

Nota de Desenvolvimento 03 (nd03) Utilizando o Pycharm com o Github

Sérgio Rivero

7 de abril de 2020

Sumário

1	Introdução	2
2	A integração do Git no PyCharm	2
3	Configurando uma Conexão Segura	2
4	Aprendendo a usar Branching e Merging	2
5	Usando o Git no PyCharm	3
5.1	Conectando ao GitHub	3
5.2	Clonando o Código da Aplicação (EcoSim)	3
5.3	Implementando código	4

1 Introdução

Olá pessoal. Aqui estamos na nossa nota de desenvolvimento 03.
Vamos discutir um pouco o uso do Git no Pycharm.

2 A integração do Git no PyCharm

O PyCharm, como a maioria das IDEs já tem uma integração com sistemas de controle de versão (VCS - version control systems). Esta integração facilita muito a vida de quem está trabalhando em equipe.

O Github é um VCS pragmático e seguro que estamos utilizando para desenvolvimento das aplicações. Então é o que usaremos integrado ao PyCharm ou a outra IDE de preferência do programador.

Neste documento utilizarei principalmente o material que está no livro do git de Chacon and Straub (2014) disponível aqui. Também iremos ver um pouco de detalhe sobre o uso do github, material que pode ser encontrado aqui. Também usaremos como referência Burke (2015)

3 Configurando uma Conexão Segura

No github você pode configurar uma conexão segura usando https ou ssh. Estes dois métodos encriptam a conexão entre você e o servidor do github evitando uma interceptação do código. É uma medida padrão de segurança de conexão.

Há instruções específicas para cada um dos principais S.O. (Windows, Linux e Mac) no próprio github:

Aqui você vê a explicação para o uso do *ssh*, que evita a necessidade de colocar sua senha, toda vez que fizer uma atualização. Você pode ter mais instruções sobre o uso do *ssh* nas páginas de ajuda do github (aqui)

E aqui você vê mais instruções específicas sobre trabalhar com o GitHub.

4 Aprendendo a usar Branching e Merging

Uma prática importante no trabalho em projetos usando o git é o estabelecimento de "ramos" (branching) na hora que você estiver alterando o código. Isso evita que eventuais problemas ou erros sejam incluídos no código principal (master) quando você estiver desenvolvendo seu componente e atualizando o repositório. O *branching* evita a incorporação precoce de código parcialmente desenvolvido no ramo principal do seu repositório, quando sua

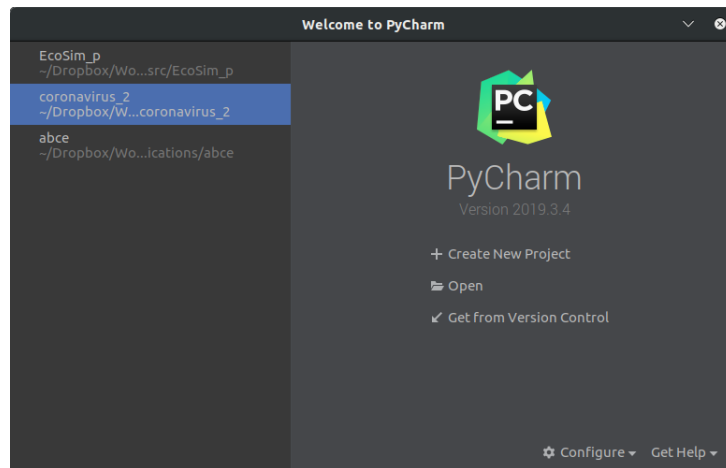


Figura 1: Tela Inicial do PyCharm

alteração do código estiver madura e estada, você pode, eventualmente, introduzi-la no código principal usando o *merging*. Para aprender mais sobre isso, você pode olhar o livro do git (aqui)

Uma instrução simples e interessante sobre isso pode ser vista aqui

5 Usando o Git no PyCharm

O PyCharm fornece um caminho para utilizar diretamente o GitHub via o componente VCS no aplicativo.

No site do PyCharm você pode ver instruções de como clonar um repositório (aqui).

O sistema de controle de versões está apresentado aqui.

