

# **RESTing on MVC using Node.js and Express**

**Daniel Amariei**

**FII, UAIC**

# Outline

Patterns

MVC

REST

Use-case

# Patterns

# Patterns

software construction

software architecture

framework

Patterns

# Patterns

How do we define a Pattern?

*A solution to a problem in a context [1].*

decomposing complex structures

general solution

# Pattern hierarchy

Programming
Design
Architectural

# Pattern hierarchy

Programming
Design
Architectural

# Pattern hierarchy

Programming
Design
Architectural



# Pattern hierarchy

Programming
Design
Architectural

# MVC

# MVC

Programming  
Design  
Architectural

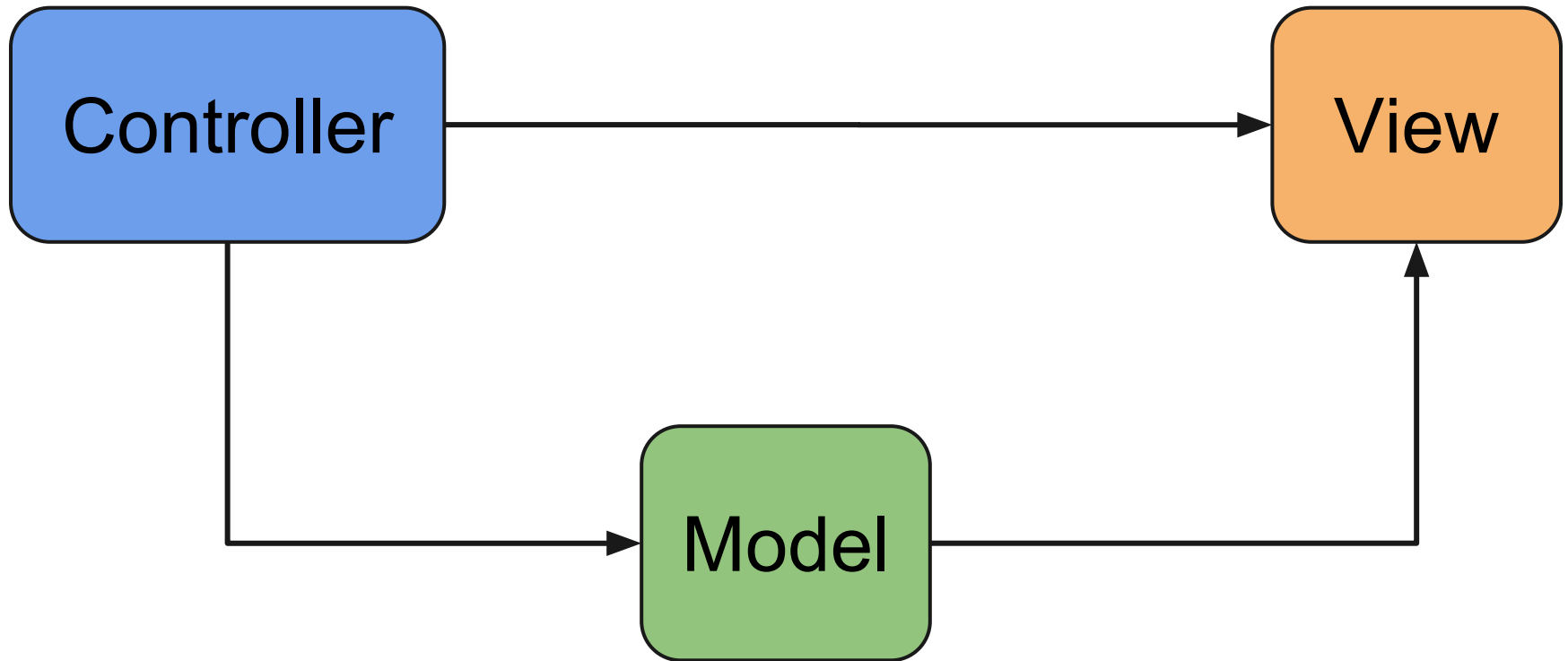
# MVC

~~Programming~~

~~Design~~

Architectural

# MVC



# REST

# REST

WADE, FII, UAIC

[http://profs.info.uaic.  
ro/~busaco/teach/courses/wade/web-film.  
html#week3](http://profs.info.uaic.ro/~busaco/teach/courses/wade/web-film.html#week3)

# REST

What is REST?

architectural style

resources vs. actions

not limited to HTTP

resources vs. representations



# REST

constraints

uniform interface

stateless

cacheable

client and server

layered system

code on demand (optional)

