|  |  |
| --- | --- |
| it_New | CMJ |
| **Ифраструктура** |
| версия документа 1.0  редакция № от 06.05.2013 |

Настройка рабочего места разработчика СМJ

История изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 06.05.2013 | 1 |  |  |
|  |  |  |  |
|  |  |  |  |

Содержание

[1. Git 4](#_Toc355635053)

[1.1 GIT-клиенты 4](#_Toc355635054)

[1.1.1 Eclipse 4](#_Toc355635055)

[1.1.2 GIT-Extensions 4](#_Toc355635056)

[1.2 Аутентификация в GIT по SSH-сертификату 6](#_Toc355635057)

[1.3 Извлечение проекта из Git-репозитория 8](#_Toc355635058)

[1.4 Создание новых проектов на основном Git-сервере 9](#_Toc355635059)

[2. Nexus 9](#_Toc355635060)

[3. Maven 10](#_Toc355635061)

[4. STS 10](#_Toc355635062)

[4.1 Импорт проекта CMJ из Git в STS 11](#_Toc355635063)

[4.2 Настройка стиля 22](#_Toc355635064)

[4.2.1 Стиль Java-кода 23](#_Toc355635065)

[4.2.2 Настройка стиля XML 24](#_Toc355635066)

1. Git
   1. GIT-клиенты

Git-клиентов существует много, но обязателен для установки “GIT-Extensions”, т.к. вместе с ним ставится и прочие необходимые для комфортной работы утилиты. Дальнейшие инструкции подразумевают его наличие.

* + 1. Eclipse

В STS начиная с версии 2.7 предустановлена интеграция с Git по средством плагина EGit и клиента Git на Java – Jgit. На всякий случай update-сайт EGit:

[**http://download.eclipse.org/egit/updates**](http://download.eclipse.org/egit/updates)

Туториал: <http://www.vogella.de/articles/EGit/article.html> (инструкция по установке п.2 не актульна)

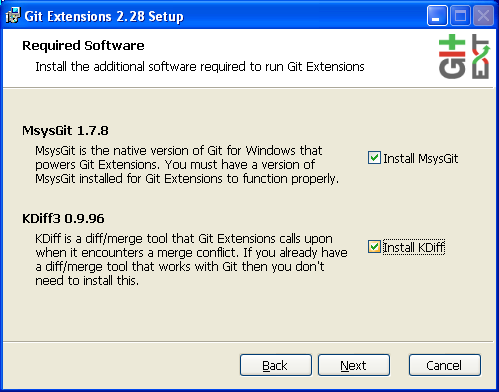
Для работы с GIT из Eclipse ничего другого не надо, но, как правило, функционала плагина недостаточно. Поэтому ставим дополнительно ещё GIT-Extensions. Все дальнейшие инструкции рассчитаны на работу с GIT-Extensions.

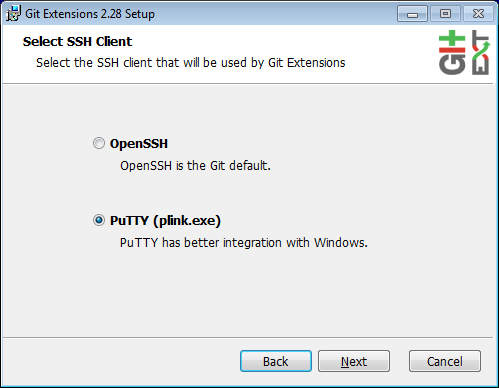
* + 1. GIT-Extensions

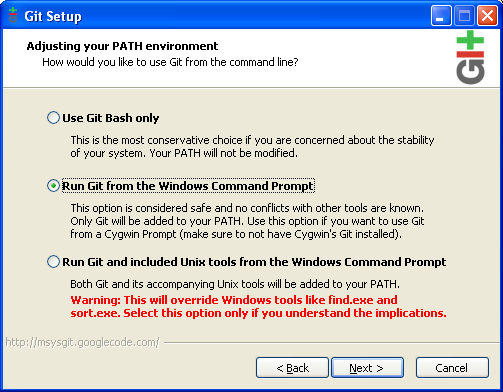
Дистрибутив <https://code.google.com/p/gitextensions/downloads/list>

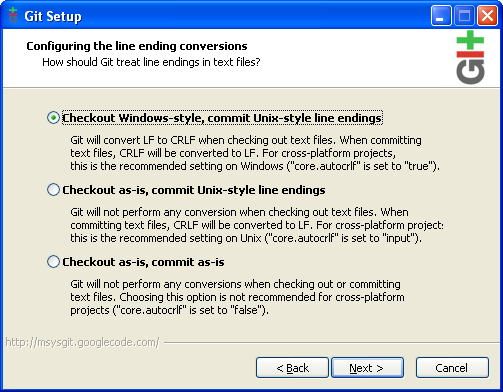
ставит также и «консольный GIT» (см. выше).

