

OGC Symbology

Winter School 2022



Ms. Prajwalita J. Chavan
IIT, Bombay

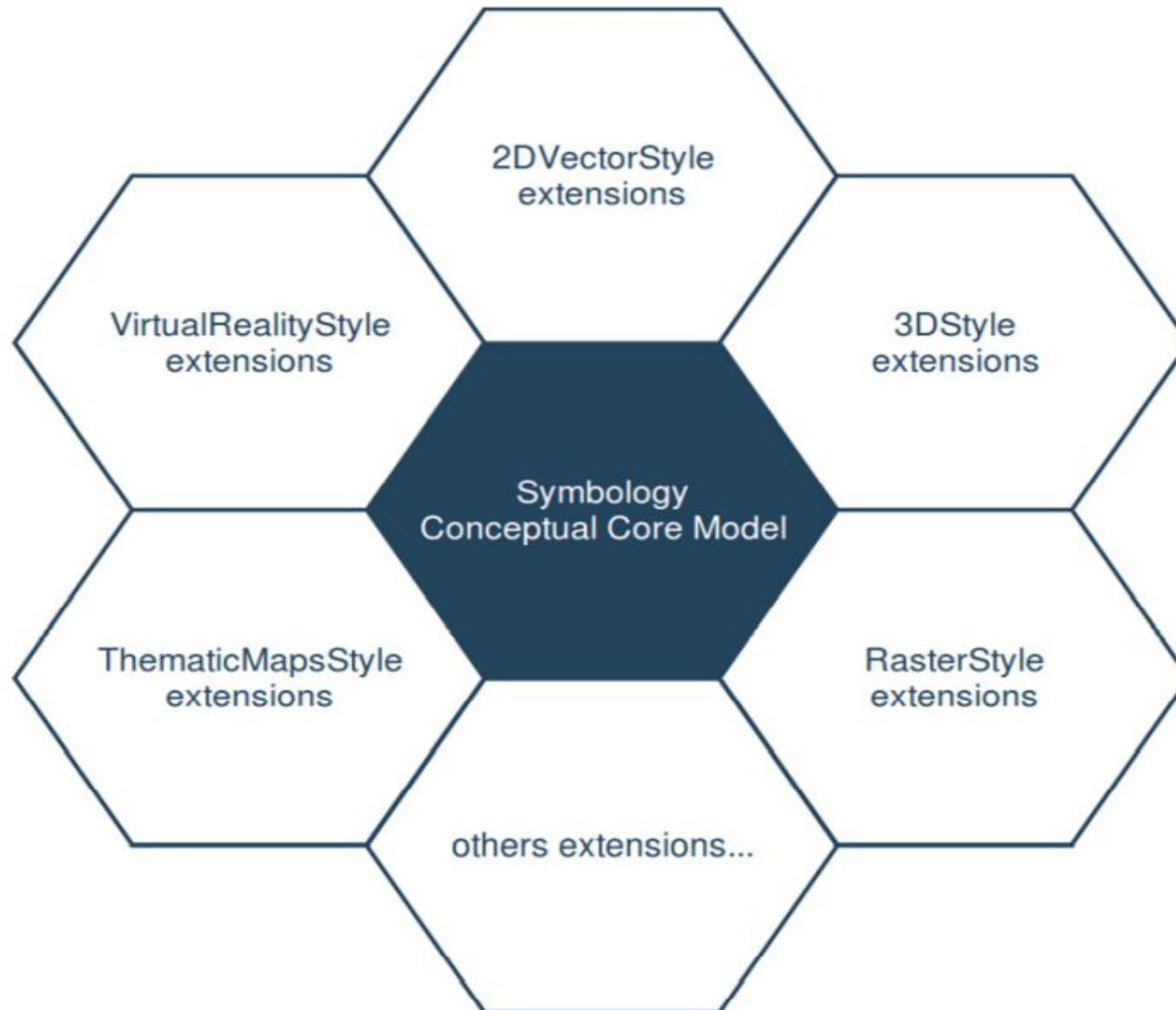
Overview

- About OGC API – Symbology
- SymCore
- Principles of Implementation
- Conceptual Model Core
- Class Style
- Class Rule
- Class Symbolizer
- Class Parameter Value
- Class Literal and color
- Class Fill
- Class Stroke
- Class Graphic and Graphic Size
- Class Label
- Class Font

