PHP – Ramverk – Lumen

En introduktion till ramverk

Förändring nästa vecka!

Inget på onsdagen, föreläsning + labb på torsdagen

Vad vi gjort hittills…

- Introduktion till PHP
  - Var vi använder PHP
  - Hur vi använder PHP
  - Variabler, iterationer, if-satser, namespace, klasser, etc.
- Introduktion till *dependency menagement* i.e. Composer
- Versionshantering, git flow
- HTTP
- REST

Användning av PHP (1)

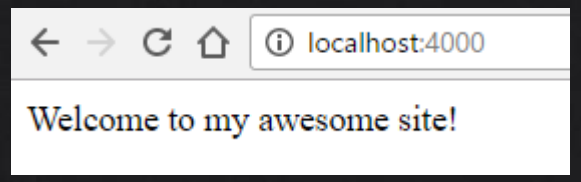
```php
<?php

echo "This is an awesome script!";

```

```
php index.php
This is an awesome script!
```

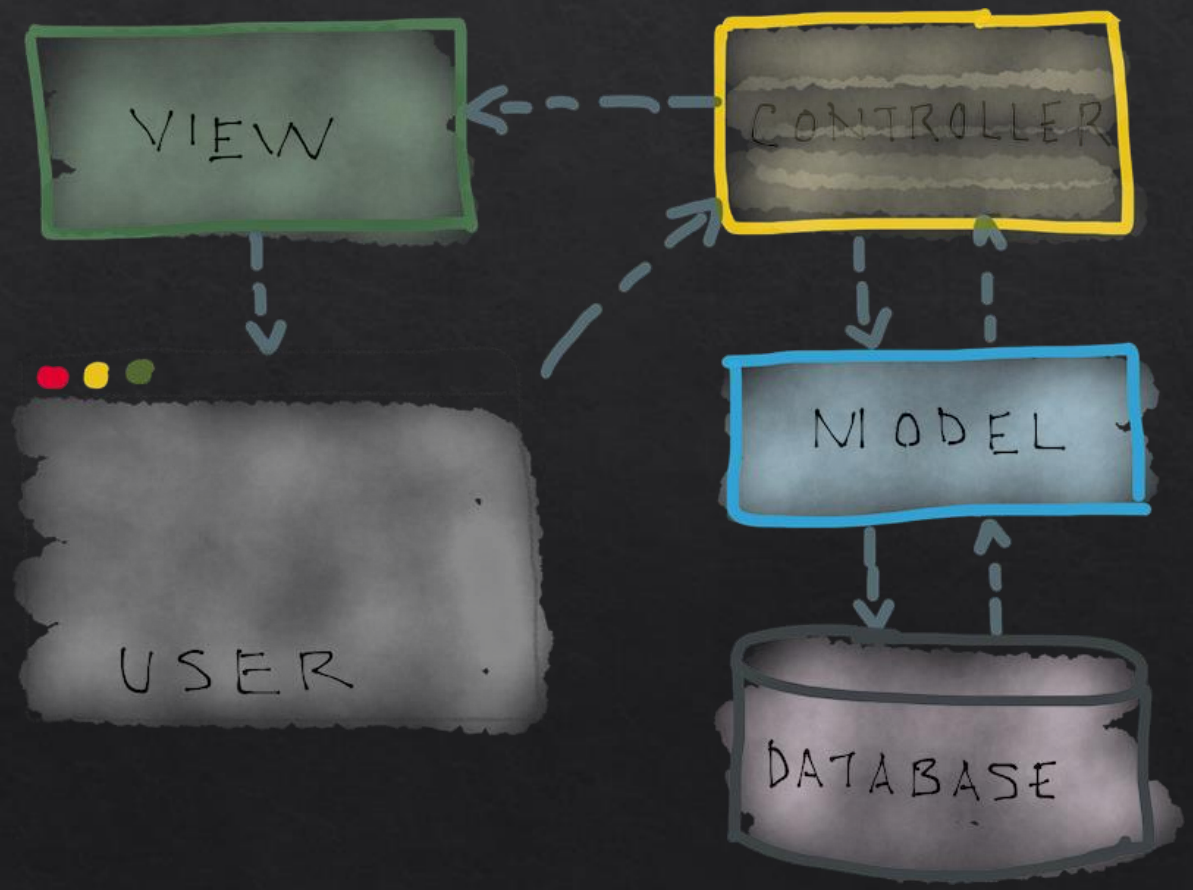
Användning av PHP (2)

```php
<!doctype html>
<html>
<head>
    <title>Example document</title>
</head>
<body>
    <?php
        echo "Welcome to my awesome site!";
    ?>
</body>
</html>
```

```html
<!doctype html>
<html>
<head>
    <title>Example document</title>
</head>
<body>
    Welcome to my awesome site!
</body>
</html>
```



Användning av PHP (3)



Hantera formulärsdata

Hantera data från GET-anrop

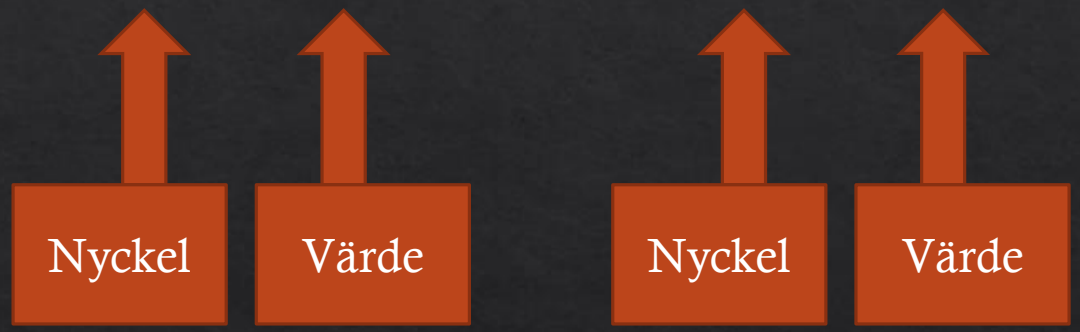
- Men HTTP-anrop (GET) kan man skickar med parametrar, t.ex.
  - index.php?course=DA287A

movie.php | index.php

```php
<?php

$course = $_GET['course'];
echo $course; // Prints "DA287A"

```

Hantera data från GET-anrop

- Men HTTP-anrop (GET) kan man skickar med parametrar, t.ex.
  - index.php?course=DA287A&courseResponsible=Anton



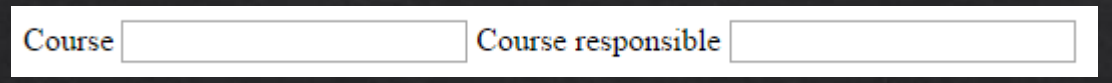
movie.php | index.php

```php
<?php

$course = $_GET['course'];
$courseResponsible = $_GET['courseResponsible'];
echo $course; // Prints "DA287A"
echo $courseResponsible; // Prints "Anton"

```

Hantera data från POST-anrop

- Till skillnad från GET (som används för att efterfråga data) så används POST för att skicka med data. Detta görs t.ex. genom ett HTML-formulär:

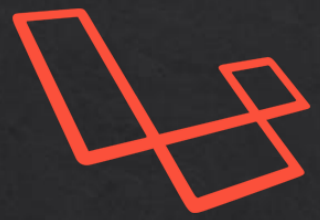
```
<form action="index.ph      ethod="post">
    <label for="course">Co      </label>
    <input type="text" name="course" id=      ourse">
    <label for="courseResponsible">Co      e responsible</label>
    <input type="text" name="courseResponsible" id="courseResponsible">
</form>
```



- Och tas sedan emot genom $_POST-funktionen i PHP.

```
ovie.php    index.php
1  <?php
2
3  $course = $_POST['course'];
4  $courseResponsible = $_POST['courseResponsible'];
5  echo $course; // Prints "DA287A"
6  echo $courseResponsible; // Prints "Anton"
7
8
```

