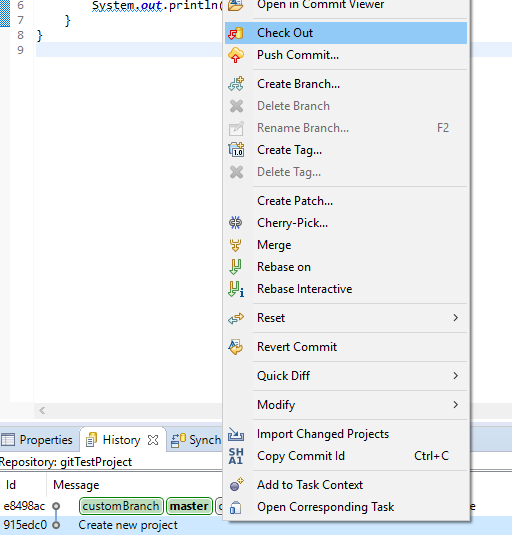


Новия изглед ни предоставя възможност да видим какви камити сме правили, кои са променените файлове, кой ги е променил и какви са промените спрямо други по нови или предходни версии.



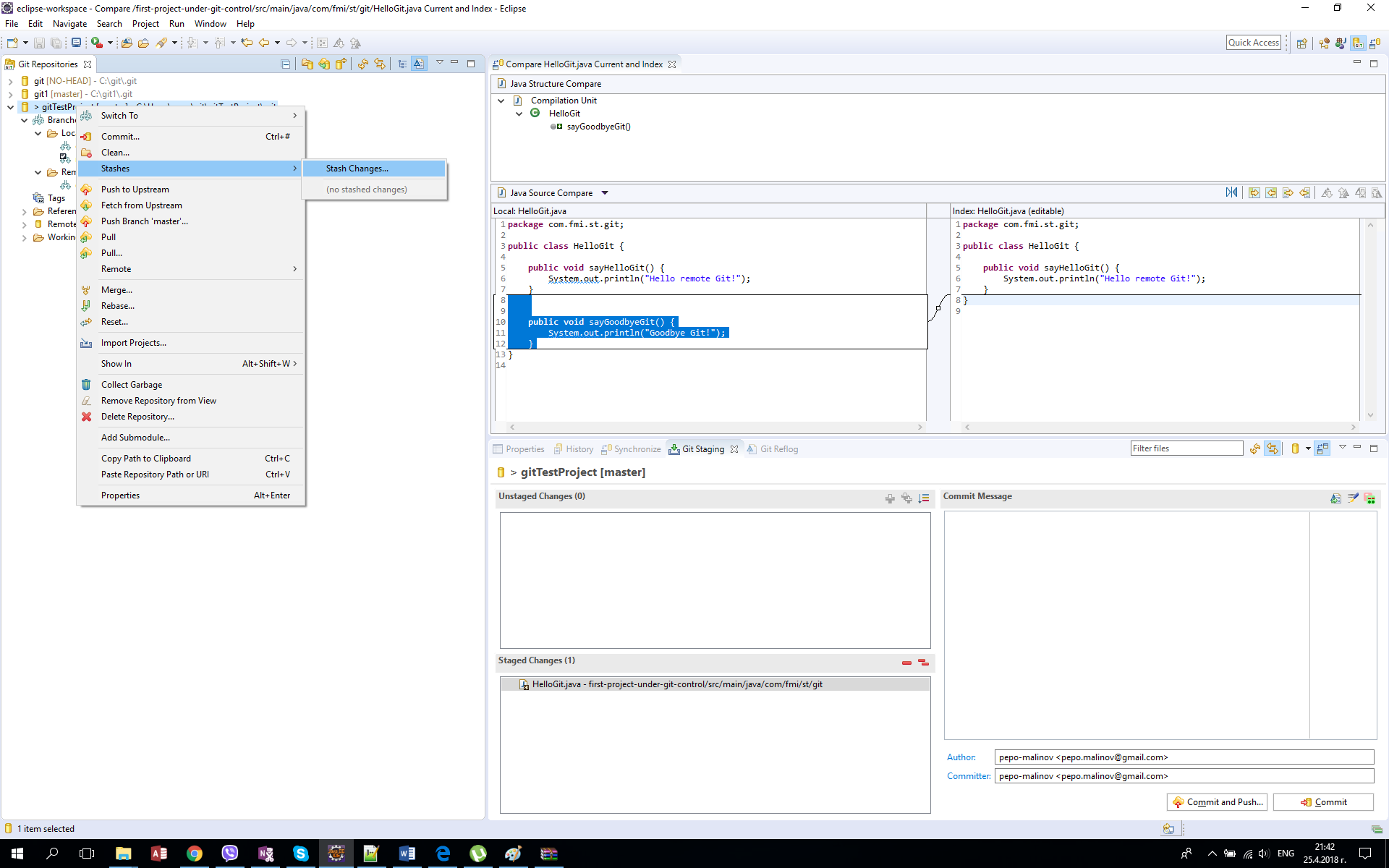


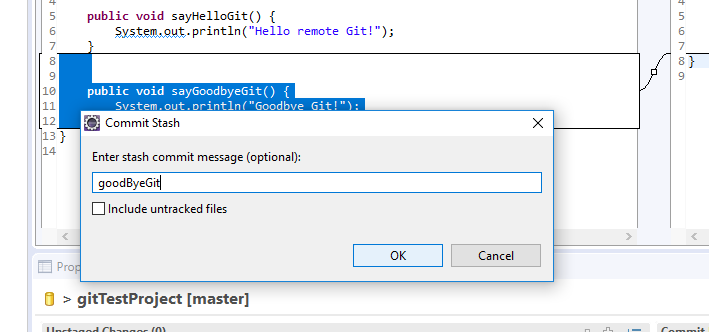
Можем да се закачим за определена точка от историята, като просто използваме бутона „Check Out“.



# 11 Използване „Stashes“

Когато имаме дадени промени има ситуации в които не желаем да ги камитваме, но желаем да ги запазим. Тогава може да ползваме “Stashes”. Аналогично на камита, но с натискане на „Stashes=>Stash Changes…“. В диалоговия прозорец добавяме пояснение. Отметката „Include untracked files“ е много важна. Ако не е отметната ще загубим промените във файловете, които се намират в прозореца „Unstaged Changes“.





За да приложим съхранените промени просто използваме бутона „Apply Stashed Changes“ върху предварително селектиран „stash“