A integração com o Git está aqui

5.1 Conectando ao GitHub

A conexão com o GitHub no PyCharm pode ser encontrada aqui

5.2 Clonando o Código da Aplicação (EcoSim)

Com sua conta no GitHub configurada e o PyCharm instalado, você pode clonar o diretório do EcoSim. Ao abrir o PyCharm você verá a seguinte tela (figura 1):

A opção "Get from Version Control" permite a você baixar o diretório do EcoSim_p diretamente do GitHub.

Você deverá ir para a pasta onde você quer baixar o repositório (o segundo campo na figura 2)

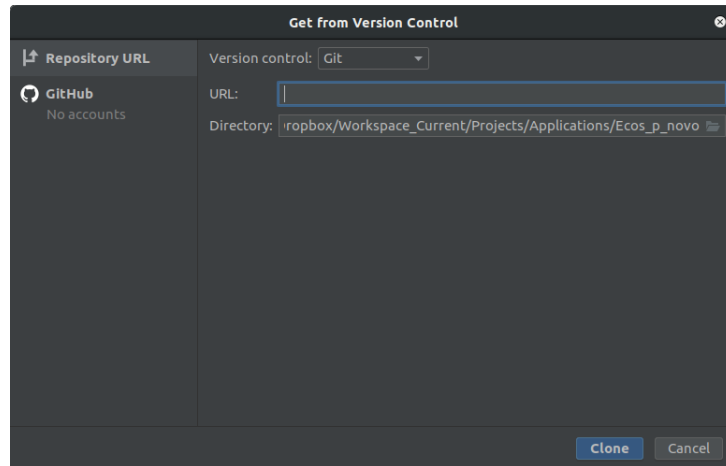


Figura 2: Clonando um Diretório

A url que você vai colocar no campo "url" é aquela do Diretório do Ecos no GitHub, quando você aperta o botão "clone or download" no site do repositório do projeto (figura 3):

Copiando o link que aparece na figura 3 e que pode ser visto aqui: Colocando este link na janela da aplicação (do PyCharm) você estará clonando o repositório da app diretamente por dentro da IDE.

Aí você coloca seu usuário e senha do GitHub ou escolhe a instalação de um token (figura 4)

Se tudo der certo ;-) você terá o repositório clonado para a pasta escolhida (figura 5). Aí é só criar o ramo e começar a codificar

5.3 Implementando código

Você agora pode começar a codificar (figura 6):

Abrindo o código no editor.

Mas não esqueça de, primeiro criar um ramo (fork). Veja aqui.

Happy Coding

Referências

Burke, C. (2015). Introduction do git.

Chacon, S. and Straub, B. (2014). *Pro git*. Apress.