Webbprogrammering



Laravel

Lumen 5

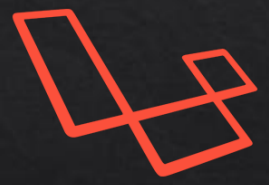
Bootstrap

Kursmoment

1. PHP & Composer

2. MVC & Lumen/Laravel

Lumen 5



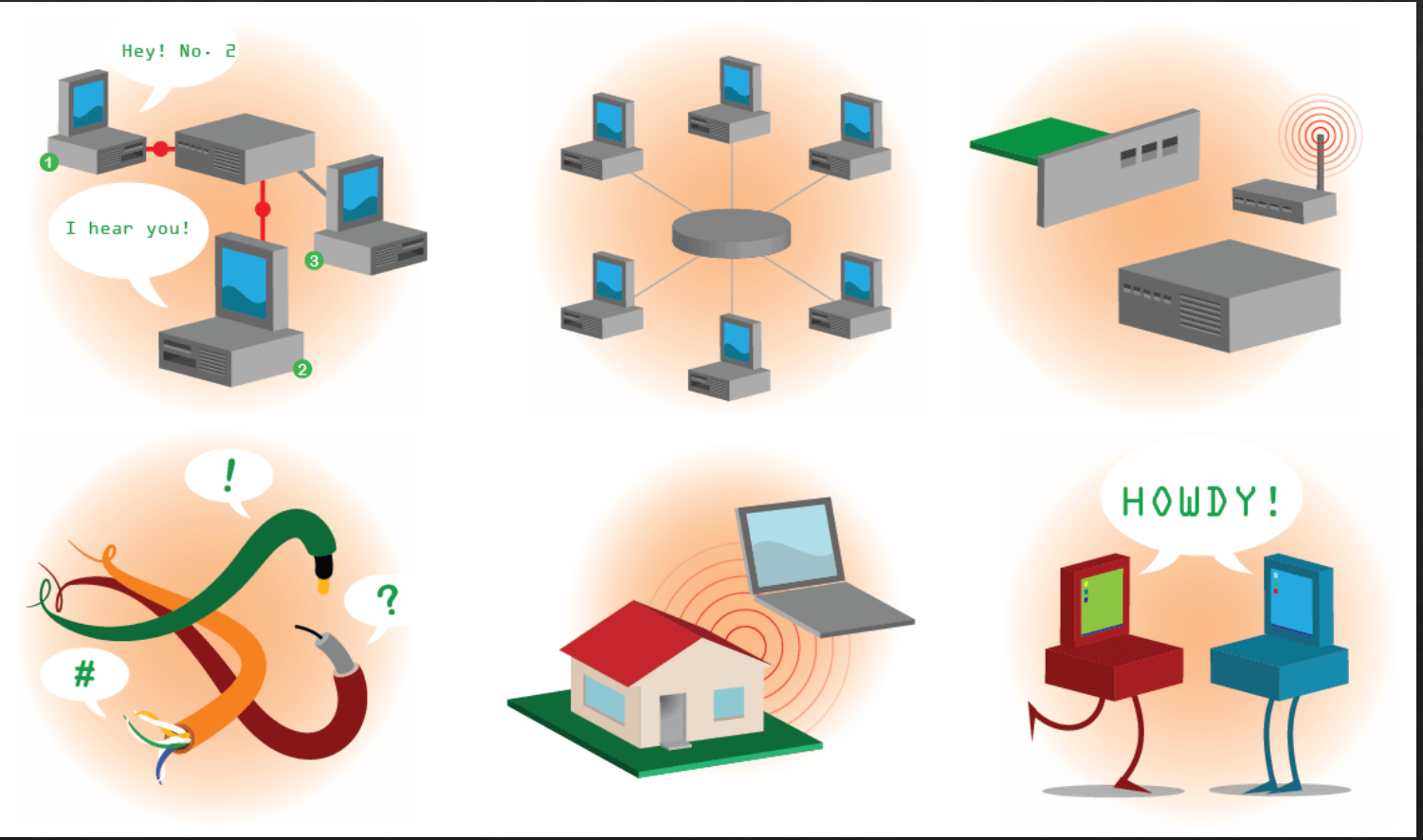
Laravel

3. Testning & deployment

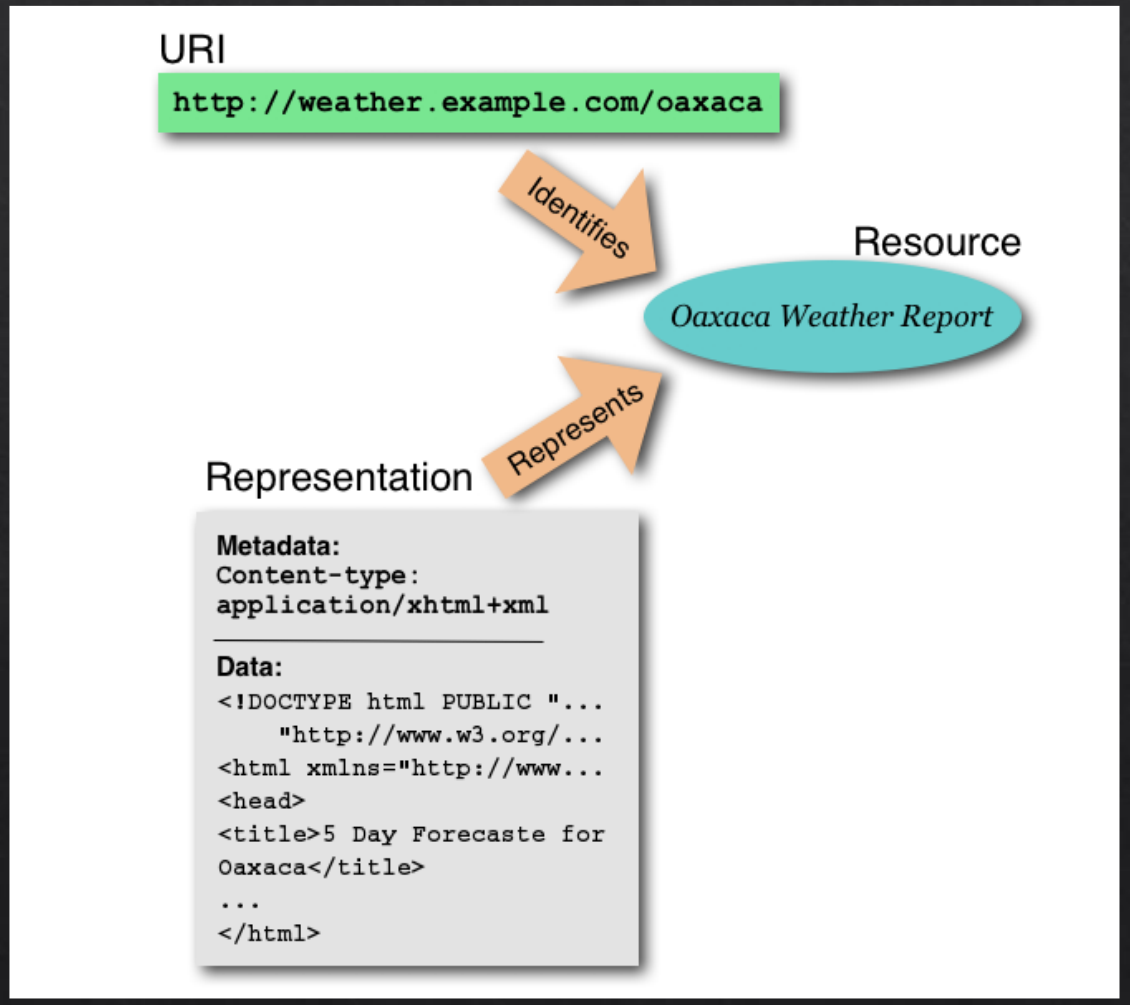
Webbtjänster (Web services)

- Webbtjänster (engelska: Web services) betecknar **webbaserade tjänster** som **kommunicerar** och **samarbetar dynamiskt** med **andra webbtjänster** på samma vis som en människa kan surfa till olika webbsidor.