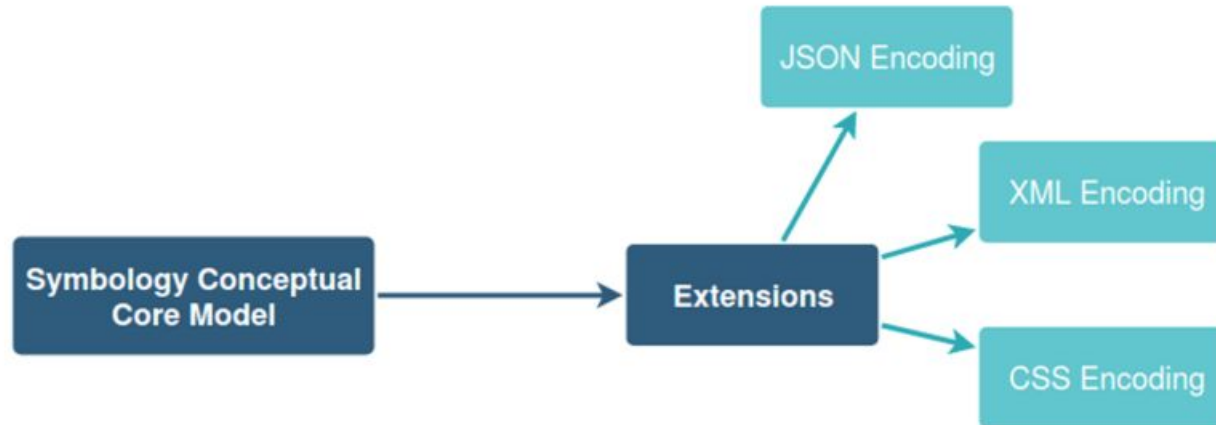
OGC API - Symbology

- **Publication Date:** 2020-10-15
- **Submitter:** : Erwan Bocher, Olivier Ertz: Switzerland
- The **SymCore** is a conceptual, modular, neutral model for the **portrayal of geographical data**
- The SymCore is a new approach to provide the **flexibility** required to achieve adequate **cartographic styling** and fill the needs of a variety of information communities; e.g., aviation symbols, weather symbols, thematic maps, etc
- It achieves high level **styling interoperability** without encoding dependencies

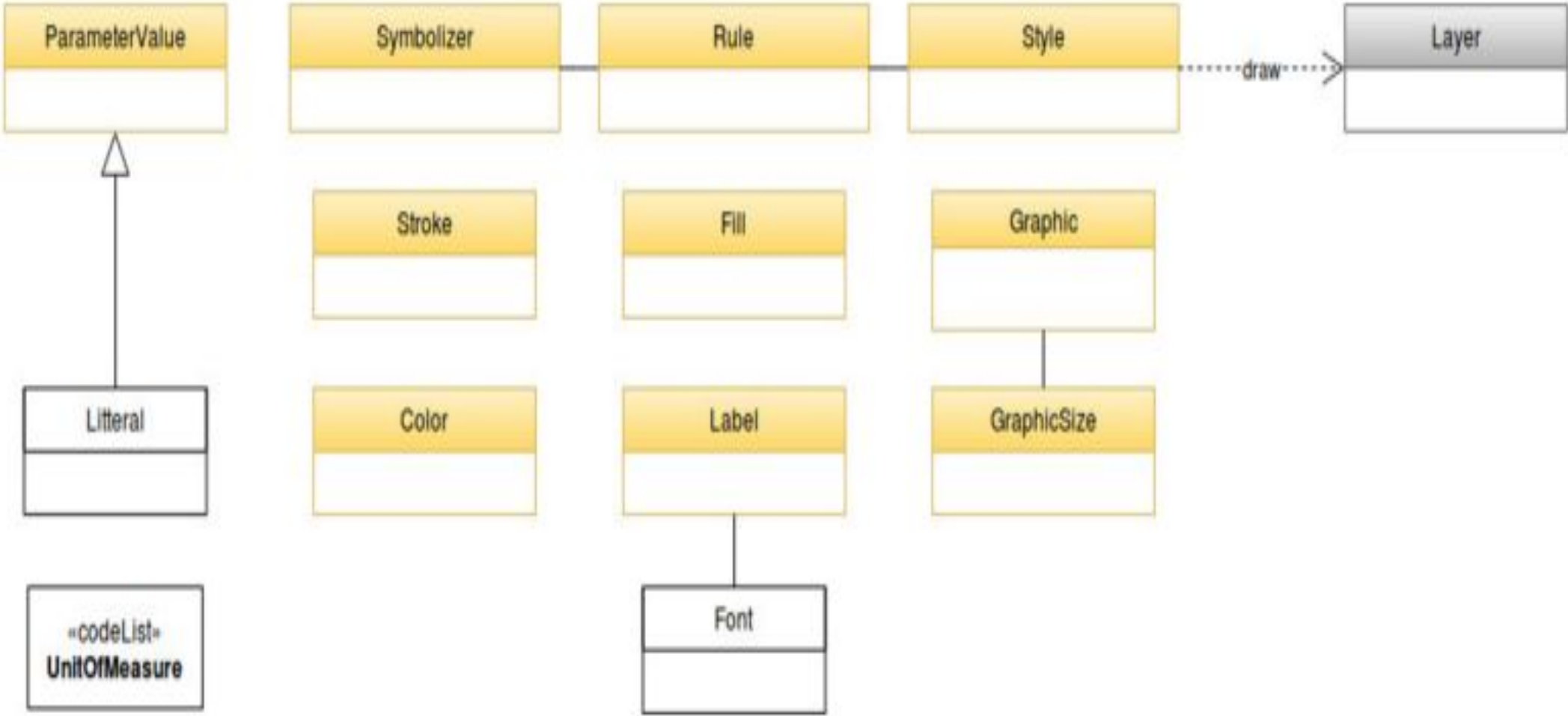
OGC Symbolology: Conceptual Model Core (SymCore)



