REST ORM

Rory Tulk - Mohammad Jalali

University of Toronto

REST

REST (Representational State Transfer)

- REST is a software structure mainly used to produce machine readable contents using the natural way the Internet works
  - HTTP protocol
  - Hypermedia formats
- Concepts:
  - Addressability – resources uniquely identified by URI
  - Statelessness – resource is the same, regardless of the chain of navigation to get to it
  - Connectedness – every resource should be linked to by another resource
  - Uniform interface – same set of methods to operate on all resources
- Data is represented as resources
- Resources are addressed with a URI
- Many MIME types such as XML, JSON and YAML are supported
- http://www.bla.com/users/johnsmith

ORM

ORM (Object Relational Mapping)

- Store and load data from an RDBMS into an Object Oriented Data Model
- Application programmer no longer needs to solve the Object-Relational Impedance Mismatch
- Maps between Classes and Tables as defined by the application and database programmer, either programmatically or through the use of configuration files

Goals

Ideal Goals

- Automatically define REST API and relational database tables from data models, creating “persistent web objects” in a single click/operation
- Client layer which exposes web API as a set of shared objects – the same set that make up the data model on the server

Realistic Goals

- Define REST APIs in the same way as ORM tables, with as little effort as possible, leveraging similarities/redundancies wherever they exist

Improving Django's REST Interface

Prestudy

Investigated different technologies and tools for ORMs such as Hibernate, Django ORM and SQLAlchemy and their main features

Django

Django is a web applications platform, written entirely in Python, which enables easy creation of data-driven web applications, thanks to its intuitive programming model and strong ORM integration

Existing methods for creating REST web services with Django left several features to be desired, and none were integrated with the Object Relation Mapping interface. Using an existing REST solution (Django Rest API – URL), we implemented several features to improve the RESTfulness of the Django applications:

Client Friendly Serialization

Created new serializers and plugged them into the Django's Serialization Framework to better reflect the needs of RESTful Web Service client applications.

URL Links for Object Relationships

Implemented a mechanism for determining the URL of a REST resource from it's model name and primary key. Using this, serialized representations of Django REST resources contain proper hyperlinks to their related items, not simply primary key values.

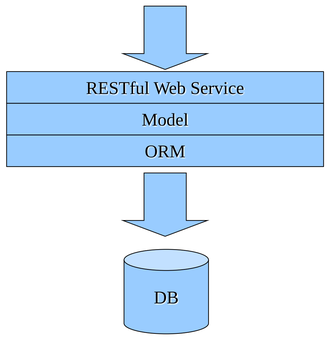
Automatic Query String Attribute Searching

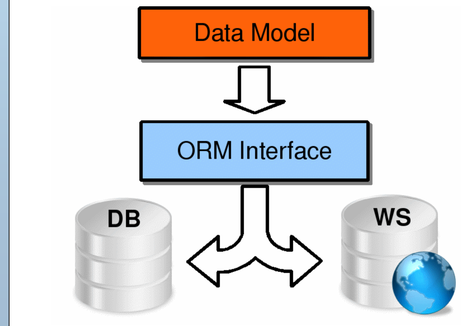
In addition to retrieving resources using URLs based on their primary key identifier, we have created a mechanism which allows clients to retrieve representations based on any fields in the model.

Delayed-GET Caching

Ability to enable Delayed-GET patterns (based on HTTP IF_MODIFIED_SINCE header) for all resources seamlessly without additional effort on the part of the application programmer.

Architecture of a Service





RESULTS

Successfully implemented selection of RESTful feature improvements in Django.

Created a proof-of-concept demonstration web service, based on the Django example in *RESTful Web Services*. This results in a remarkably similar service, albeit with improved XML resource representation, query string searching, etc, with much less effort required to produce.

Could not implement client library due to time and resource constraints.

CONCLUSIONS

There is a great opportunity to alleviate redundancy and excessive effort when creating data-driven web services. The tradeoff between control and automation in tools to create these services seems to be the major point of contention among the open source community.

The tool we developed leveraged implicit functionality and automation where possible, resulting in drastically reduced effort when creating REST services. This approach isn't for everyone, however.

BIBLIOGRAPHY

Copeland, R. (2008). Essential SQLAlchemy. OREILLY.

Django Project Website. Retrieved 2009, from http://www.djangoproject.com/

Django Rest Interface. Retrieved 2009, from Google: http://code.google.com/p/django-rest-interface/

Ruby, S., & Richardson, L. (2007). RESTful Web Services. OREILLY.

Tulk, R., & Jalali, M. (n.d.). Object Relational Mapping Investigation: Hibernate and SQLAlchemy. Retrieved from Google: http://docs.google.com/Doc?id=dg5wjm4b_1d8cxfbf5&hl=en

Red Hat Middleware, LLC. (2004). Hibernate Reference Documentation 3.1.1. url: http://www.hibernate.org/hib_docs/reference/en/html/index.html

Larson, W. (2008 , 7 23). Replacing Django's ORM with SQLAlchemy. Retrieved 01 18, 2009, from lethain.com: http://lethain.com/entry/2008/jul/23/replacing-django-s-orm-with-sqlalchemy/

Visual elements courtesy of dryicons (http://dryicons.com)