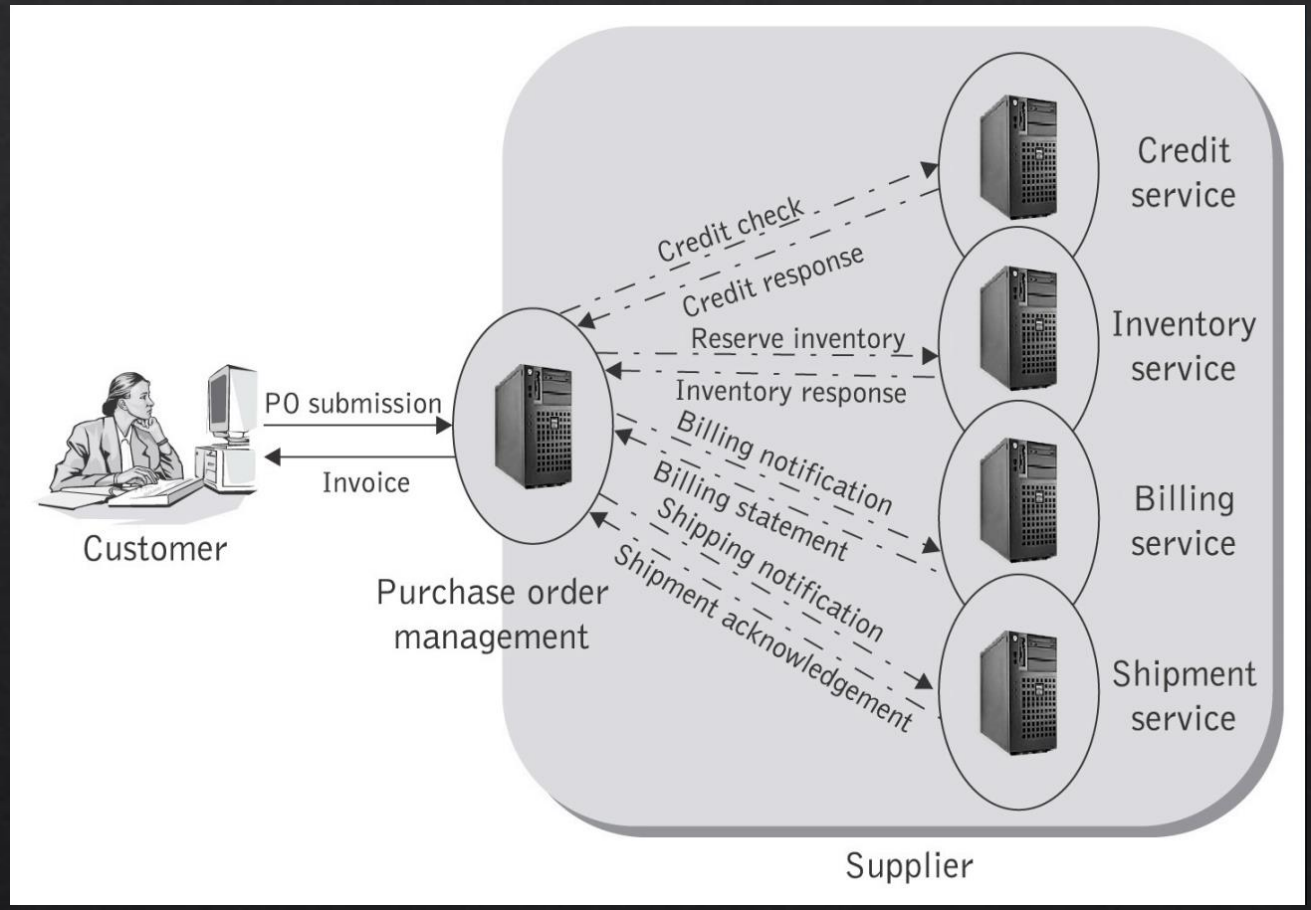
Perspektiv på WWW



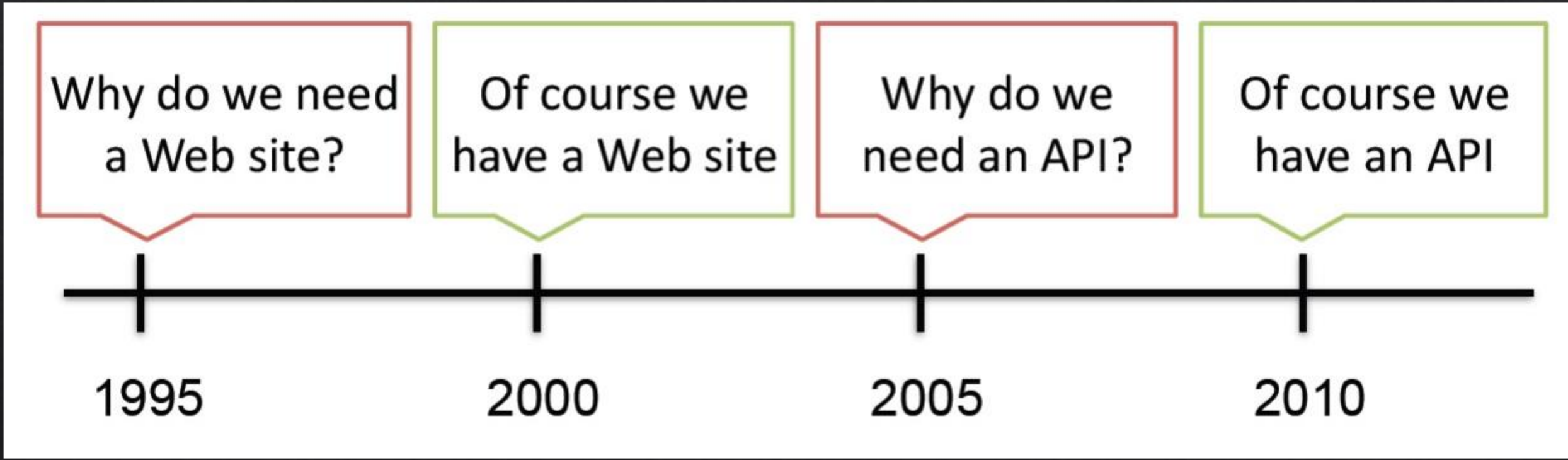
Resurser på webben

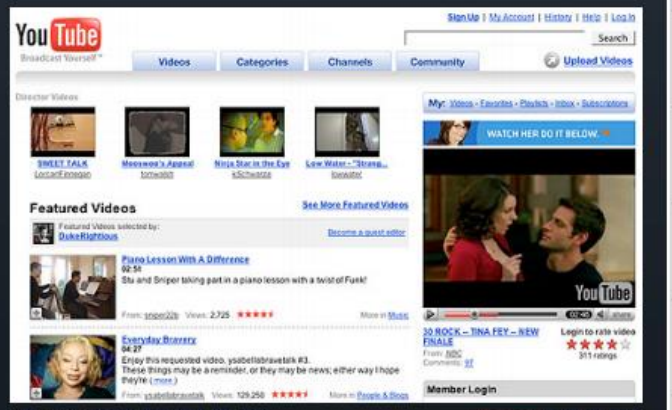


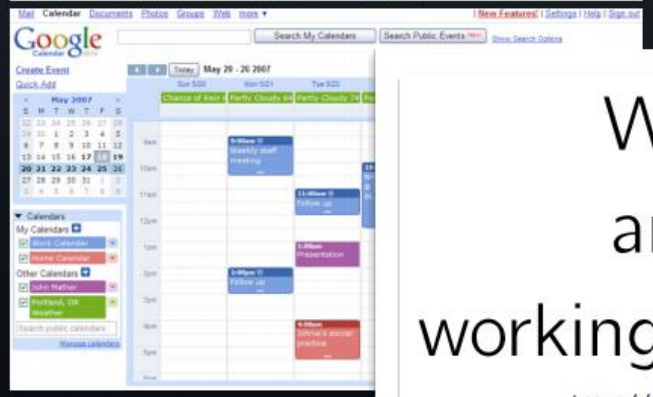
Service-orienterad arkitektur - Traditionella webbtjänster