Principles of implementation



Core Conceptual Model



Core Conceptual Model

REQUIREMENTS CLASS CORE

<http://www.opengis.net/spec/symbology/2.0/req/core>

Target type

Token

Dependencies

none

REQ 1

<http://www.opengis.net/spec/symbology/2.0/req/core/StyleClass> Implementations shall support the encoding of all properties of the StyleClass and meet all of the tabulated constraints and notes.

REQ 2

<http://www.opengis.net/spec/symbology/2.0/req/core/RuleClass> Implementations shall support the encoding of all RuleClass properties and meet all of the tabulated constraints and notes.

REQ 3

<http://www.opengis.net/spec/symbology/2.0/req/core/SymbolizerClass> Implementations shall support the encoding of all SymbolizerClass properties and meet all of the tabulated constraints and notes.

REQ 4

<http://www.opengis.net/spec/symbology/2.0/req/core/ParameterValueClass> Implementations shall support the encoding of all ParameterValue parameters class and meet all of the tabulated constraints and notes.

Core Conceptual Model

REQUIREMENTS CLASS CORE

REQ 5	http://www.opengis.net/spec/symbology/2.0/req/core/LiteralClass Implementations shall support the encoding of all parameters of the LiteralClass and meet all of the tabulated constraints and notes.
REQ 6	http://www.opengis.net/spec/symbology/2.0/req/core/UOMClass Implementations shall support the encoding of all properties of the UOMClass and meet all of the tabulated constraints and notes.
REQ 7	http://www.opengis.net/spec/symbology/2.0/req/core/ColorClass Implementations shall support the encoding of all properties of the ColorClass and meet all of the tabulated constraints and notes.
REQ 8	http://www.opengis.net/spec/symbology/2.0/req/core/FillClass Implementations shall support the encoding of all properties of the FillClass and meet all of the tabulated constraints and notes.
REQ 9	http://www.opengis.net/spec/symbology/2.0/req/core/StrokeClass Implementations shall support the encoding of all properties of the StrokeClass and meet all of the tabulated constraints and notes.
REQ 10	http://www.opengis.net/spec/symbology/2.0/req/core/GraphicClass Implementations shall support the encoding of all properties of the GraphicClass and meet all of the tabulated constraints and note.
REQ 11	http://www.opengis.net/spec/symbology/2.0/req/core/GraphicSizeClass Implementations shall support the encoding of all properties of the GraphicSizeClass and meet all of the tabulated constraints and notes.
REQ 12	http://www.opengis.net/spec/symbology/2.0/req/core/LabelClass Implementations shall support the encoding of all properties of the LabelClass and meet all of the tabulated constraints and notes.
REQ 13	http://www.opengis.net/spec/symbology/2.0/req/core/FontClass Implementations shall support the encoding of all properties of the FontClass and meet all of the tabulated constraints and notes.