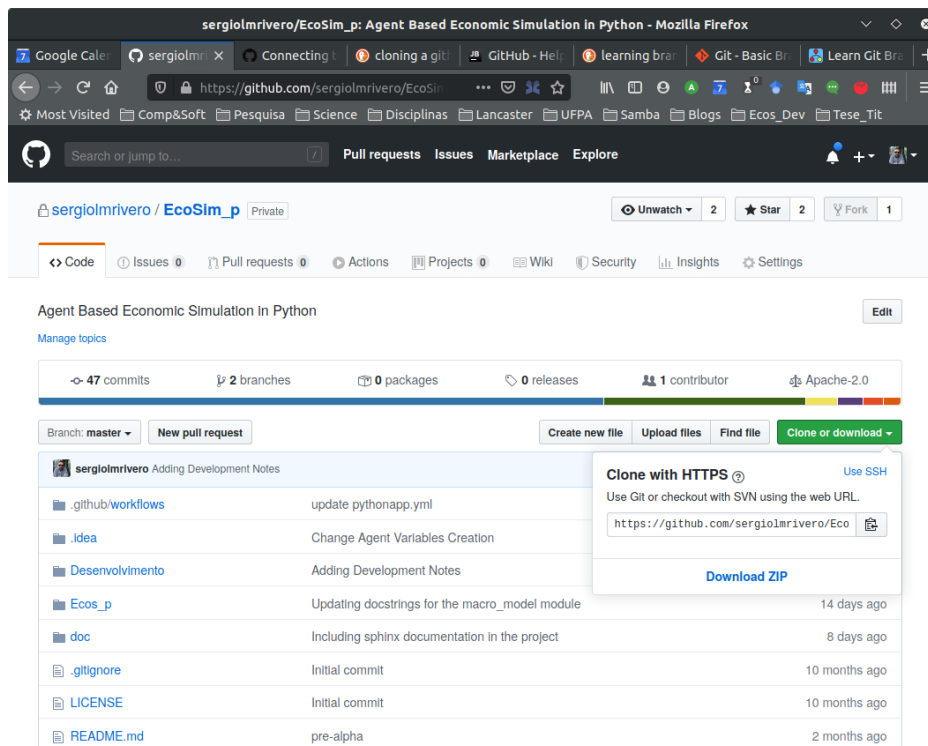


Figura 3: A url para copiar e incluir no PyCharm

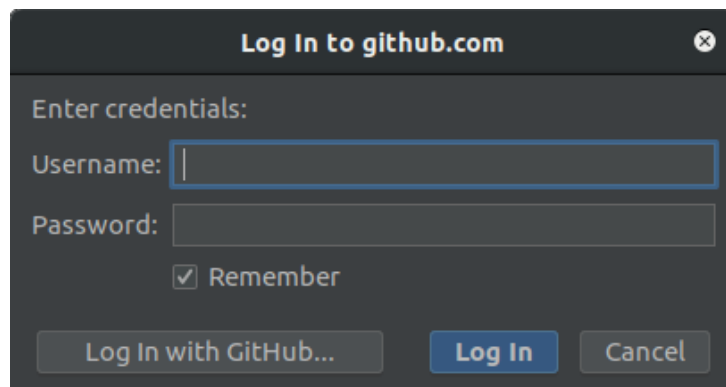


Figura 4: A janela de autenticação do github no pycharm

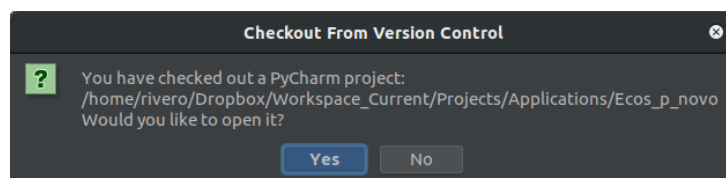


Figura 5: Janela com mensagem indicando o sucesso do check

```

1  # coding: utf-8 -*-
2
3  """
4  Definition of the class Model
5
6  This class receives a yaml file with the definition of the simulation scenario and then creates the simulation with all
7  simulation objects
8  """
9
10 import yaml
11
12 class Model:
13     """
14     Model Basic Class
15     Receives the yaml file name and read it
16     Creates all objects in the simulation
17     """
18
19     def __init__(self, yaml_file):
20         """Load the definition of the yaml file"""
21         self.seed = time.time()
22         self.random = random.Random(self.seed)
23         self.simulation = None
24         self.init_file = yaml_file
25         with open(self.init_file, 'r') as read_file:
26             self.yaml_def = yaml.load(read_file, Loader=yaml.FullLoader)
27         self.name = self.yaml_def['model']['name']
28         self.schedule_def = self.yaml_def['schedule']
29         self.create_schedule(self.schedule_def)
30         self.spaces = dict()
31         self.create_spaces()
32         self.agents = OrderedDict()
33         # self.model_observers = {}
34         self.agent_observers = {}
35
36     def create_schedule(self, schedule_def):
37         """Creates the main schedule using the yaml schedule definition"""
38         self.schedule_factory = SchedulerFactory(self)
39         self.schedule = self.schedule_factory.provided_schedule(schedule_def)
40
41     def create_spaces(self):
42         """Creates SpaceFactory (SpaceCreator) and create space objects for the simulation from the yaml definition"""
43         self.spaces_def = self.yaml_def['spaces']
44         self.spaces_factory = SpaceFactory(self, self.spaces_def)
45         self.spaces = self.spaces_factory.spaces
46

```

Figura 6: A classe Model do Kernel do Ecos_p