# REST

constraints

**uniform interface**

resources are identified by URIs

manipulation through representation

self-descriptive

HATEOAS

# REST

constraints

**stateless**

# REST

constraints

**cacheable**

improves scalability

# REST

constraints

**client and server**

separation of concerns

# REST

constraints

**layered system**

# REST

constraints

**code on demand**

extending client functionality

executable logic

# Use-case



# Use-case

Online Agenda

# Step 1

resource conceptualization

agenda

contact

# Step 1

identifying resources (URIs)

`http://authority/agenda`

`http://authority/agenda/id`

# Step 2 (Representation)

```
{ contact:  
  {  
    firstName: String,  
    lastName: String,  
    phoneNumbers: [phoneNumber],  
    emails: [String],  
    tags: [String]  
  }  
}
```

# Step 2 (Representation)

```
{ phoneNumber:  
  {  
    type: String,  
    number: String,  
  }  
}
```

# Step 2 (Representation)

**discussion**

advantages and disadvantages

# Step 2 (Representation)

{ agenda: [contact] }

vs.

{ agenda: [id] }

# Step 2 (Representation)

**discussion**

advantages and disadvantages



# Step 3

## hypermedia controls (agenda)

```
{ hypermediaControlsAgenda:  
  [  
    {link: {rel: 'self', method: 'GET', uri: '/agenda'}},  
    {link: {rel: 'add', method: 'POST', uri: '/agenda'}},  
    {link: {rel: 'filter', method: 'GET', uri: '/agenda?filter=tag'}},  
  ]  
}
```

# Step 3

## hypermedia controls (contact)

```
{ hypermediaControlsContact:  
  [  
    {link: {rel: 'self', method: 'GET', uri: '/agenda/id'}},  
    {link: {rel: 'update', method: 'PUT', uri: '/agenda/id'}},  
    {link: {rel: 'add', method: 'POST', uri: '/agenda'}},  
  ]  
}
```

# Step 4

## dereferencing – representations

Resource	GET	PUT	POST	DELETE
/agenda	contact-list	—	contact contact-list	—
/contact/id	contact	contact	—	—

# Step 4

## dereferencing – status codes

Resource	GET	PUT	POST	DELETE
/agenda	200, 404, 500	405	201, 400	200, 204, 500
/contact/id	200, 404, 500	204, 400	405	204, 409, 500

# HATEOAS

**discussion**  
problems

# Online Agenda

required software and structure

MongoDB

Node.js

Express-generator

Generate app. structure

Dependencies

Mongoose

Server Startup Code

# Online Agenda

## developing the application

create the Schema and Model for the Contact

add the required middleware that handles access to  
the resources

**Thank you!**



# Questions

?

# Bibliography

1. \*\*\*, Bob Tarr, Design Patterns in Java. *Introduction To Design Patterns*. <http://userpages.umbc.edu/~tarr/dp/lectures/IntroToDP.pdf>.
2. \*\*\*, Bob Tarr, CMSC446. *Introduction To Design Patterns*. <http://userpages.umbc.edu/~tarr/dp/spr06/cs446.htm>.
3. \*\*\*, Stack Overflow. *Difference between design pattern and Architecture?* <https://stackoverflow.com/questions/4243187/difference-between-design-pattern-and-architecture>
4. Brahma Dathan, Sarnath Ramnath. *Object-Oriented Analysis and Design*. Springer, 2011.
5. \*\*\*, Sabin Corneliu Buraga, Web App Development. *Node.js*. <http://profs.info.uaic.ro/~busaco/teach/courses/web/presentations/web-nodejs.pdf>.

# Bibliography

6. \*\*\*, Bob Tarr, Design Patterns in Java. *Introduction To Design Patterns*. <http://userpages.umbc.edu/~tarr/dp/lectures/IntroToDP.pdf>.
7. \*\*\*, Bob Tarr, CMSC446. *Introduction To Design Patterns*. <http://userpages.umbc.edu/~tarr/dp/spr06/cs446.htm>.
8. \*\*\*, Stack Overflow. *Difference between design pattern and Architecture?* <https://stackoverflow.com/questions/4243187/difference-between-design-pattern-and-architecture>.
9. Brahma Dathan, Sarnath Ramnath. *Object-Oriented Analysis and Design*. Springer, 2011.
10. \*\*\*, Sabin Corneliu Buraga, Web App Development. *Node.js*. <http://profs.info.uaic.ro/~busaco/teach/courses/web/presentations/web-nodejs.pdf>.
11. \*\*\*, Martin Fowler. *Richardson Maturity Model*. <http://martinfowler.com/articles/richardsonMaturityModel.html>.

# Bibliography

12. \*\*\*, *httpstatus.es*. <http://httpstatus.es/>.