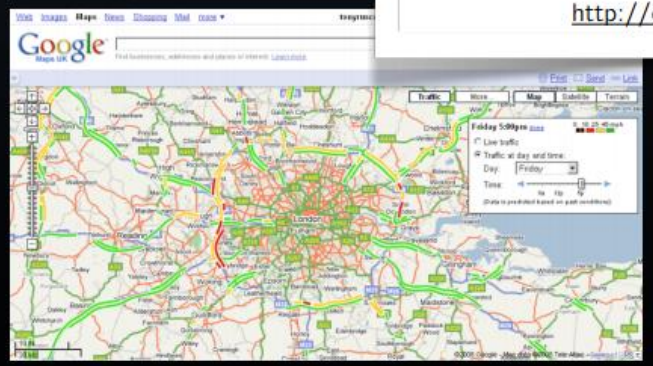


Webb API:r


Why do we need a Web site?
Of course we have a Web site
Why do we need an API?
Of course we have an API
1995
2000
2005
2010







Web applications
and web services
working with the same data.

http://code.google.com/apis/gdata/docs/directory.html

API Home	Guides	Client Libraries
Google Analytics Data Export API	Developer's Guide Reference Guide	Client Libraries and Sample (JS, Java, PHP, Python, Ru
Google Apps APIs	List of All Apps APIs	
Google Base Data API	Developer's Guide Reference Guide	
Blogger Data API	Developer's Guide Reference Guide	Client Libraries and Sample (Java, .NET, PHP, Python,
Google Booksearch Data API	Developer's Guide Reference Guide	Client Libraries and Sample (Java, PHP)
	s Guide Guide	Client Libraries and Sample (Java, .NET, PHP, Python,
	s Guide Guide	
	s Guide Guide	Client Libraries and Sample (Java, .NET, Python, JS, O
	s Guide Guide	Client Libraries and Sample (Java, .NET, PHP, Python,
Google Finance Portfolio Data API	Developer's Guide Reference Guide	
Google Health Data API	Developer's Guide Reference Guide	Client Libraries and Sample (Java, .NET, PHP, Python,
Google Maps Data API	Developer's Guide Reference Guide	
Picasa Web Albums Data API	Developer's Guide Reference Guide	Client Libraries and Sample (Java, .NET, PHP, Python,
Google Project Hosting Issue Tracker API	Reference Guide	

Back to WordPress.com

Developer Resources

Create cool applications that integrate with WordPress.com

Documentation Blog My Applications Contact

Search ...

Documentation → REST API Resources

Rest API

Welcome to the REST API. Below, you'll find a full listing of all the available endpoints. As we add more endpoints, they will be automatically documented here and available through the Developer Console.

For more information about a particular endpoint, click on its name under the Resource header. You'll be taken to the endpoint's documentation page, which includes what query parameters the endpoint will except, what the JSON object's parameters will be in the response, and an example query/response.