Class Style

- This class is the **root concept of the Symbology Conceptual Core Model**. This class organizes the **rules of symbolizing instructions** to be applied by a rendering engine on a layer of geographic features

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
name	A string value to reference the Style	ParameterValue data type	Zero or one
title	Human readable title	ParameterValue data type	One
abstract	Human readable description	ParameterValue data type	Zero or one
rule	Rule(s) that drive(s) the rendering engine	Rule	One or more
extension	Any encoding should allow the user to extend the class to include custom items	Any	Zero or more

Class Rule

- This core class describes the **concept of a rule** in the Symbology model. Rules are used to organize symbolizing instructions and potentially to **define conditions of application** of these associated symbolizers

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
name	A string value to reference the Rule	ParameterValue data type	Zero or one
title	Human readable title	ParameterValue data type	One
abstract	Human readable description	ParameterValue data type	Zero or one
symbolizer	Symbolize(s) to apply by the rendering engine	Symbolizer	One or more
extension	Any encoding should allow the user to extend the class to include custom properties	Any	Zero or more

Class Symbolizer

- This class describes how to **portray geographic data given a shape** (e.g., area fill, line stroke, point marker, etc.) **and graphical properties** (e.g., color, opacity, font-family, etc.). As an abstract class, it is designed to be extended

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
name	A string value to reference the Symbolizer	ParameterValue data type	Zero or one
title	Human readable title	ParameterValue data type	One
abstract	Human readable description	ParameterValue data type	Zero or one
uom	Unit of measure to apply to all graphical properties of a Symbolizer	uom code	Zero or one
extension	Any encoding should allow the user to extend the class to include custom items	Any	Zero or more

Class Parameter Value

- The ParameterValue class represents a gateway that **provides the value to be used by a parameter in a styling context** of use (almost all styling parameters such as width, opacity, displacement, etc. are “parameter-values”)

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
language	Language identifier for the ParameterValue element. (a)	Character String. This language identifier shall be as specified in IETF RFC 4646.	zero or more
extension	Any encoding should allow the ability to extend the class to include custom items	Any	zero or more

(a) The language identifier should offer a way to adapt the ParameterValue to a specified language, e.g., display the title of a Rule element both in English and French.

Class Literal and Color

- The Literal class is a concrete implementation of the ParameterValue class. LiteralClass represents a **typed atomic literal value** as a constant explicitly specified

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
value	A value for the literal data	Any	one

- The ColorClass allows the **definition of color**. As an abstract class and part of the base of the core graphical concepts, this class is a global point of extension for specifying concrete definitions of colors (e.g., RGBColor extension)

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
extension	Any encoding should allow the extension of <i>ColorClass</i> with custom items	Any type	zero or more

Class Fill

- FillClass defines the graphical symbolizing parameters required to **draw the filling of a twodimensional shape** such as a polygon. As an abstract class and part of the base of the core graphical concepts, FillClass is a global point of extension for specifying concrete definitions for shape fill operations (e.g., the SolidFill and GraphicFill extensions).

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to all graphical properties within a Fill	uom code	zero or one
extension	Any encoding should allow the extension of a Fill operation with custom items	Any type	zero or more

Class Stroke

- StrokeClass defines the graphical symbolizing parameters for **drawing an outline** (e.g., for linear geometries or the exterior of a polygon geometry). As an abstract class and part of the base of the core graphical concepts, StrokeClass is a global point of extension to specify concrete ways to draw outlines (e.g., the PenStroke and GraphicStroke extensions).

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to all graphical properties inside a Stroke	uom code	zero or one
extension	Any encoding should allow to extend a Stroke with custom items	Any type	zero or more

Class Graphic and Graphic Size

- The Graphic class defines the parameters for **drawing a graphic symbol such as shape, color(s), and size.**

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to all graphical properties within a Graphic	uom code	zero or one
graphicSize	Rendering size of the graphic	GraphicSize data type	zero or one
extension	Any encoding should allow to extend a Graphic with custom items	Any type	zero or more

Class Graphic and Graphic Size

- The GraphicSize class determines the **size of the graphic** when it is rendered. As an abstract class, it is designed to be extended to support the various ways the size could be specified such as by a single value, a rectangular box, or by a three-dimensional cube

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
extension	Any encoding should allow to extend a GraphicSize with custom items	Any type	zero or more

Class Label

- LabelClass defines the graphical symbolizing properties for **drawing a text label**. As an abstract class and part of the base of the core graphical concepts, LabelClass is a point of extension to specify concrete ways to draw text label according to placement behaviors (e.g., a PointLabel or LineLabel).

