

Building a REST API with Django & Django REST Framework

Kenny Yarboro

Goal: Build REST API

- ❖ Interface for Firewall Rule Additions
- ❖ Requirements
 - ❖ Support Direct API Calls
 - ❖ Provide Self-Service Portal
- ❖ Preference for Using Python

Data to Receive as Input

- ❖ Customer
 - ❖ Text Field
- ❖ Source/Destination IP Address
 - ❖ Single IPv4 Address
- ❖ Destination Port
 - ❖ Single Integer Value
- ❖ Business Justification
 - ❖ Text Field - Optional

Selecting a Starting Point

- ❖ Django

- ❖ Had Heard of It
- ❖ Python-based
- ❖ Built-In Admin Interface
- ❖ Found REST Module

- ❖ Flask

- ❖ Seemed Complex
- ❖ Required Python Knowledge



Learning Basic Building Blocks

- ❖ Models

- ❖ Define Data Object

- ❖ Attributes

- ❖ Behaviors

- ❖ Optionally Defines Database Table

- ❖ Serializers

- ❖ Translate Models into Other Formats

- ❖ JSON

- ❖ XML

- ❖ Others

Learning Basic Building Blocks

❖ Views

- ❖ Callable Functions or Classes
- ❖ Take a HTTP Request
- ❖ Return a HTTP Response

❖ ViewSets

- ❖ Single Class for Set of Related Views
- ❖ Predefined & Build Your Own

Learning Basic Building Blocks

- ❖ URLs

- ❖ Mapping Addresses to Views & Templates

- ❖ Routers

- ❖ Automatic Way of Mapping for ViewSets

- ❖ Settings

- ❖ Configuration of Application

Data Formats Provided by Django REST Framework

- ❖ Add Data Formats Using Renderers
 - ❖ JSONRenderer
 - ❖ XMLRenderer
 - ❖ More in `rest_framework.renderers` Package
- ❖ Enable Browsable API Renderer
 - ❖ BrowsableAPIRenderer

Considerations Before Building a REST API

- ❖ 'Disabling' Certain API Views
- ❖ Adding Custom Validators & Handling Validation Errors
- ❖ Manipulating Data Flowing within the Application
- ❖ Enabling Single API Call to Accept Mult. Input Formats
- ❖ Customizing the Save Process Behind an API Post
- ❖ Documentation Considerations

'Disabling' Certain API Views

- ❖ Who Can Access Which Views?
 - ❖ Maintain User & Groups Tables
 - ❖ Auto-Add New Users or Not
 - ❖ Checking User Data from Request
 - ❖ Active Directory & Group Memberships
- ❖ How Do You 'Disable' a View?
 - ❖ Define Individual Views
 - ❖ Using ViewSets
 - ❖ Disable Based on Users
 - ❖ Return 403 Forbidden or Other HTTP Response Code

Data to Receive as Input

Iteration 1

- ❖ Customer
 - ❖ Text Field
- ❖ Source/Destination IP Address
 - ❖ Single IPv4 Address
- ❖ Destination Port
 - ❖ Single Integer Value
- ❖ Business Justification
 - ❖ Text Field - Optional

Iteration 2

- ❖ Lists
- ❖ Source/Destination IP Addresses
 - ❖ Ranges, Subnet Masks
- ❖ Destination Port Ranges
- ❖ Business Justification
 - ❖ Required
- ❖ User
 - ❖ Text Field - Optional

Adding Custom Validators & Handling Validation Errors

- ❖ When to Use an Existing Validator?
 - ❖ Simple Validations
 - ❖ IPv4 Address
- ❖ When to Create a Custom Validator?
 - ❖ Complex Validations
 - ❖ Multiple Inputs Allowed
 - ❖ No Existing Validator Available
 - ❖ Grouping Single Validators
- ❖ What to Do When Data is Invalid?
 - ❖ Try Another Validator if Applicable
 - ❖ Raise ValidationError

Manipulating Data Flowing within the Application

- ❖ What Data is Needed for the Model?
 - ❖ Consider All Potential Input
 - ❖ Think About Future API Plans
 - ❖ 'Umbrella' Model
- ❖ What Data to Return & is Not Part of Model?
 - ❖ Simple to Add to Response Object
 - ❖ Consider What Data the User May Need
 - ❖ ID Associated With Actions
 - ❖ Error Data Needed by User
 - ❖ Consider Data Needed by Support Team

Enabling a Single API Call to Accept Multiple Input Formats

- ❖ When Would This Be Necessary?
 - ❖ Building the API for the Users
 - ❖ Updated Functionality for API
- ❖ How to Maintain Backwards-Compatibility?
 - ❖ Intercept Request via View Logic
 - ❖ Modify Input/Output as Necessary
- ❖ Versioning the API
 - ❖ Use Default Version When No Version Specified
 - ❖ How to Version
 - ❖ HTTP Accept Header
 - ❖ URL

Customizing the Save Process Behind an API Post

- ❖ Why Would This Need to be Done?
 - ❖ More Data Behind the Scenes
- ❖ Single API POST Saving to Multiple Models & Database Tables
 - ❖ Override Save Method of Main Model
 - ❖ Call Multiple Save Methods for Each Model
- ❖ Considerations
 - ❖ Save Each Database Entry
 - ❖ Performance Impact Possible
 - ❖ Using Bulk Create
 - ❖ Becomes Challenging When Dealing with Foreign Keys

