Implementing TypeScript Decorators

David Tucker
TECHNICAL ARCHITECT & CLOUD CONSULTANT
@_davidtucker_ davidtucker.net

Globomantics

Manufacturing company transitioning to TypeScript for internal API's

Interested in using Decorators to define cross-cutting data rules

Looking look automate route creation for data types using Node with Express

Looking to integrate data validation standards across all data types

```
@entity("people")
class Person {
  @id
  id: string;

  @persist
  @required
  firstName: string;

  @persist
  @required
  lastName: string;

  @persist
  @required
  @isEmail
  email: string;
}
```

- Define entity name
- Identify id property
- Specify what persists to the database
- Specify what values are required
- Define validation rules

API Server Automatic Entity Handling

Standard REST-based API routes created per entity

Logging for all calls to the API server

Automatic validation of data including error messaging based on configuration

Authentication with HTTP basic auth

Role-based authorization for reading, writing, and deleting per entity

Overview

Understand decorator metadata and the reflect-metadata module

Implement entity definition using decorators

Implement API call logging using decorators

Implement entity validation using decorators

Implement authentication and authorization using decorators

Metadata and Reflection with Decorators

Types of Decorators

Action Decorators

Perform an action where they are defined

Description Decorators

Simply describe an item for future use by the project

Action Decorator

```
class Person {
  @log
  public toJSON(): string {
    …
  }
}

function log(target: any, propertyKey: string, descriptor:
PropertyDescriptor) {
    console.log('Decorator Called');
}
```

Decorator Metadata

Decorators may simply describe a characteristic of what it is describing. In these cases, we need to store this metadata so it can be utilized at another time. This requires the use of an external module.

Description Decorator

```
import ‘reflect-metadata’;

class Person {
  @id
  firstName: string;
}

function id(target: any, propertyKey: string) {
    Reflect.defineMetadata("entity:id", propertyKey,
target);
}
```

Decorator Metadata

TypeScript documentation recommends the reflect-metadata module

Metadata reflection is experimental and could change in a future release

Requires the emitDecoratorMetadata configuration value to be true

Enables you to store, read, and delete metadata for each decorator type

```
@entity("people")
class Person {
  @id
  id: string;

  @persist
  @required
  firstName: string;

  @persist
  @required
  lastName: string;

  @persist
  @required
  @isEmail
  email: string;
}
```

- Define entity name for route creation
- Identify id property for fetching from database
- Specify what persists to the database
- Specify what values are required for validation of objects
- Define validation rules per field for validation of objects

Implementing the API Server

API Server

Utilizing jsonDB to mimic a production database

EntityRouter defines standard actions per entity based on REST standards

APIServer wraps express configuration for the server

BaseEntity is the base class for all of our entity types

Demo

Review initial project for API server

Review entity architecture to implement with decorators

Defining Entities with Decorators

Demo

Creating decorators to define entity characteristics

Update BaseEntity to work with persistence objects

Update EntityRouter based on entity metadata

Update APIServer to work with entity types

Implementing Logging

Demo

Examine how to wrap existing functionality using a method decorator

Log all API calls from Entity Router

Validating Entities with Decorators

Demo

Describing data rules for our Person entity

Creating validation decorators

Automating validation and error messages within EntityRouter

Authentication and Authorization

Demo

Implement HTTP basic auth for all API routes

Implement role-based permissions for users per entity type

Summary

Summary

- Reviewed decorator metadata and the reflect-metadata module
- Implemented entity definition using decorators
- Implemented API call logging using decorators
- Implemented entity validation using decorators
- Implemented authentication and authorization using decorators