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to the affected graphical properties within a Label	uom code	zero or one
labelText	Text-label content to draw	ParameterValue data type String	one
font	Font definition to draw the text-label content	Font data type Default value: system-dependent	zero or one
fill	Filling style to draw the glyphs	Fill data type	zero or one
extension	Any encoding should allow to extend a Label with custom items	Any type	zero or more

Class Font

- The FontClass describes the **font properties to apply for the rendering of a text string**

NAME	DEFINITION	DATA TYPE AND VALUE	MULTIPLICITY
uom	Unit of measure to apply to the affected graphical properties within a Font	uom code	zero or one
fontFamily	Font family name (a)	ParameterValue data type CharacterString	zero or more
fontSize	Font size when applying the font to a text string (b)	ParameterValue data type Float	zero or one
fontWeight	Amount of weight or boldness to use for a font	ParameterValue data type CharacterString	zero or one
fontStyle	Style to use for a font	ParameterValue data type CharacterString	zero or one
extension	Any encoding should allow to extend a Font with custom items	Any type	zero or more

Hands-on

Software Testing, Monitoring and x +

try.smartbear.com/swaggerhub/log-in

Bookmarks Welcome to Drona... Department of Co... OGC Portal System... GIS MiraMon | CREAF OGC Testbed-17: At... opf-style-api | 1.0.0... opf-features-api | 1... My Projects | OS Da... Other bookmarks

SMARTBEAR

Don't have an account? [SIGN UP](#)

SMARTBEAR SwaggerHub

Log into SwaggerHub

Work Email

Password

Log In

[Login with SSO](#) [Forgot password?](#)

or

[Continue with GitHub](#)

We use SwaggerHub to organize and define our schema, as well as automatically generate front-end models. It's great, helps us communicate and keep everyone's code up to date!

Nicholas Papadopoulos
Software Engineer, Boyle Transportation

29°C Sunny

Search

ENG IN 18:42 25-12-2022

Hands-on

My Hub | SwaggerHub

app.swaggerhub.com/home

SMARTBEAR SwaggerHub

MY hub

Create New

MY hub

Search

ORGANIZATIONS

Prajwalita

Select a Template or create a Blank API

Enter a unique name for your API and select a Template. Select 'None' for a Blank API.

Owner: Prajwalita

Project: -- None --

Specification: OpenAPI 3.0.x

Template: Simple API

Name: PMT

Visibility: Public

Auto Mock API: OFF ON

Cancel Create API

Create New APIs Document APIs Share and Collaborate

29°C Sunny

Search

ENG IN

17:59 25-12-2022

Hands-on

The screenshot displays the SwaggerHub interface for the 'emp API' (version 1.0.0). The central editor shows the following JSON Schema:

```
89 - salary
90 - joinDate
91 properties:
92   id:
93     type: string
94     format: uuid
95     example: d290f1ee-6c54-4b01-90e6-d701748f0851
96   name:
97     type: string
98     example: Widget Adapter
99   releaseDate:
100     type: string
101     format: date-time
102     example: '2016-08-29T09:12:33.001Z'
103   manufacturer:
104     $ref: '#/components/schemas/Manufacturer'
105 Manufacturer:
106   required:
107     - name
108   properties:
109     name:
110       type: string
111       example: ACME Corporation
112     homePage:
113       type: string
114       format: url
115       example: 'https://www.acme-corp.com'
116     phone:
117       type: string
118       example: 408-867-5309
119   type: object
```

The right-hand pane displays the API title 'emp API', version '1.0.0', and a description 'This is a simple API'. It also includes links for 'Contact the developer' and 'Apache 2.0'. Below this, the 'Servers' section shows a dropdown menu with the URL 'https://virtserver.swaggerhub.com/Prajwalita/emp/1.0.0 - Swag...'. The 'admins' section is titled 'Secured Admin-only calls' and features a 'POST /emp' endpoint with the description 'adds an emp item'. The 'developers' section is titled 'Operations available to regular developers'.