Documentation Considerations

- ❖ Built-In Browsable API
 - ❖ Able to Customize the Theme
 - ❖ Potential for Fully-Functional API Access
 - ❖ Supports Viewing Multiple Data Formats
- ❖ Swagger
 - ❖ Similar to Browsable API
 - ❖ Enhanced Visuals
 - ❖ Potential Difficulty Setting What Actions Are Available
- ❖ Wikis

Other Lessons Learned

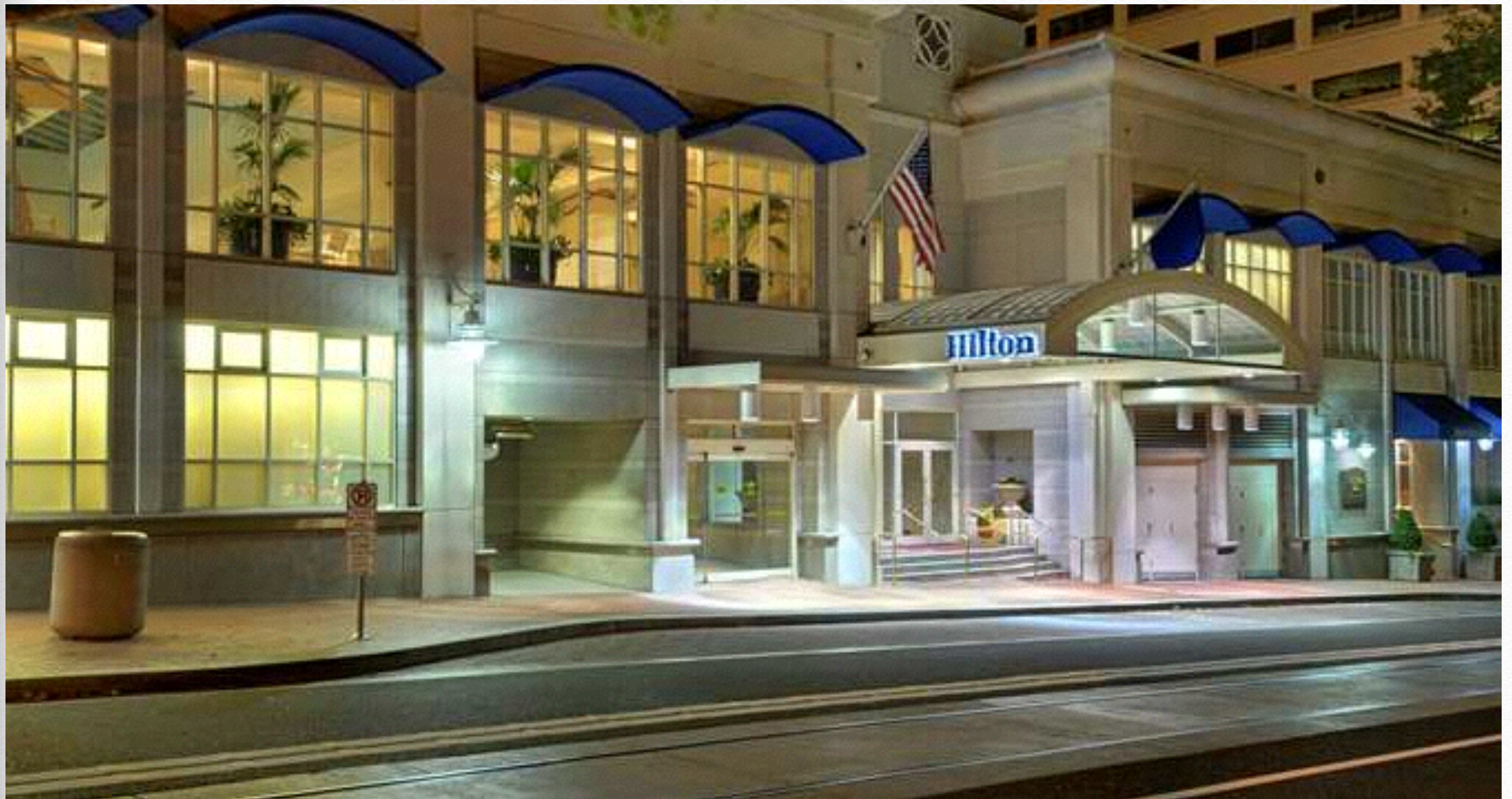
- ❖ Django Usage of Pre-Existing Database Tables
 - ❖ inspectdb
 - ❖ IntegerField vs AutoField
 - ❖ Add to Admin Interface for Web Management
- ❖ 'MySQL Server Has Gone Away' Error
 - ❖ Close Database Connection Prior to Each Query
 - ❖ Caution: Should NOT Be Done for Testing
- ❖ Coverage Module
 - ❖ Calculates Code Coverage of Tests

Other Lessons Learned

- ❖ Dynamic Customization of HTTP Response Header
- ❖ Parallel Support of Direct Server Access & Access via Web Proxy
 - ❖ Select Production-Level Web Server such as Apache
 - ❖ Configure Settings File(s)
 - ❖ FORCE_SCRIPT_NAME
 - ❖ Browsable API Links
 - ❖ Customize HTML Template
 - ❖ Customize Router for Addresses of Views
- ❖ Benefit of Multiple Settings Files
 - ❖ Potential Challenges

Closing Remarks

- ❖ Django & Django REST Framework
 - ❖ Easy Entry Point for Novices
 - ❖ Customizable for Advanced Users
 - ❖ Just Requires Willingness to Explore



Thank you!

Email: kenny.yarboro@gmail.com

Twitter: @KoreNetwork

Slides: <http://github.com/KennyNCSU/DjangoCon2014>