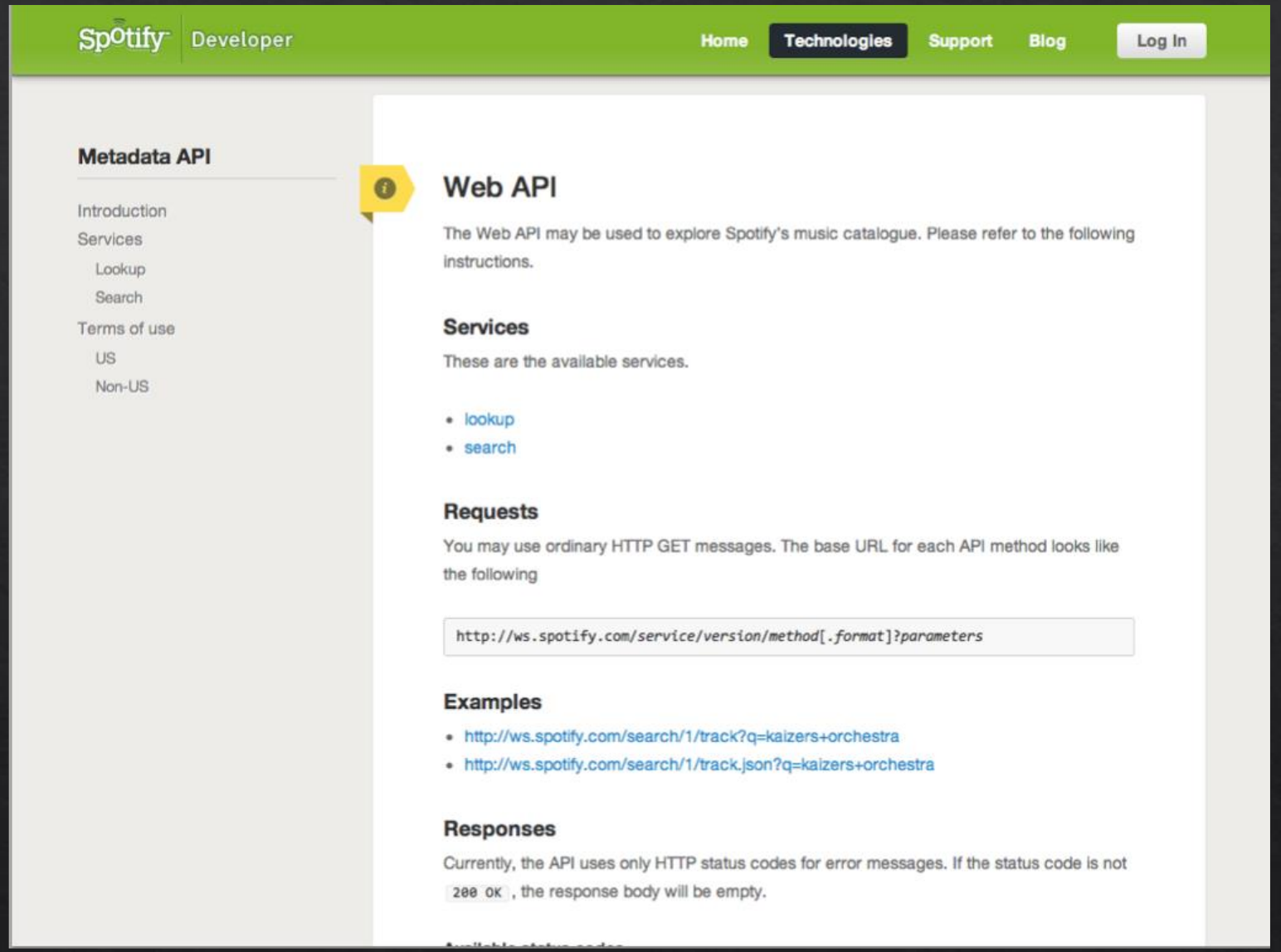
Users

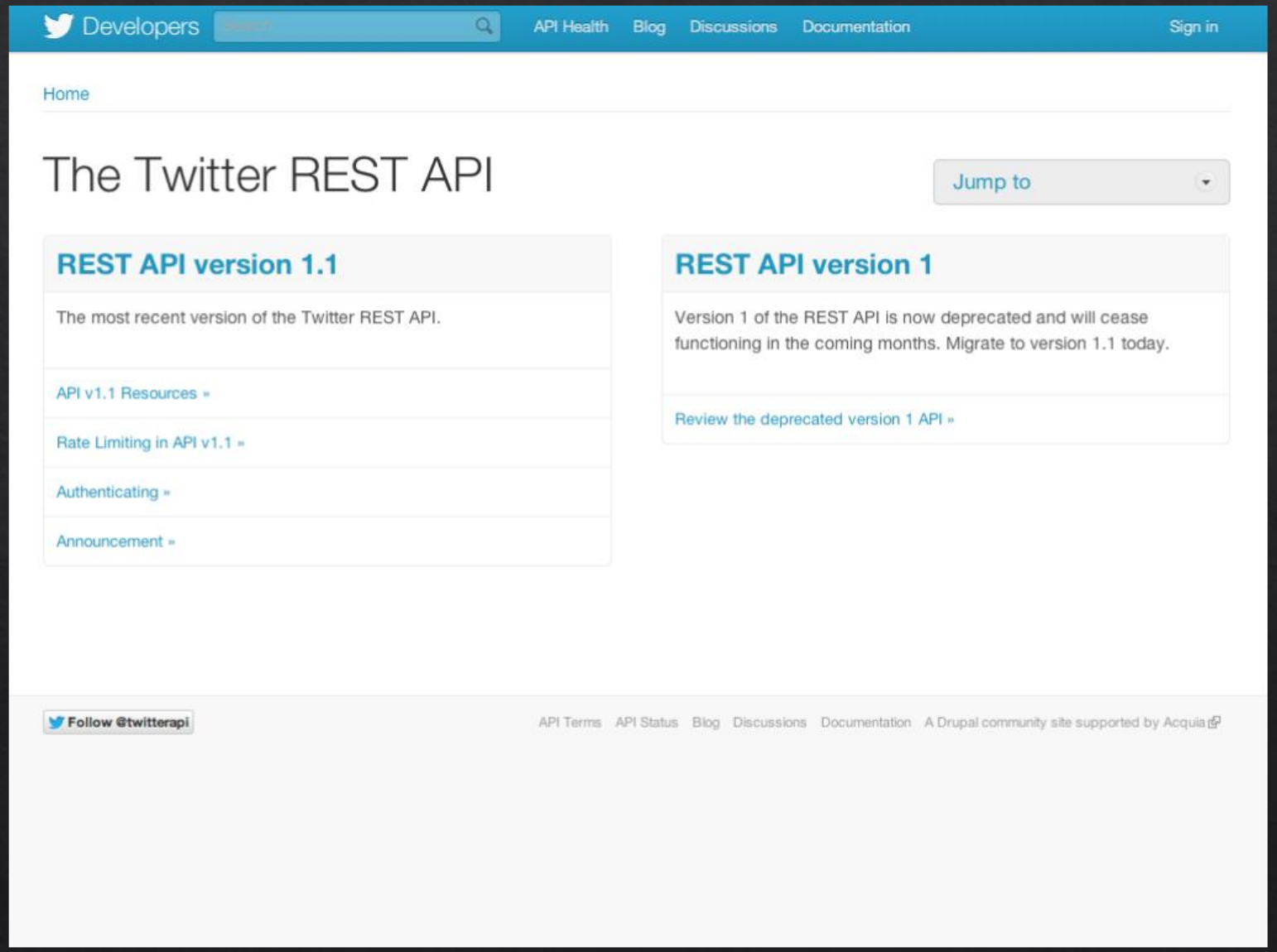
User information including profile data.

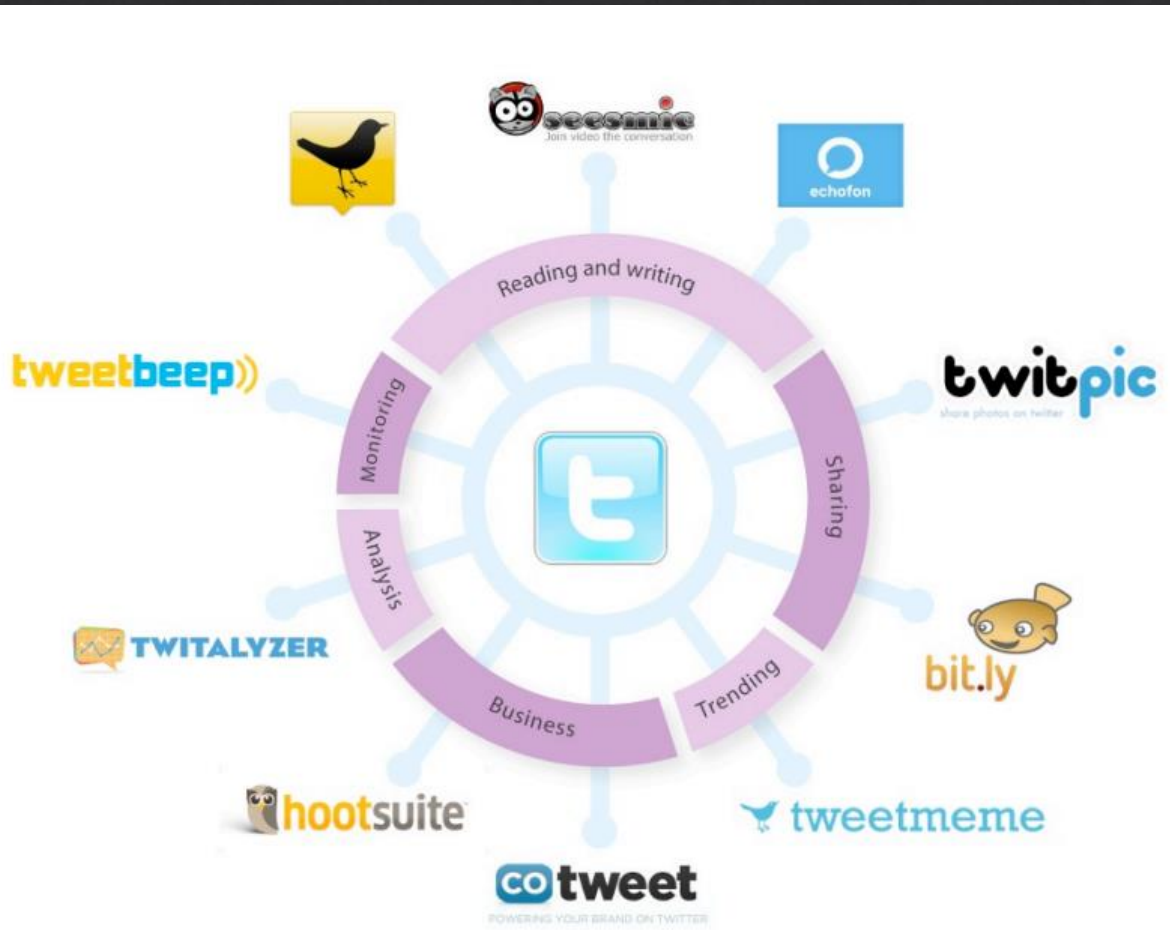
Resource	Description
GET /me	Meta data about auth token's User

Sites

Metadata on a blog


Spotify Developer
Home
Technologies
Support
Blog
Log In
Metadata API
Introduction
Services
Lookup
Search
Terms of use
US
Non-US
Web API
The Web API may be used to explore Spotify's music catalogue. Please refer to the following instructions.
Services
These are the available services.
lookup
search
Requests
You may use ordinary HTTP GET messages. The base URL for each API method looks like the following
http://ws.spotify.com/service/version/method[.format]?parameters
Examples
http://ws.spotify.com/search/1/track?q=kaizers+orchestra
http://ws.spotify.com/search/1/track.json?q=kaizers+orchestra
Responses
Currently, the API uses only HTTP status codes for error messages. If the status code is not 200 OK, the response body will be empty.


Developers
API Health
Blog
Discussions
Documentation
Sign in
Home
The Twitter REST API
Jump to
REST API version 1.1
The most recent version of the Twitter REST API.
API v1.1 Resources »
Rate Limiting in API v1.1 »
Authenticating »
Announcement »
REST API version 1
Version 1 of the REST API is now deprecated and will cease functioning in the coming months. Migrate to version 1.1 today.
Review the deprecated version 1 API »
Follow @twitterapi
API Terms
API Status
Blog
Discussions
Documentation
A Drupal community site supported by Acquia


seesmic
Join video the conversation
echofon
Reading and writing
tweetbeep
twitpic
share photos on twitter
Monitoring
Sharing
Analysis
TWITALYZER
bit.ly
Business
Trending
hootsuite
tweetmeme
cotweet
POWERING YOUR BRAND ON TWITTER

“


Siri
OpenTable
Taxi Magic
NUANCE
Citysearch
movie tickets.com
eventful
Gayot
livekick
WeatherBug
BeatRah
Rotten Tomatoes
allmenus.com
YAHOO! LOCAL
yelp
Google maps
FLIGHTSTATS
twitter
True Knowledge
StubHub!
maponics

”

REST

Representational State Transfer

“

[The] Web's major goal was to be a shared information space through which people and machines could communicate.

