RESTful API, MVC, OOP, ORM

Language Technology and Web Applications

Johannes Graën, Nikolina Rajović, Igor Mustač

Department of Computational Linguistics &
Linguistic Research Infrastructure (LiRI)

November 08, 2023

Topics

1. RESTful API
2. MVC
3. OOP
4. ORM

Topics

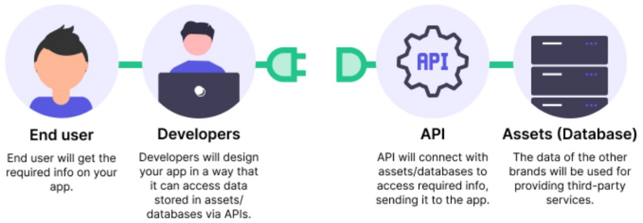
1. RESTful API
2. MVC
3. OOP
4. ORM

What is an API?

- Application Programming Interface
- Way the two computers or applications talk to each other

What is an API?

- Application Programming Interface
- Way the two computers or applications talk to each other


API
End user
End user will get the required info on your app.
Developers
Developers will design your app in a way that it can access data stored in assets/ databases via APIs.
API
API will connect with assets/databases to access required info, sending it to the app.
Assets (Database)
The data of the other brands will be used for providing third-party services.

API Examples

- Bot API – building a bot for Telegram
- Telegram API – own client

- Posting Tweets and images
- Read Tweets

- Instagram Graph API – publishing, comments, …
- Instagram Messaging – for messages

What is a RESTful API?

- **Re**presentational **S**tate **T**ransfer
- Set of rules (architectural style)
- API that follows these rules -> RESTful API

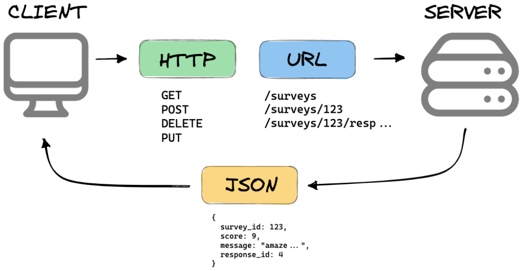
Key Principles

- Client-server
- Rest APIs are stateless
- Uniform Interface – one resource, one URI
- Cacheable

REST and HTTP

- **H**yper**T**ext **T**ransfer **P**rotocol
- HTTP methods
  POST, GET, PUT, DELETE

CREATE
READ
UPDATE
DELETE
C
R
U
D



```
GET
POST
DELETE
PUT
```

```
/surveys
/surveys/123
/surveys/123/resp ...
```

```
{
  survey_id: 123,
  score: 9,
  message: "amaze ... ",
  response_id: 4
}
```

URIs organization

- **U**niform **R**esource **I**dentifiers

```
GET https://www.example.com/api/v1/users

GET https://www.example.com/api/v1/books
```

URIs organization

- Resources should be named by plural nouns

```
GET https://www.example.com/api/v1/books

GET https://www.example.com/api/v1/getAllBooks
POST https://www.example.com/api/v1/addBook
```

URIs organization

- Query parameters for filtering

```
GET https://example.com/api/v1/books?author=tolkien
```

URIs organization

- Endpoint for individual resources

```
GET https://www.example.com/api/v1/books/12
```

URIs organization

- Endpoint for resources

```
GET https://www.example.com/api/v1/users/20/books
```