**Примечание.** Инсталлятор Git-Extensions требует предустановленного “.Net Framework”









* 1. Аутентификация в GIT по SSH-сертификату

1. От администратора получаем приватный SSH-ключ в виде файла.
2. Конвертируем полученный приватный ключ в формат putty (.ppk).

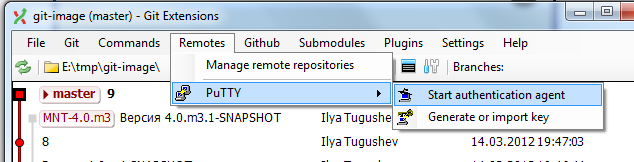
* Запускаем %GIT\_HOME%\PuTTY\puttygen.exe
* Выбираем меню Conversions/Import key



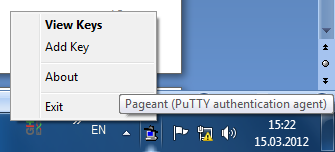
* Выбираем файл с приватным ключом (id\_rsa).
* Жмём кнопку «Save private key».
* Указываем файл, где будет сохранён приватный ключ в формате putty (.ppk).

1. Добавляем приватный ключ в формате putty в хранилище ключей pageant. Для чего:

* Запускаем Git-Extensions.
* Выбираем меню Remotes/PuTTY/Start authentication agent



* В трее правый клик на Pageant.
* Меню Add Key



* Выбираем ppk-файл с приватным ключом putty

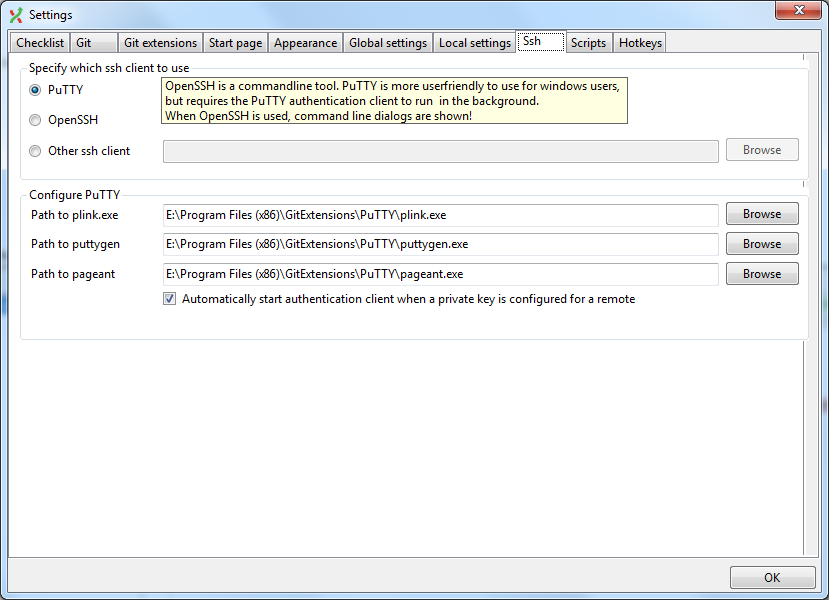
1. Проверяем соединение с сервером, где расположен репозиторий.

* %GIT\_EXTENSIONS%\PuTTY\plink.exe -l <*login*> <*host*>
* Сервер должен ответить без запроса пароля

**ВАЖНО!** До подключения к репозиторию при помощи GitExtension или Eclipse, необходимо добавить сервер с репозиторием в список доверенных хостов протокола ssh (в файл known\_hosts). Для этого необходимо первое обращение к серверу выполнять из шелла/консоли и подтвердить добавление хоста. Plink.exe выполнеяет такое обращение к хосту.

1. Переключаемся на использование PuTTY вместо OpenSSH

* В Git-Extrensions меню Settings/Settings. Закладка SSH.
* Переключаем радиокнопку на PuTTY.
* Проверяем, что пути указывают на PuTTY, установленный вместе с Git-Extensions.



* 1. Извлечение проекта из Git-репозитория

1. Установка доверия SSH-серверу с удалённым GIT-репозиторием

* В командной строке:

plink <логин>@169.254.0.208

* Подтвердить добавление сертификата вводом «y».