”

- Tim Berners-Lee

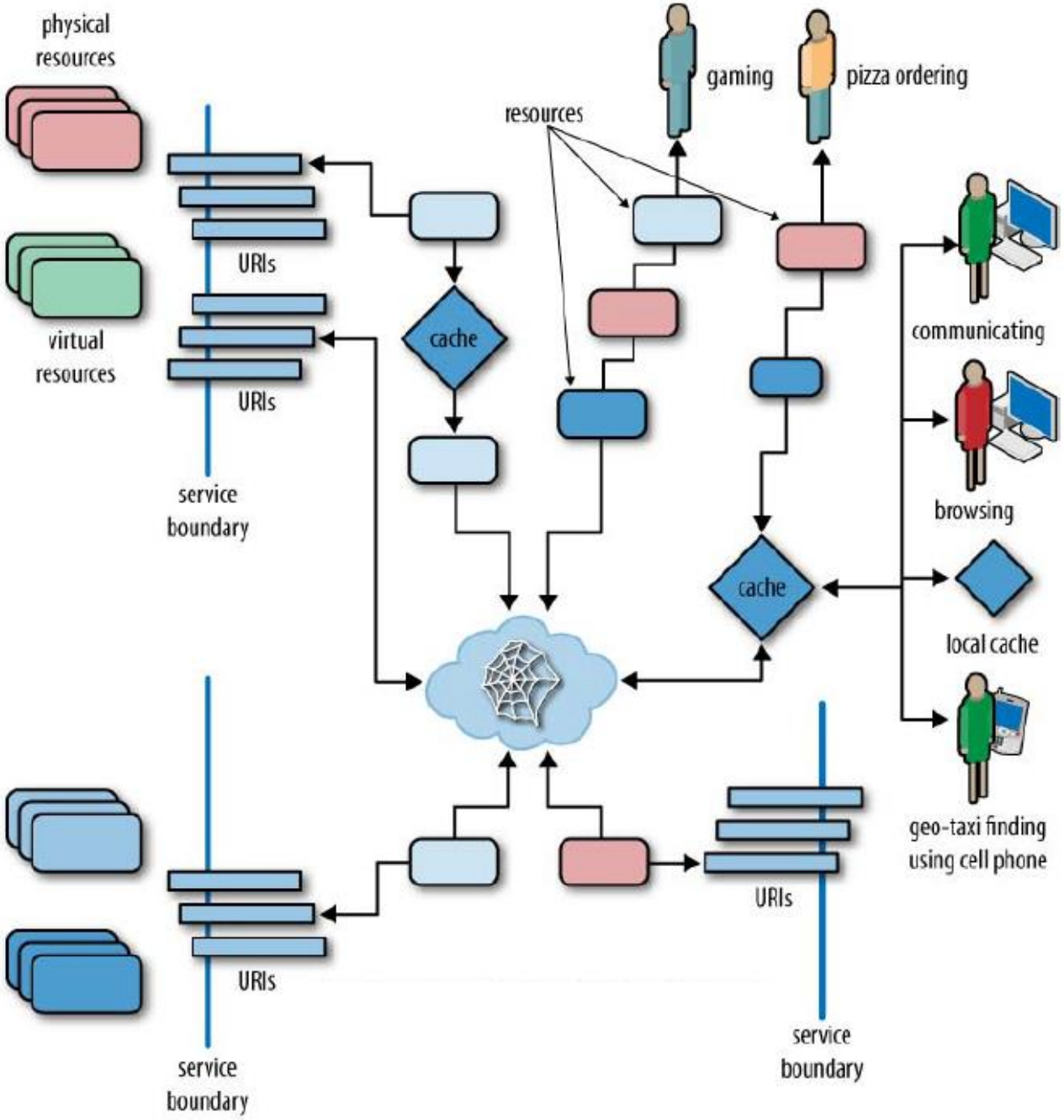
Webbarkitekturer - REST

Att bygga restful services

Resurser
Identifierare

Representationer

HTTP



Grundidén bakom REST

- **Resurser**, snarare än tjänster, och som identifieras genom logiska URL
- **En webb av resurser**, ett svar behöver inte ge svar på *allt* utan kan länka vidare till andra resurser
- **Klient – Server**-modell
- **Tillståndslöst (stateless)**, varje anrop står på egna ben
- **Cachningsbara resurser**

Resurser

Vad är en resurs?

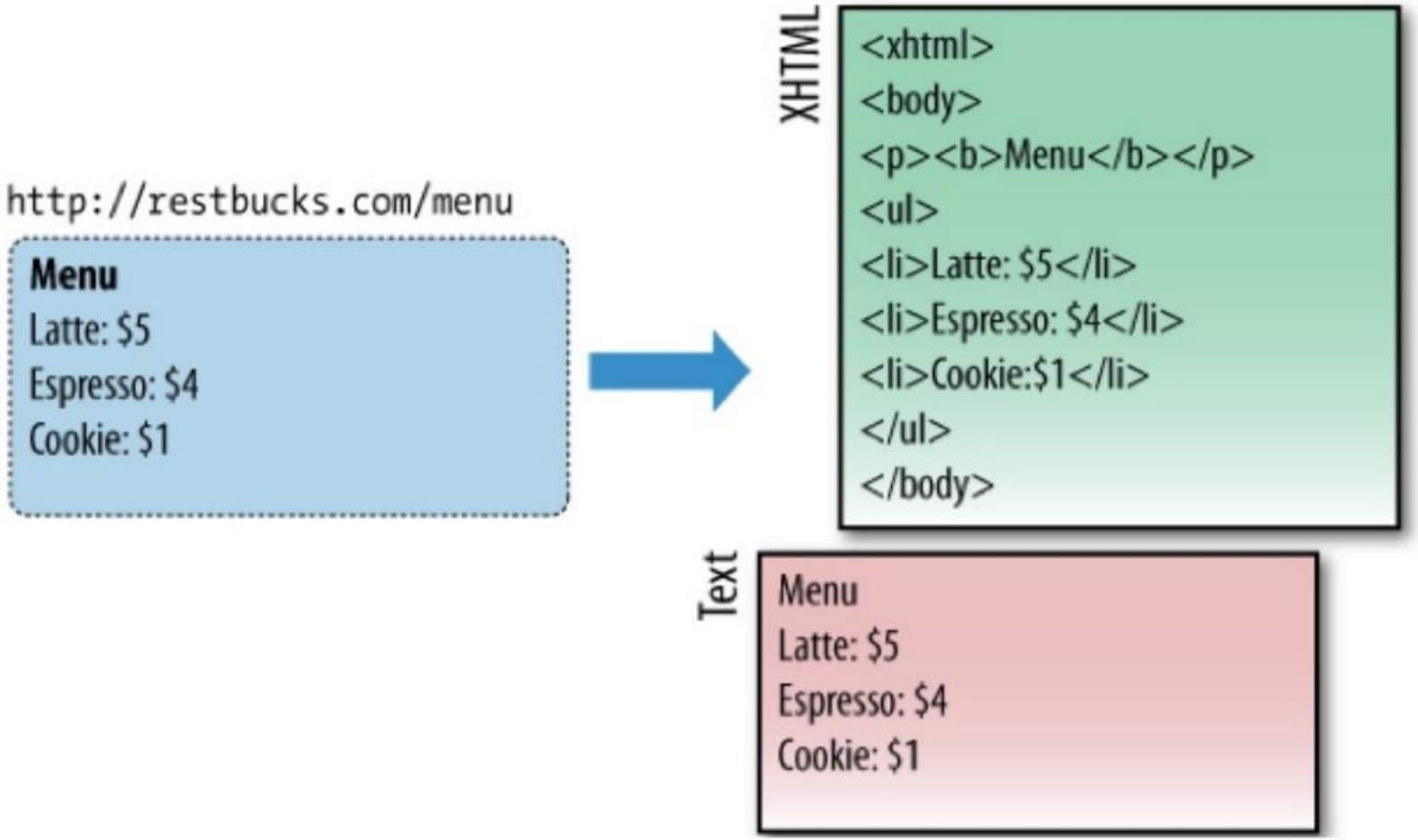
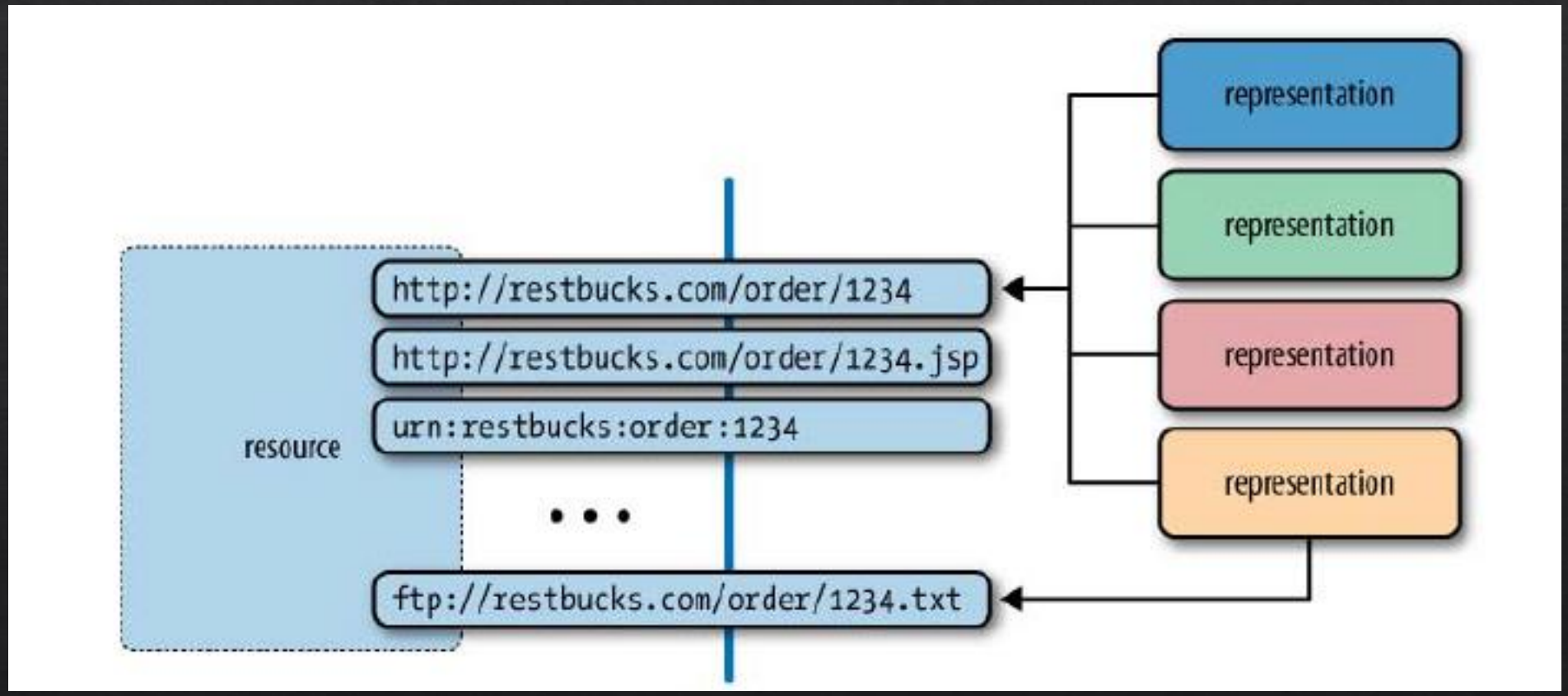