URIs organization

- Update a book

```
PUT https://www.example.com/api/v1/books/12
```

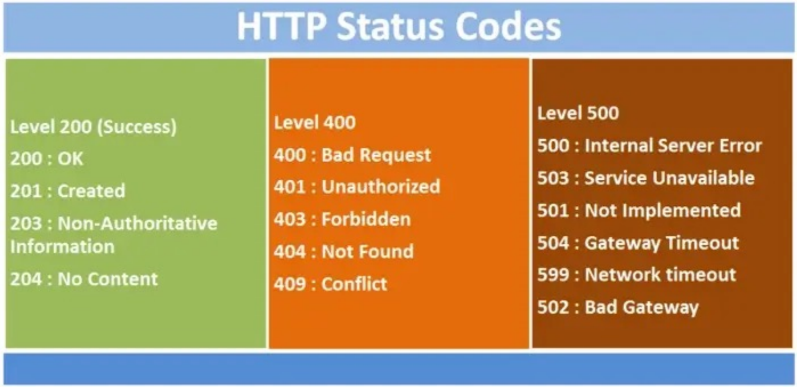
URIs organization

- Delete a book

```
DELETE https://www.example.com/api/v1/books/12
```

REST and HTTP

- Use HTTP codes to provide feedback



HTTP Status Codes

Level 200 (Success)	Level 400	Level 500
200 : OK	400 : Bad Request	500 : Internal Server Error
201 : Created	401 : Unauthorized	503 : Service Unavailable
203 : Non-Authoritative Information	403 : Forbidden	501 : Not Implemented
204 : No Content	404 : Not Found	504 : Gateway Timeout
	409 : Conflict	599 : Network timeout
		502 : Bad Gateway

Documentation

- When building an API good documentation is critical for developers
- Request and response examples

Security and REST APIs

- Authentication and Authorization (e.g., OAuth 2.0)
- Bearer token
- Secure transport (HTTPS/SSL)
- Audits

Versioning in RESTful APIs

- URI versioning (e.g., /v1/users)
- Header versioning

Topics

1. RESTful API
2. MVC
3. OOP
4. ORM

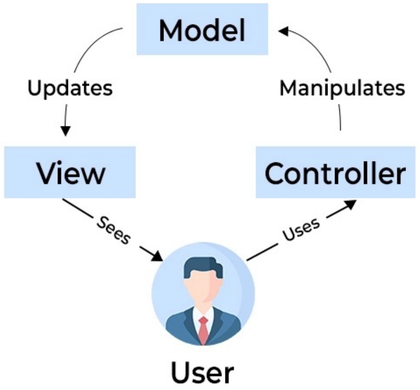
What is an MVC?

- Model: The data layer of the application
- View: The presentation layer (UI)
- Controller: The logic layer that handles user interaction, works with the model, and ultimately selects a view to render

- Separates application logic from the user interface

What is an MVC?

- **Model**
  The kitchen where the food is prepared
  (data and business logic)
- **View**
  The dining area where guests eat
- **Controller**
  The waitstaff takes orders and bring food



Benefits of MVC

- Simplifies management of complex applications
- Improves organized coding and development processes
- Facilitates scalability and maintenance

Topics

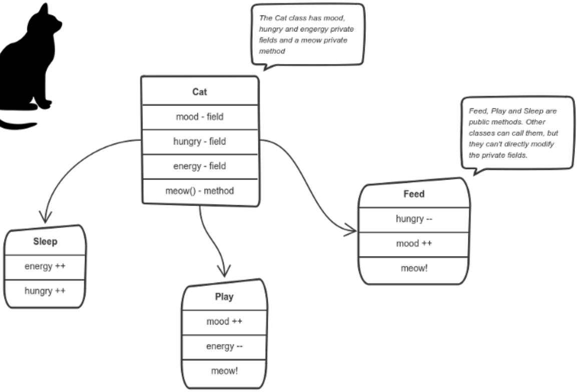
1. RESTful API
2. MVC
3. OOP
4. ORM

What is an OOP?

- Object-Oriented Programming
- A way of computer programming using the idea of “objects” to represent data and methods

What is an OOP?

- Object-Oriented Programming
- A way of computer programming using the idea of “objects” to represent data and methods


The Cat class has mood, hungry and engergy private fields and a meow private method
Cat
mood - field
hungry - field
energy - field
meow() - method
Feed, Play and Sleep are public methods. Other classes can call them, but they can't directly modify the private fields.
Sleep
energy ++
hungry ++
Play
mood ++
energy --
meow!
Feed
hungry --
mood ++
meow!

Classes vs Objects

- Class is blueprints for objects defining the common attributes and behavior

```
class Cat():
  def __init__(self, name, hungry,
                energy, mood):
    self.name = name
    self.hungry = hungry
    self.energy = energy
    self.mood = mood

  def meow(self):
    print("Meow!!")

  def feed(self):
    self.hungry -= 1
    self.mood += 1
    self.meow()

  def play(self):
    self.energy -= 1
    self.mood += 1
    self.meow()
```