1. Клонирование удаленного GIT-репозитория

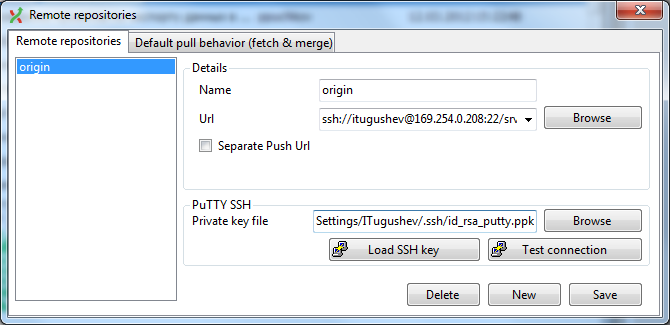
* В командной строке:

git clone ssh://<логин>@169.254.0.208/srv/repos/git/CMJ <локальный путь>

*<локальный путь>* – путь для размещения локального клона репозитория. Рекомендуется чтобы локальный путь был поддиректорией рабочей области (workspace) STS (Eclipse).

1. Проверяем или указываем, что для доступа к нашему удалённому GIT- репозиторию. используется приватный ключ.

* В Git-Extensions меню Remotes\Manage remote repositories
* В поле «PuTTY SSH/Private key file» указываем путь к ppk-файлу приватного ключа.



* Не забываем нажать Save
  1. Создание новых проектов на основном Git-сервере

На случай если Бараонва Олега нет, инструкция по созданию проекта на Git-сервере.

Чтобы выполнить нижеследующие команды, необходимо войти на сервер по ssh под своей учетной записью.

**Создать проект**

sudo -s

cd /srv/repos/git

git init --bare имя проекта

cd имя проекта

git config --bool core.bare true

touch git-daemon-export-ok

git --bare update-server-info

cd ..

chmod -R 775 имя проекта

chown -R root:repogit имя проекта

**Для добавления в вэб интерфейс**

cd public

ln -s /srv/repos/git/имя проекта/ имя проекта.git

1. Nexus

Хранилище бинарников (jar, war). А именно внешних jar-модулей и скомпилированных jar и war наших.

<http://sup.inttrust.ru:8081/nexus>

Деплой сюда выполняется maven'ом с рабочего места билд-менеджера.

Считываение – посредством maven, либо через web-интерфейс.

1. Maven

Инструмент управления сборкой проекта.

Maven поставляется вместе с STS.

Для выкачивания maven'ом из внутреннего репозитория бинарников компании Nexus, в pom.xml проекта уже должно быть (просто проверьте):

<repositories>

<!-- InterTrust's repositories -->

<repository>

<id>intertrust.repo</id>

<name>InterTrust maven repository</name>

<url>http://sup.inttrust.ru:8081/nexus/content/groups/public</url>

<releases><enabled>true</enabled></releases>

<snapshots><enabled>true</enabled></snapshots>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<id>intertrust.repo</id>

<url>http://sup.inttrust.ru:8081/nexus/content/groups/public</url>

<releases><enabled>true</enabled></releases>

<snapshots><enabled>true</enabled></snapshots>

</pluginRepository>

</pluginRepositories>

При первой настройке необходимо создать файл индивидуальных настроек для maven. Для чего надо скопировать %MAVEN\_HOME%\conf\settings.xml в папку %USERPROFILE%\.m2\ (USERPROFILE - это путь к домашней папке пользователя)

В %USERPROFILE%\.m2\settings.xml вписать логины и пароли доступа к репозиториям. Для всех разработчиков они одинаковы:

<servers>

…

<server>

<id>intertrust.repo</id>

<username>%login%</username>

<password>%pass%</password>

</server>

</servers>

…

<mirrors>

…

<mirror>

<id>nexus</id>

<name>InterTrust Proxy Cache</name>

<url>http://sup.inttrust.ru:8081/nexus/content/groups/public</url>

<mirrorOf>\*</mirrorOf>

</mirror>

</mirrors>

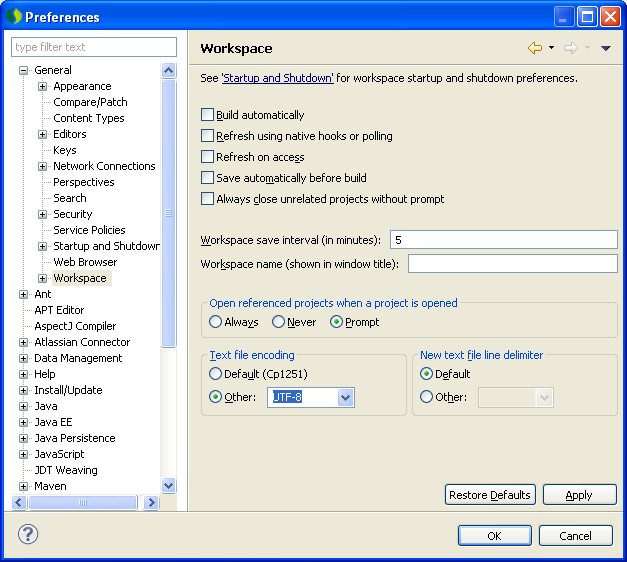
1. STS

STS – SpringSource Tools Sute. IDE на базе Eclipse с плагинами для разработки с использованием фреймворка SpringFramework.

[\\FREENAS\distrib\FreeSoftware\java\SpringSource Tool Suite\](file:///\\FREENAS\distrib\FreeSoftware\java\SpringSource%20Tool%20Suite\)

Если при инсталляции STS не был установлен JDK или надо его заменить, путь к JDK указывается в sts.ini.

После установки изменить дефолтную кодировку на UTF-8



* 1. Импорт проекта CMJ из Git в STS

В состав STS входит и Maven. Перед импортом проекта, необходимо выполнить настройку maven. См. главу 3.

Надо импортировать проект таким образом, что бы

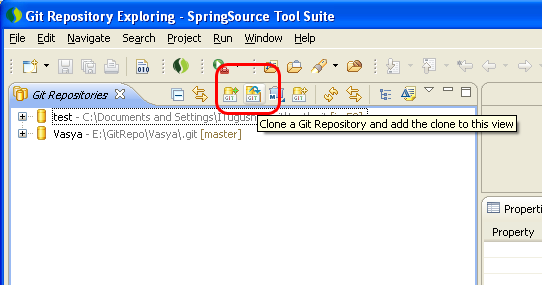
* Eclipse-проект был типа «Maven»
* для Eclipse-проекта работал Git-plugin (через меню Team)
* для Eclipse-проекта работал AspectJ-plugin
* поместить проект в желаемую папку файловой системы, а не автосгенерированную

Чтобы добиться этого в STS, надо сделать двойной импорт. Сначала «Import->Maven/Existing Maven Projects», что сделает у проектов правильный .project и .settings, затем удалить импортированные Eclipse-проекты без удаления в файловой системе. Затем второй импорт как «Project from Git», что активирует плагин по работе с GIT.

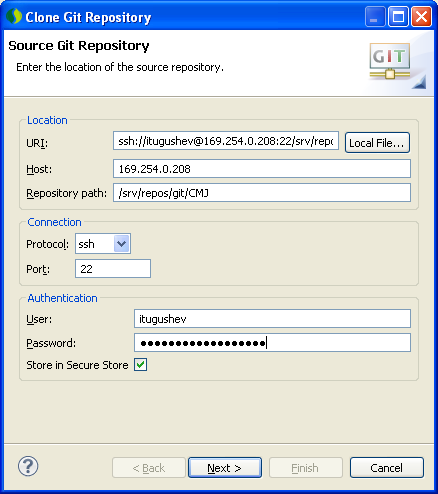
Пошаговая инструкция:

1. Если локальный клон Git-репозитория ещё не создан, это можно сделать и в STS. Переключаемся в перспективу “Git Repository exploring” нажимаем кнопку “Clone a Git Repository”.

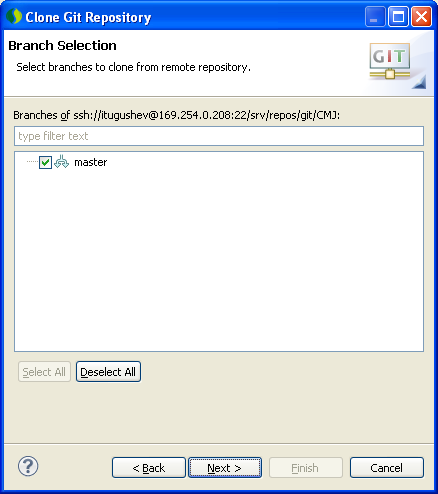
**Примечание**. Если локальный клон уже есть, нажимаем другую конопку – “Add an existing local repository”.