Figure 1-3. *Example of a resource and its representations*


representation
representation
representation
representation
http://restbucks.com/order/1234
http://restbucks.com/order/1234.jsp
urn:restbucks:order:1234
resource
• • •
ftp://restbucks.com/order/1234.txt

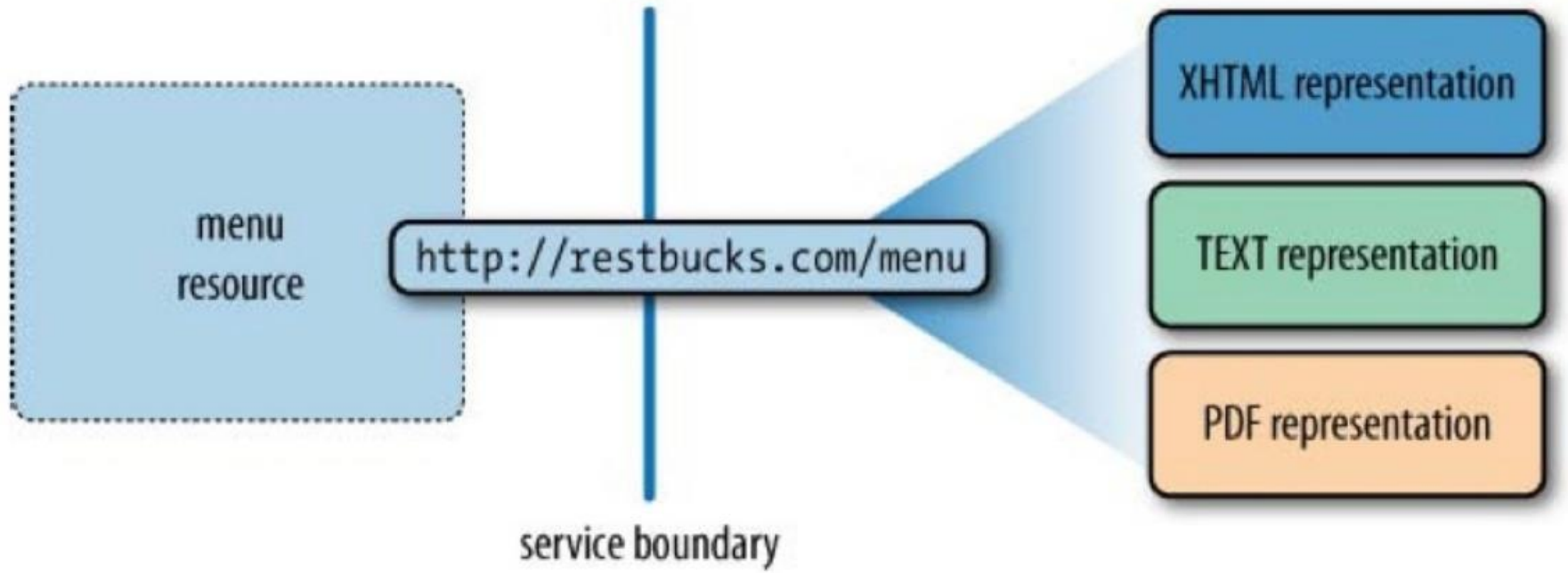


Figure 1-5. *Multiple resource representations addressed by a single URI*

Identifierare

URI	Uniform Resource Identifier
URL	Uniform Resource Locator

URI

<scheme>:<scheme-specific-structure>

HTTP Scheme URI (=an URL)

http://example.com/documents/report.txt

mailto Scheme URI (=not an URL)

mailto:anton.tibblin@mah.se

HTTP URL

The "http" scheme is used to locate network resources via the HTTP protocol. This section defines the schemespecific syntax and semantics for http URLs.

http_URL = "http:" "//" host [":" port] [abs_path ["?" query]]

REST och representation

- Fokus i REST är resurser och hur man når dessa resurser
- En resurs är kan ofta jämföras med ett objekt
  - En resurs kan bestå av andra resurser
  - När man designar en RESTful tjänst är det därför viktigt att se vilka resurser som finns och hur dessa relaterar till varandra.
- Representationen av resursen är inte beroende av ett specifikt format
  - Man kan använda flera format!

```
{
    "ID": "1",
    "Name": "M Vaqqas",
    "Email": "m.vaqqas@gmail.com",
    "Country": "India"
}
```

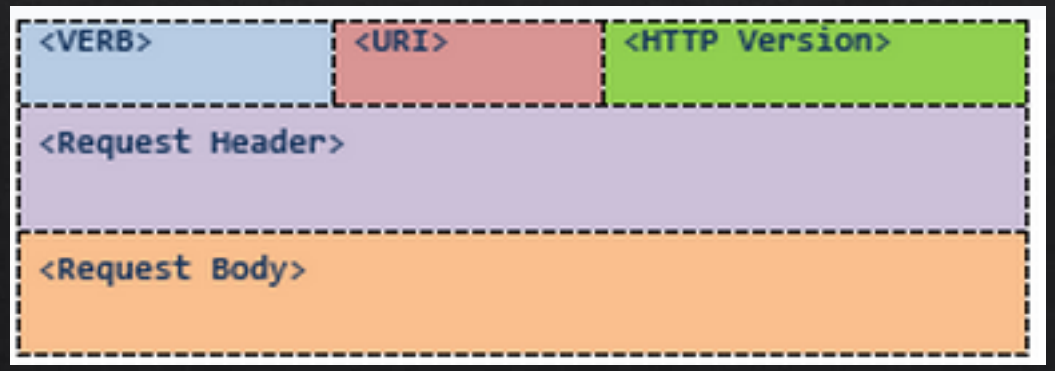
```
<Person>

<ID>1</ID>

<Name>M Vaqqas</Name>

<Email>m.vaqqas@gmail.com</Email>

<Country>India</Country>
</Person>
```