Classes vs Objects

- An object is an instance of a class

```
// Create an object
cat1 = Cat('Fluffy', 7, 7, 7)
cat1.play()

cat1.energy  // 6
```

```
class Cat():
  def __init__(self, name, hungry,
               energy, mood):
    self.name = name
    self.hungry = hungry
    self.energy = energy
    self.mood = mood

  def meow(self):
    print("Meow!!")

  def feed(self):
    self.hungry -= 1
    self.mood += 1
    self.meow()

  def play(self):
    self.energy -= 1
    self.mood += 1
    self.meow()
```

OOP Benefits

- **Modularity**
  Just like LEGO bricks, OOP lets us build modules that fit together. It makes complex systems easier to manage

- **Maintainability**
  Because objects are self-contained, we can swap out parts of our system with less risk of breaking other parts

- **Reusability**
  Just like using a template to create multiple documents, OOP lets us reuse code for multiple purposes

Inheritance

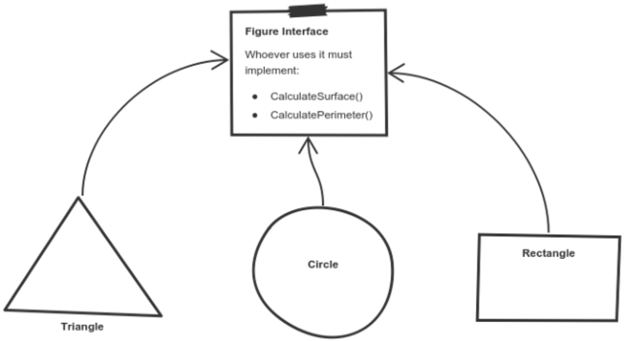


Figure Interface

Whoever uses it must implement:

- CalculateSurface()
- CalculatePerimeter()

Triangle, **Circle** and **Rectangle** inherit the Figure interface or abstract class.

They implement their own versions of **CalculateSurface()** and **CalculatePerimeter()**.

They can be used in a mixed collection of Figures.

Inheritance

```python
class Figure():
  def __init__(self, color):
    self.color = color

  def area(self):
    pass

  def perimeter(self):
    pass
```

Inheritance

```python
class Figure():
  def __init__(self, color):
    self.color = color

  def area(self):
    pass

  def perimeter(self):
    pass


class Square(Figure):
  def __init__(self, color, side):
    super().__init__(color)
    self.side = side

  def area(self):
    return self.side ** 2

  def perimeter(self):
    return 4 * self.side
```

Inheritance

```python
class Figure():
  def __init__(self, color):
    self.color = color

  def area(self):
    pass

  def perimeter(self):
    pass


class Square(Figure):
  def __init__(self, color, side):
    super().__init__(color)
    self.side = side

  def area(self):
    return self.side ** 2

  def perimeter(self):
    return 4 * self.side
```

```python
class Rectangle(Figure):
  def __init__(self, color, sideA, sideB):
    super().__init__(color)
    self.sideA = sideA
    self.sideB = sideB

  def area(self):
    return self.sideA * self.sideB

  def perimeter(self):
    return 2 * self.sideA + 2 * self.sideB
```

Topics

1. RESTful API
2. MVC
3. OOP
4. ORM

What is an ORM?

- Object-relational mapping
- A technique that lets you query and manipulate data from a database using an object-oriented paradigm
- Solving: Mismatch between the object-oriented world of applications and the relational world of databases

Why ORM?

- Write Python code instead of SQL
- Object-oriented code is often easier to read and maintain
- Allows developers to switch between different databases with minimal changes in code

SQLAlchemy

- SQL toolkit and ORM for Python
- Integrates with Python models

SQLAlchemy

- SQL toolkit and ORM for Python

- Integrates with Python models

```
@app.route("/get-entries", methods=["GET"])
def get_entries():
  """ Return the data from the database """
  select_query = "SELECT text FROM sentences"
  cursor.execute(select_query)
  messages = cursor.fetchall()
  return jsonify(messages)
```

SQLAlchemy

- SQL toolkit and ORM for Python
- Integrates with Python models

```
@app.route("/get-entries", methods=["GET"])
def get_entries():
  """ Return the data from the database """
  messages = [sentence.text for sentence in Sentence.query.all()]
  return jsonify(messages)
```

SQLAlchemy

```
class Sentence(db.Model):
    __tablename__ = "sentences"

    id = db.Column(db.Integer, primary_key=True)
    text = db.Column(db.String(1000), unique=True, nullable=False)
```