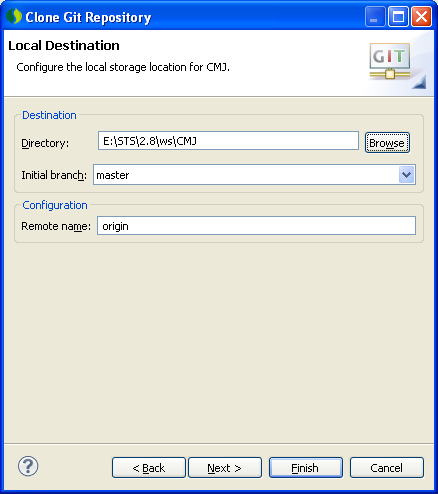
1. Указываем URL репозитория ssh://[169.254.0.208/srv/repos/git/CMJ](mailto:пользователя@169.254.0.208/srv/repos/git/CMJ)



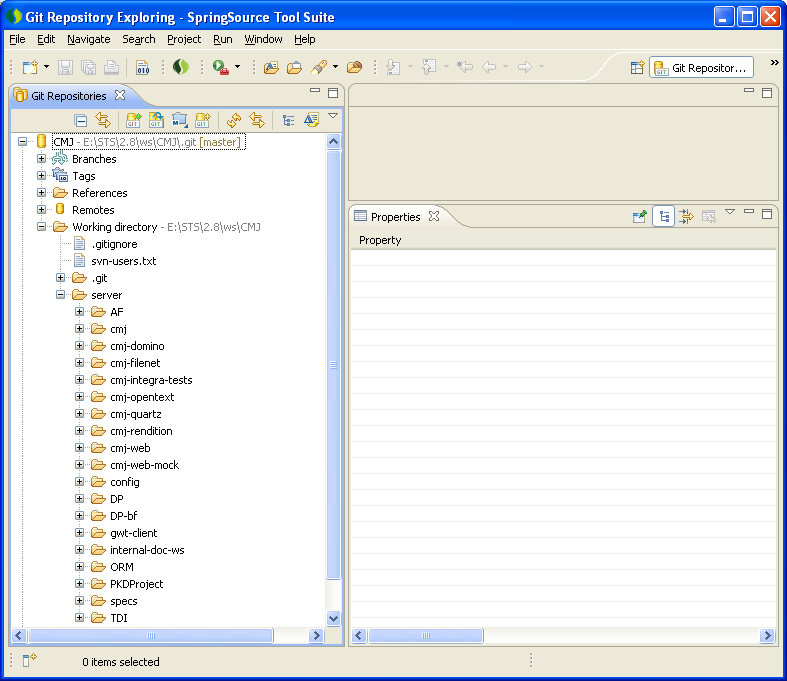
1. Выбираем все имеющиеся ветки кода



1. Указываем желаемую папку для размещения репозитория в поле Directory. Значения master и origin лучше оставить.

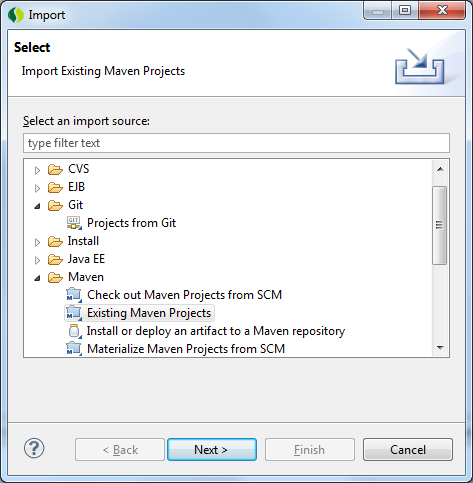


1. Жмём Finish. Получили продключенный GIT-репозиторий в Eclipse:

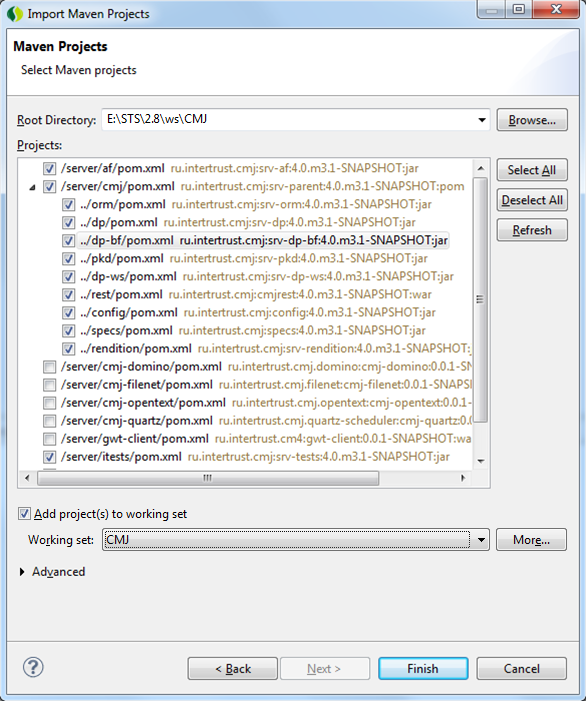


1. Импортируем из репозитория Maven-проекты. Выбираем меню File/Import… Далее Maven/Existing Maven Projects

