Developer's Guide

Checking out the project:

Install Git on your system, any version will do, and then create yourself an account on github.com following their instructions for configuring your git account and ssh keys if applicable.

Next grab the codebase with

git@github.com:NCSU-VSR/wolfscout.git

You now have the latest version of the code and all of the branches used to create the project.

Starting Development

1. First create your own branch of the code with
   git branch yourfirstnamelastnamehere
2. Now checkout this branch so that you can edit it.
   git checkout yourfirstnamelastnamehere
3. When you would like to save your work:
   git add -u .
   git commit -m "your commit message here"
4. And when you would like to share those changes with others:
   git push origin yourfirstnamelastnamehere

Directory Structure Explained:

apps

crawler

cronos

admin.py

Edit this to add new models to the admin interface of django.

models.py

Where all business logic and data models are described for the database, these are used to store the items for the CRONOS API data that we collect.

tests.py

tests for the views.

views.py

What functions are required to fetch data from CRONOS and store it in our application.

migrations

Directory used to store migrations when you have changed a model, write a migration to add the new structure to the DB.

gpscollar

fixtures
Directory for sample testdata to live
migrations
Directory used to store migrations when you have changed a model, write a migration to add the new structure to the DB.
admin.py
Edit this to add new models to the admin interface of django.
collar.py
This file contains a class that is used to parse the data from CSVs into the database for the entire world to see.
cron_fileProcessor.py
the same as fileProcessor.py but with extensions to support it running as a cronjob.
fileProcessor.py
Iterates through all known sample_data indexing the points into our database.
forms.py
used to create validation based forms for our web interface for various components found in models.py
models.py
Contains collars and collar data models for our application to store into the database, also contains business logic and validation routines.
resources.py
A bit of skeleton code to showcase how a restful API can be build around a model, in this case collars and collar data.
shape_file_builder.py
An out of date code segment that illustrates how shapefiles could be built (do not use this method, just look at how we query things.)
support.py
Early prototype to show how to find all matches(interactions), again do not use, but is nice to understand how matches can be found.
tests.py
tests for the models.py
urls.py
Locally defined urls for this application used to power the restful API demo.
views.py
All of our functions for extracting and inputting data from the web and other interfaces for collars and collar data.

general
General functionality goes here if it was not defined well in other parameters and does not require data storage.
forms.py
Generic error list for all forms is found here.
models.py
empty

tests.py
Tests for things in this app go here.
views.py
Contains the views for getting all required site content and the index(root) of the application.

study
Contains all of the information needed to setup and manage various studies from within our application.
migrations
Contains all db migrations for this app
admin.py
Edit this to add new models to the admin interface of django
cron_interactions.py
Calculates interaction data in a script that is designed to work with cronjobs.
forms.py
Used to create validation based forms for our web interface for various components found in models.py
interaction_calculator.py
The same as cron_interactions.py but without the cronjob bindings.
models.py
Contains the database descriptions for Shapes to analyze, Studies, animal-to-animal interactions, and shape-to-animal interactions, as well as all the group management for interactions.
tests.py
Tests for things in this app go here.
views.py
Functions to manipulate, view, and create interactions around studies all go here.

wildlife
migrations
Contains all db migrations for this app
admin.py
Edit this to add new models to the admin interface of django
forms.py
Used to create validation based forms for our web interface for various components found in models.py
models.py
Used to describe species and Animals here as well as their attributes.
tests.py
Tests for things in this app go here.
views.py
Used to generate CSVs and provide additional filtering functionality to the web application.

compass
Contains all the sass files needed to compile the stylesheets.

media

All static content you do not wish to track in version control goes here.

sample_data

All of the known data points as of the time that we originally started this project. These files serve as a great bed for test data.

serverConfig

gunicorn.conf - Configuration script used to setup the gunincorn web server
gunicorn.sh - The shell script which manages gunicorn on Ubuntu
nginx.conf - Configuration file for the nginx web server
wolfscout - Site configuration file of the wolfscout application for nginx

services

Any routine service that you would like to deploy to non Wolfscout boxes.

remoteCollar

All the code need to transmit new collar information to our server.

collarFileTransmitter.py

Uses requests library to send an http request to our server. The request contains the file that was most recently changed

collarFileWatcher.py

Is to be started upon login. It uses a statically defined variable in this script to determine what directory to monitor. Once launched it will look for changes in the directory. As changes occur it will start a new thread which uses collarFileTrasnmitter to send the files to The master server, which is also statically defined in this file.

settings

common.py

Core settings file for wolfscout

sample.py

File that provides a blueprint for building your own basic settings file with your private db credentials.

testSettings.py

Settings file used for working with our test cases and Jenkins the CI server.

shapes

An external library installed in our application to provide support for generating shapefiles based on django-querysets

static

All compiled and versioned static data goes here (CSS, JS, etc)

templates

All html files for templating

tests

All additional tests

.**gitignore**

A config file which manages what goes in and out of version control.

fabfile.py

A collection of scripts to automate routine development tasks

hudson.txt

The configuration script used to launch our application in Hudson/Jenkins CI

requirements.txt

All required python packages and their versions(if needed).

urls.py

A file that uses regular expressions to provide a mapping from a URL to a view or function in our application.