Instructions

Overview

Introduction:

During the course so far we have developed a number of database backed web servers using Django. For this assignment you are tasked with developing a RESTful web service using the knowledge you have gained so far. The server should use Django Models to appropriately design and model the input data and provide usable migrations for the underlying database. As per the examples in the lesson you will be working with a small dataset of biological data, the CSV files of the data and a document describing the data is provided so you can design an appropriate model.

To successfully implement the application you should have a good understanding of the flow of data through a Django application. You should make appropriate use of Serilaizers, forms and validation if appropriate. You should ensure you make appropriate use of Django Views. The application should be laid out in a manner that is clear and easy to follow.

Task:

You are working with a group of bioscience researchers who are working on protein domains. They need a system that they can regularly query for the data they've generated. It has been decided that a RESTful web service will be implemented and you have been asked to build that service.

Data background:

The datasets included in this tasks are the dataset that the researchers have generated. A little background is worth considering in order to contextualise the data and understand how to model it.

Every living organism is a cellular organism. This means that organisms are made of one or more cells. Each cell contains DNA and proteins. Each protein has a biochemical function. Proteins performing their functions in concert with one another are how cells and organisms perform all the tasks they need to be alive. A protein is a linear sequence of amino acids joined together like beads in a necklace. Proteins are typically between 10 and 200 amino acids in length but can be as large as 40,000 amino acids long. There are 20 amino acids and these can be arranged in any order. Proteins also have sub-regions of sequence that they share with other proteins which are called domains. Domains can be identified within a protein by naming its position in terms of linear coordinates. The researchers you are working with have provided you with a file of protein sequences and domain.

Region annotations:
Organisms are classified using a system of Taxonomy. Each organism is assigned a Genus and Species name and the NCBI Taxonomy Database assigns a single integer number as an ID value for each species.

Resources:
[Cell (biology) Wikipedia]
[NCBI Taxonomy]

To summarise this information:
Organisms *HAVE MANY* Proteins
Organisms *HAVE A* Genus name
Organisms *HAVE A* Species name
Proteins *HAVE MANY* domains
Proteins *HAVE ONE* sequence
Domains *HAVE ONE* pfam domain ID

Use this information to guide you on what tables you may need and what the primary_key - foreign_key relationships may be. Use the dataset information (refer to last page of this document) to decide what data goes in to which table.

Task:
The researchers have given you four files. The first a file of protein animo acid sequences. Each protein has an ID and an animo acid sequence. The animo acid sequence is represented in a convenient 20-letter code. The second file is a series of domain annotations they have created. The third file is data about the domains. They would like you to build a RESTful web application they can query to get data out of it. It should host the data in the three provided files and return json data using the specification outlined below. The final file contains the REST specification, this describes five endpoints and gives examples of what data should be returned when a client makes a request to that end point.

For this application ensure you use the default SQLite3 database. **DO NOT** use postgres

Deliverables:

1. A zip file of the Django application that implements the REST API as described in the accompanying REST specification
2. The instance of the Django application in the zip file should have all the data loaded
3. The zip file should contain a python script or method for loading the provided CSV data in to the database. You must include a pip requirements.txt file to ensure that your application libraries can be loaded
4. A short report (2500-3000 words) explaining the code in your application and how it meets the requirements. You may use focused, short code extracts if they make your explanation clearer. Explain the logic of your approach, why is your code arranged as it is? This report should also include how to run the unit tests and the location of the data loading script.

Review Criteria

The application will be graded on whether it is technically correct and implements the API as requested. Code should be clear and easy to follow. The application should be well organised, for instance it should make correct use of models, API, view and serilaiser files. A good application will include a suite of tests that ensure that application correctly implements the API that is described.

Requirements

We will assess your work based on the following requirements and criteria:

R1: The application contains the basic functionality described in class

a) correct use of models and migrations

b)correct use of form, validators and serialisation

c) correct use of django-rest-framework

d) correct use of URL routing

e) appropriate use of unit testing

R2: Implements an appropriate data model for the data

R3: Implementation of appropriate code for the required REST endpoints

R4: Implementation of appropriate method of bulk loading data

Code style and technique

Your code should be written according to the following style and technique guidelines:

C1: Code is clearly organised into appropriate files (i.e. view code is placed in an appropriate view.py or api.py file, models are placed in an appropriate models.py file)

C2: Appropriate comments are included to ensure the code is clear and readable

C3: Code is laid out clearly with consistent indenting, ideally following python pep8 standard

C4: Code is organised into appropriate functions with clear, limited purpose

C5: Functions, classes and variables have meaningful names, with a consistent naming style

C6: Appropriate Unit tests to cover the REST API functionality are provided

Submission

You should write a brief report and submit your source code. The submission should contain the following items and information:

D1: Django code in standard ZIP format

D2: Short report (2500-3000 words) in PDF format. Including how to unpackage and run your application and how to run the tests for your application. Explain where it can be found in the code. The report should explain the code in your application and how it meets the requirements. You should explain the logic of your approach, why is your code arranged as it is. This report should also include how to run the unit tests and the location of the data loading script.

The detailed marking rubric for this coursework is in **marking_criteria.xlsx** (separate file)

Dataset information:

The following are the files needed for the coursework assignment

1. Input Data Files:
   data_sequences.csv

   A CSV file of protein sequences. Each record consists of two columns. The first column is the Protein ID and the second column is the protein sequence in single letter code. The longest sequence is just less than 40,000 characters long. Not all proteins in the data set have sequences

   protein ID
   Protein Sequence

2. data_set.csv
   A CSV file of protein domain assignments. Each row is the data for a Protein Domain assigned to a different Protein. There are 10,000 records in this file.

   The columns are: Protein ID,
   Organism TAXA ID,
   Organism Clade Idenitifer
   Organism Scientific name ("Genus Species")
   Domain description
   Domain ID
   Domain Start Coordinate
   Domain End Coordinate
   Length of Protein

3. pfam_descriptions.csv
   A CSV file.
   There are two columns:
   Domain/Pfam IDs
   Domain/Pfam Family description

4. The API specification
   The file gives the URI for each GET and POST, each GET includes a full example (i.e http://127.0.0.1:8000/api/protein/A0A016S8J7) followed by the JSON that should be returned by a working version of the application