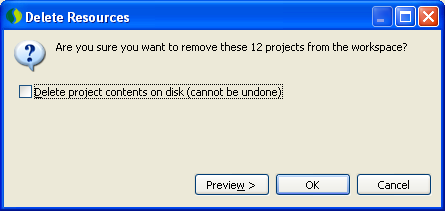
**Примечание**. Перед импортом, в maven должно быть настроено подключение к Nexus (см. п. 3)



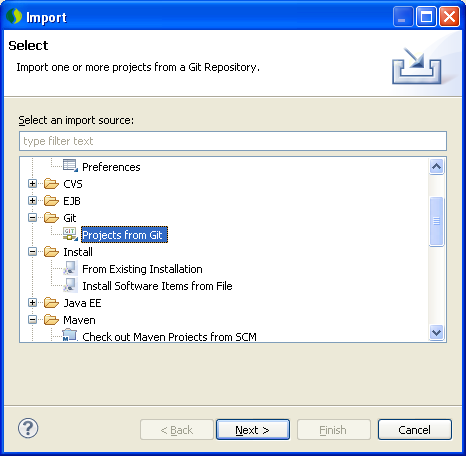
1. Указать путь к клонированному на локал GIT-репозиторию. Выбрать интересующие maven-модули. Рекомендуется отнести проекты в Working Set, выбрав из списка, либо создав новый, нажав «More».



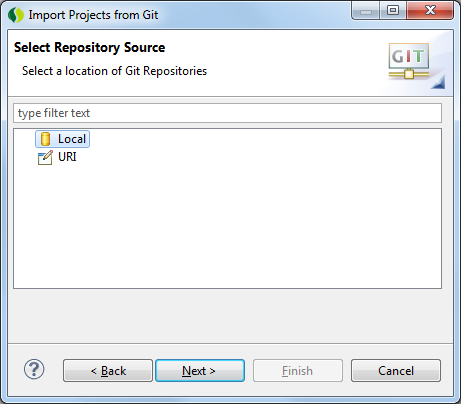
1. Переключаемся в перспективу «Spring». Дожидаемся обновления Maven-конфигурации.
2. Если у вас версия STS 3.0 или старше, сразу переходим к п.17
3. Теперь удаляем импортированные проекты **без** удаления в файловой системе. Для этого выделяем проекты и жмём на клавиатуре кнопку “Delete”. Подтверждаем удаление.



1. Импортируем проекты обратно, но уже через «Projects from Git». Выбираем меню “File/Import” и “Git/Projects from Git”

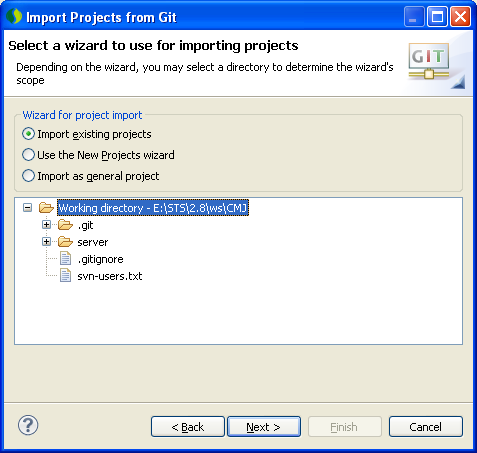


1. Указываем ранее добавленный GIT-репозитарий

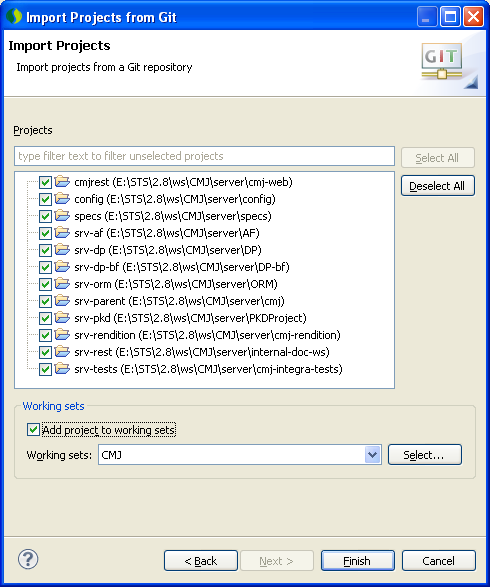




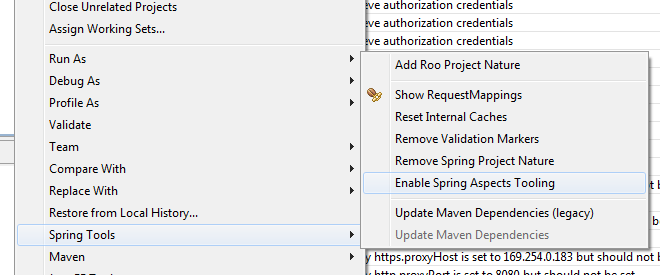
1. Жмём “Next” и выбираем импорт уже имеющихся проектов