The bottom of the image shows a Windows taskbar with the system tray displaying '29°C Sunny', the date '25-12-2022', and the time '18:39'. Various application icons are visible in the taskbar.

Hands-on

The screenshot shows a web browser window with the URL `app.swaggerhub.com/settings/apiKey`. The page title is "API Key | SwaggerHub". The SwaggerHub logo is visible in the top left. The user's name, "Prajwalita", is shown in the top right. The left sidebar contains navigation options: "MY hub", "Account Overview", "API Key" (highlighted), and "My Organizations". The main content area is titled "API Key" and displays a text input field containing the API key: `24097147-0a5d-4efe-8128-80ab3e597b41`. To the right of the input field is a "Copy API Key" button and a small menu icon. The Windows taskbar at the bottom shows the system tray with a temperature of 29°C, the date 25-12-2022, and the time 18:40.

Hands-on

The screenshot displays the SwaggerHub interface for an API named 'emp' with version '1.0.0'. The browser address bar shows the URL: `app.swaggerhub.com/apis/Prajwalita/emp/1.0.0#/admins/addEmp`. The SwaggerHub logo and user profile 'Prajwalita' are visible in the top navigation bar.

The main content area is divided into two panels. The left panel, titled 'Info', shows a sidebar with navigation options: 'Info', 'Tags', 'Servers', and a search bar. Below these, there are sections for 'admins' (with a 'POST /emp' endpoint), 'developers' (with a 'GET /emp' endpoint), and 'Schemas' (listing 'Empltem' and 'Company').

The right panel displays the details for the 'POST /emp' endpoint, which is described as 'adds an emp item'. It includes a 'Try it out' button and a 'Request body' section set to 'application/json'. An 'Example Value' is provided, showing a JSON object with an 'id' field.

The central editor area shows the OpenAPI specification for the 'EmpItem' schema, with line numbers 81 through 111. The code is as follows:

```
81     description: Emp item to add
82   components:
83     schemas:
84       EmpItem:
85         type: object
86         required:
87           - id
88           - name
89           - salary
90           - joinDate
91         properties:
92           id:
93             type: string
94             format: uuid
95             example: d290f1ee-6c54-4b01-90e6-d701748f0851
96           name:
97             type: string
98             example: Widget Adapter
99           releaseDate:
100            type: string
101            format: date-time
102            example: '2016-08-29T09:12:33.001Z'
103            manufacturer:
104              $ref: '#/components/schemas/Company'
105       Company:
106         required:
107           - name
108         properties:
109           name:
110             type: string
111             example: ACME Corporation
```


Hands-on

The screenshot displays the SwaggerHub interface for an API named 'emp'. The browser address bar shows the URL: `app.swaggerhub.com/apis/Prajwalita/emp/1.0.0#/admins/addEmp`. The SwaggerHub logo and the user name 'Prajwalita' are visible at the top.

The main content area is divided into three sections:

- Left Panel (Navigation):** Contains 'Info', 'Tags', 'Servers', a search bar, and a list of endpoints: 'admins' (POST /emp), 'developers' (GET /emp), and 'Schemas' (Empltem, Company).
- Center Panel (Code Editor):** Displays a JSON schema for the 'Company' object. The schema includes fields like 'salary', 'joinDate', 'id' (with type 'string' and format 'uuid'), 'name' (with type 'string' and example 'Widget Adapter'), 'releaseDate' (with type 'string' and format 'date-time'), 'manufacturer' (with a \$ref to a component), and 'Company' (with a 'required' field 'name' and a 'properties' object containing 'name', 'homePage', and 'phone').
- Right Panel (API Detail):** Shows the 'admins' endpoint with a 'POST' method and the path '/emp'. The description is 'adds an emp item'. Below this, there are sections for 'Parameters' (No parameters), 'Request body' (set to 'application/json'), and a 'Request body' section containing a JSON object:

```
{  "id": "1",  "name": "iitb",  "releaseDate": "99875678",  "manufacturer": {    "name": "praj",    "homePage": "https://www.iitb.com",    "phone": "989998866"  }}
```

At the bottom of the interface, there is a status bar showing 'Last Saved: 7:08:23 pm - Dec 25, 2022' and a 'VALID' indicator. The Windows taskbar at the very bottom shows the system tray with the date '25-12-2022' and time '19:24'.

THANK YOU!

prajwalita.chavan@gmail.com
prajwalita@iitb.ac.in

#OGCAPI