FHIRBALL

Usage

Demo Server

run:

```
$ export FLASK_APP=flask_app.py
$ export FLASK_DEBUG=1
$ flask run
```

Writing Maps

Create models in models.py and add a *_map_* method. All fields in the corresponding *__table__* are available as `self.field_name`.

Using these fields and possibly other models, `_map_` must build and return a Fhir Resource instance.

Package Description

db.backends -- Database specific stuff

SQLALchemy - ***SqlAlchemy backend***

- `abstractbasemodel.AbstractBaseModel`

  A subclass of SQLAlchemy's declarative_base.

  Adds additional fhir related functionality to all models. Most importantly, it provides the `.to_fhir()` method that handles the transformation from an SQLAlchemy model to a Fhir resource. User-defined models subclassing this class **must** implement the `_map_()` method that has access to all of the related table's columns and must return a Fhir Resource instance.

  `to_fhir(query=None)`

  Converts a model to a Fhir Resource. Accepts a `query` parameter of type `server.FhirRequestQuery` that may alter functionality based on the request.

  `ContainableResource(cls, id, name, force_display=False)`

  A shortcut for defining external resources that may or may not be included based on the request. It will produce a Reference containing either an endpoint link to the resource or an internal link to the contained data.

  cls: The class of the model we are referring to (eg Patient)

  id: the system id of the resource

  name: the name of the field this reference occupies in the parent's Resources

  force_display: If left to False, resources that are not contained will not include the *display* property since it requires an extra query.

  returns: A dict representing a reference object

- `fhirbasemodel.FhirBaseModel`

  Another abstract base class iheriting AbstractBaseModel.

Implements fhir functionality like querying, searching, etc

@classmethod `get(cls, query)`

Handle get requests. Uses the information contained in *query* to determine how many and which resources should be returned. Pagination happens here.

cls: The class of the resource that gas been requested

query: An instance of `server.FhirRequestQuery` representing the current query

returns: A Json dict containing the response. The responce may be a single Resource or a Bundle

Fhir -- Fhir resource models

Auto-generated classes for Resource models.

These classes handle (de-)serialization and validation and they are the building blocks for models' `_map_` method. Many additions have been made to make it as easy as possible to create Resource objects. See Writing maps for more.

Warning: Do not edit any of the files in the Fhir/Resources folder. They will be overwritten at the next generation. See Fhir.base for details.

- `Fhir.resources` **<-- Use this to import stuff!**

  An empty module that is dynamically populated by Fhir/__*init*__.py that allows easier imports of Resources like:

  ```
  >>> from fhir import resources
  >>> p = resources.Patient()
  >>> from fhir.resources import Patient
  ```

- `Fhir.Resources.extensions` **<-- Write here to extend stuff**

  This module is imported by Fhir/__*init*__.py after the root Resources folder so classes defined here will overwrite the generated ones with the same name.

  Contains shortcut wrapper classes like `AMKA` and `HumanName`

- `Fhir.base` **<-- This is where the actual magic happens**

  Contains all resources deeded for Resource generation.

  `fhirabstractbase` and `fhirabstractresource` contain the two abstract classes that all Resources inherit. This is where the actual functionality is implemented.

server -- Server related

- `FhirRequestQuery`

  A class that holds information contained in the request querystring

  Has the followng properties:

  `resource`: The name of the requested Resource

  `resourceId`: The id following the reource if any

  `operation`: $operation string

  `modifiers`: dict of key, value pairs for all _reserved parameters

  `search_params`: dict of key, value pairs for all non _reserved parameters