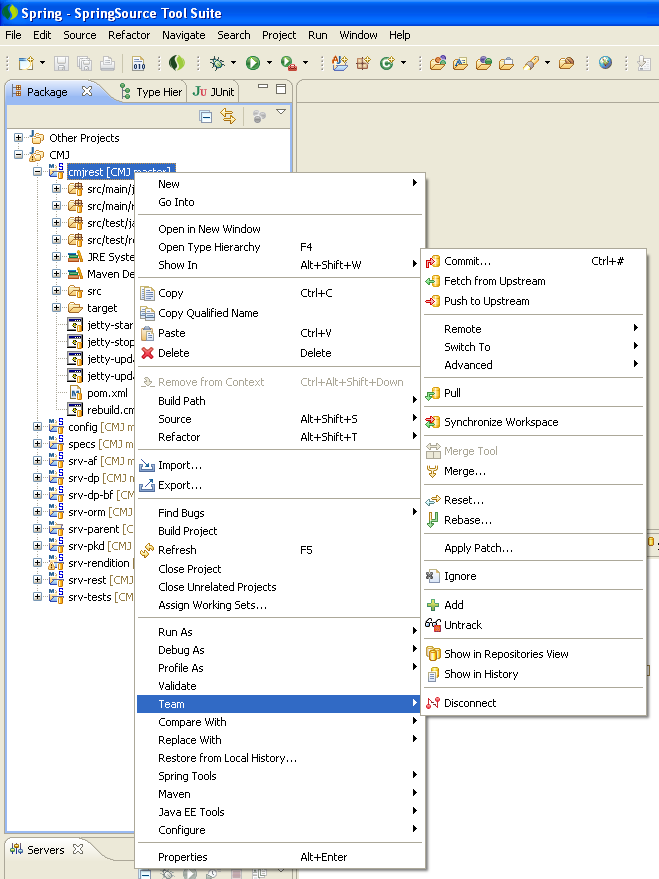
1. Выбираем интересующие eclipse-проекты.



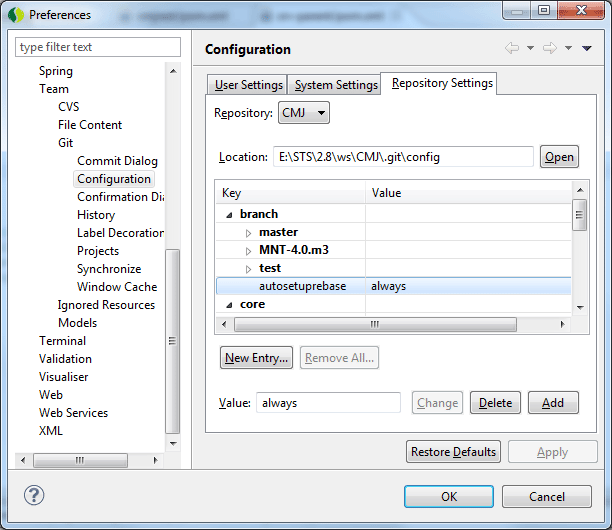
1. Далее надо включить AspectJ. В виде Package explorer выбираем все импортированные eclipse-проекты CMJ, кроме srv-parent и config. Правый клик на проекте и выбрать "Spring Tools\Enable Spring Aspect Tooling"



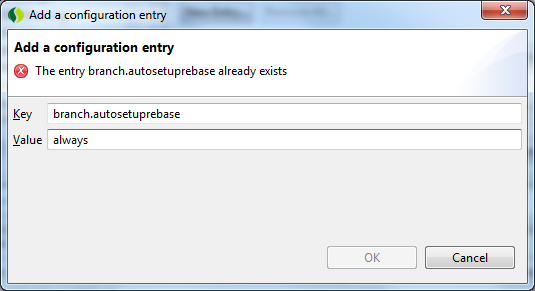
1. Меню «Team» и «Maven» должны работать.



1. Установить стратегию pull'а в rebase
   1. Либо дать команду в директории локального репозитория  
       git config branch.autosetuprebase always
   2. Либо выбрать меню Window/Preferences далее Team/Git/Configuration, закладка Repository Settings,



нажать кнопку “New Entry…”, ввести ключ branch.autosetuprebase и значение always



* 1. Настройка стиля

Настройка одинакового стиля у всех разработчиков очень важна. Разный стиль приводит к множеству фиктивных изменений кода. Фиктивные изменения осложняют идентификацию существенных изменений при сопровождении кода, осложняют разрешение конфликтов изменения одного файла и объединение веток кода.

* + 1. Стиль Java-кода

1. In Eclipse, click **Window -> Preferences**.

2. Navigate to **Java -> Editor -> Save Actions**.

3. Включить опции

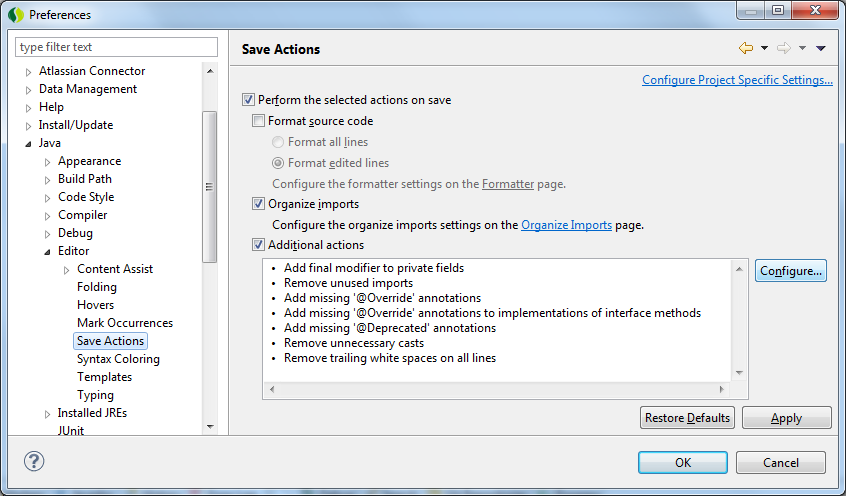
Perform the selected actions on save

Organize Imports

Additional actions

4. Нажать «Configure».

5. На закладке «Code organizing» добавить «Remove trailing white spaces on all lines» и «All lines»



1. В Ecplipse выбрать меню Windows -> Preferences

2. Перейти **Java -> Code Style -> Formatter**.

3. Нажать **New**.

4. Set the profile name to "**CMJ Conventions**".

5. From the **Initialize settings** drop-down list, select **Java Conventions [built-in]**.

6. Ensure that the checkbox for opening the edit dialog is checked.

7. Click **OK**.

8. Click **Edit**.

9. Select the **Indentation** tab.

10. Under **General Settings** -> **Tab Policy**, select **Spaces only** from the drop-down list.

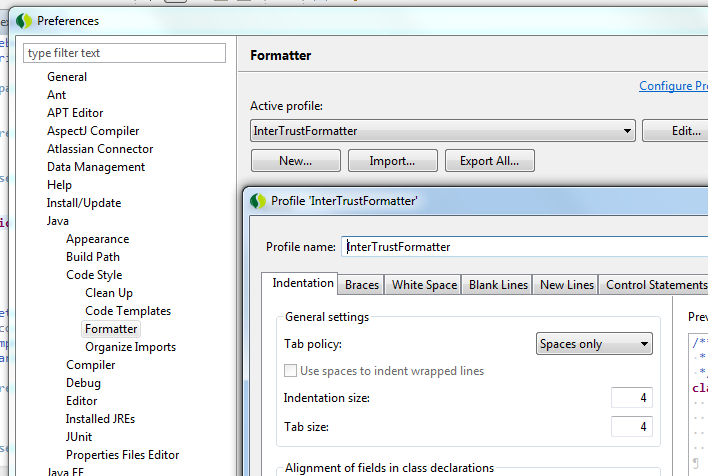
11.Set Indentation size 4, Tab size 4.

12. Select the **Line Wrapping** tab.

13. Enter a value for the maximum line width 120.

14. Click **Apply**, and rename the code profile when you are prompted.

15. Click **OK**.



* + 1. Настройка стиля XML

1. В Ecplipse выбрать меню Windows -> Preferences
2. Перейти XML -> XML Files -> Editor
3. Установить Line width в 120.
4. Проверить что установлена Indent using spaces
5. Проверить что Indentation size равен 4.
6. Снять галку «Format comments»