Flera format

- Om man vill ha möjligheten att returnera flera format kan man helt enkelt ge klienten de möjligheterna, t.ex. genom
  - Querys (söksträngar i URL)
  - Eller genom i anropet ange ACCEPT – vilket svar man accepterar från anropet

REST - Meddelanden

- Klient och Server pratar med varandra genom meddelanden
- Klienten skickar en förfrågan till servern genom
  - Data skickas
  - Metadata skickas
- Metadata skickas med HTTP



Adresser till en resurs

- Använd substantiv i plural, t.ex. ”persons”
- Använd inte mellanslag, använd istället _ eller –
- Tänk på att URI skijer på versalen och gemener
- Man kan ha egna konventioner i sin tjänst – men var konsekvent!
- Akta dig för att ändra URI:s
- Använd inte verb som resursnamn, t.ex. ”fetchPersons”

Adresser till en resurs (2)

- Använd inte query-parameter för id:n
  - persons?id=1
  - Använd istället: persons/1
    - Detta gör tjänsten enklare att läsa URL
    - Detta gör att sökmotorer indexerar sidan
- Vill man ha parametrar som inte beskriver en resurs, anända query-parametrar
  - Persons/1?format=json&encoding=UTF-8
- Query-parametrar kan vara frivilliga, till skillnad från URI

HTTP – verb

- Vi har
  - GET – Läser en resurs
  - POST – Skapar en ny resurs
  - PUT – Skapar en ny resurs, eller uppdaterar en befintlig resurs
  - DELETE – Raderar en resurs
  - OPTIONS – Listar godkända operationer för en resurs
  - HEAD – Returnerar bara huvudet av svarat (bara headers, inte body)

Method	Operation performed on server
GET	Read a resource.
PUT	Insert a new resource or update if the resource already exists.
POST	Insert a new resource. Also can be used to update an existing resource.
DELETE	Delete a resource .
OPTIONS	List the allowed operations on a resource.
HEAD	Return only the response headers and no response body.

REST – design guidelines

- Använd **inte** en fysik adress (som pekar på en fil), t.ex.
  - http://example.com/person/1.xml
- Använd istället en logisk URL för att nå en resurs
  - http://example.com/person/1
- Returnera rimligt med resultat
  - *n* antal svar, eller dela in i ”sidor”
  - Länka vidare till övriga resurser
- Dokumentera tydligt vad klienten kan förvänta sig för svar
- Istället för att returnera ID för t.ex. alla personer i exemplet ovan – returnera URL för personerna
- Använd **aldrig** GET-metoden för att ändra ett tillstånd för en resurs

Vem använder REST?

- Twitter
- Flickr
- Wordpress
- Spotify
- Youtube
- Etc.

https://lumen.laravel.com/

Lumen 5

Lumen.

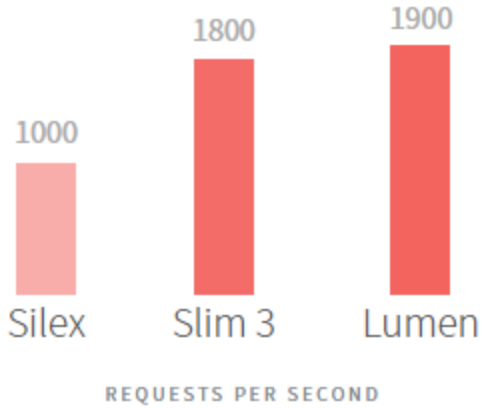
The stunningly fast micro-framework by Laravel.

```php
<?php

/**
 * Reimagine what you expect...
 */
$app->get('/', function() {
    return ['version' => '5.3']
});

/**
 * From your micro-framework...
 */
$app->post('framework/{id}', function($framework) {

```

Lightning fast micro-services and APIs delivered with the elegance you expect.

Benchmark Breaking Speed

Lumen is the perfect solution for building Laravel based micro-services and blazing fast APIs. In fact, it's one of the fastest micro-frameworks available. It has never been easier to write stunningly fast services to support your Laravel applications.




```php
<?php

$app->get('user/{id}', function($id) {
    return User::findOrFail($id);
});
```

The Convenience You Love

Don't sacrifice power for speed. Use the Laravel features you love like Eloquent, caching, queues, validation, routing, middleware, and the powerful Laravel service container. All with almost zero configuration.

Friends In High Places

Have a Lumen project you want to upgrade to the full Laravel framework? It couldn't be easier. Since Lumen is powered by

Mikroramverk

Routing

Templates

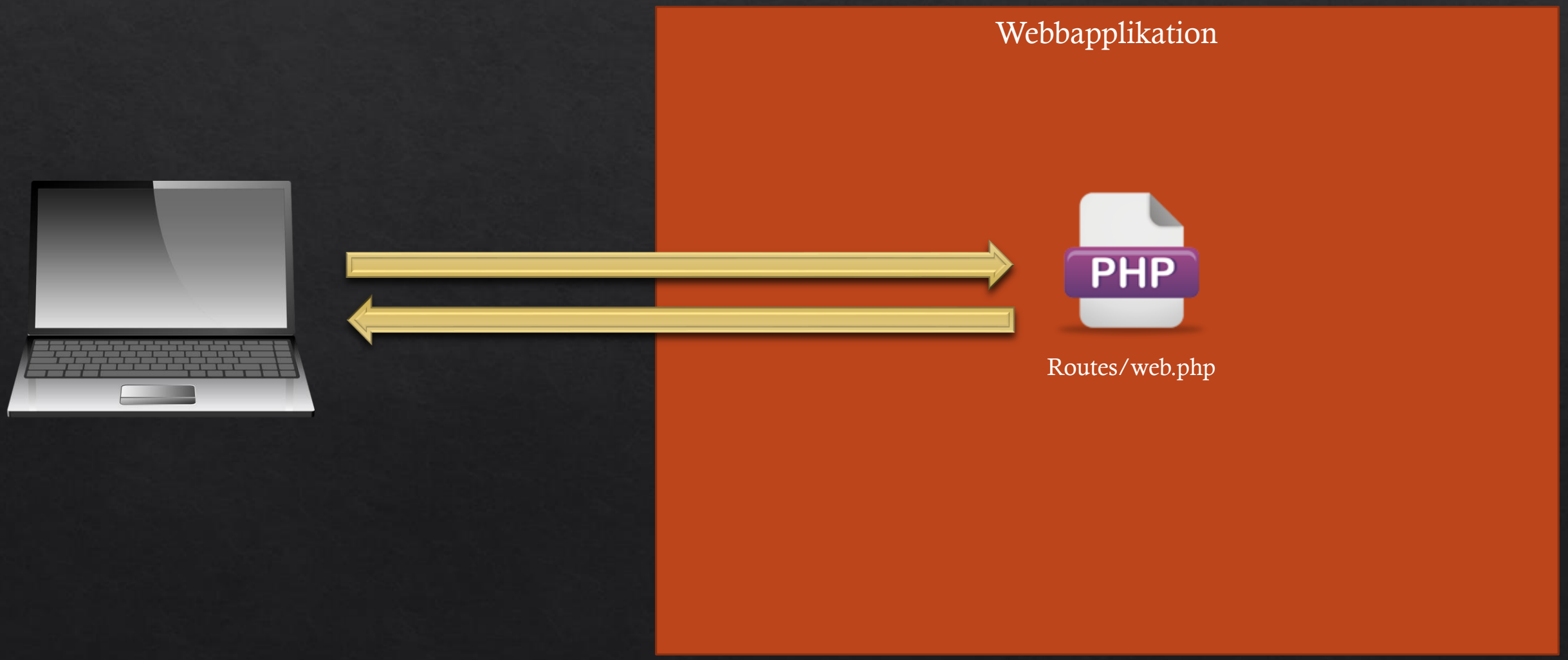
ORM

Authentiering

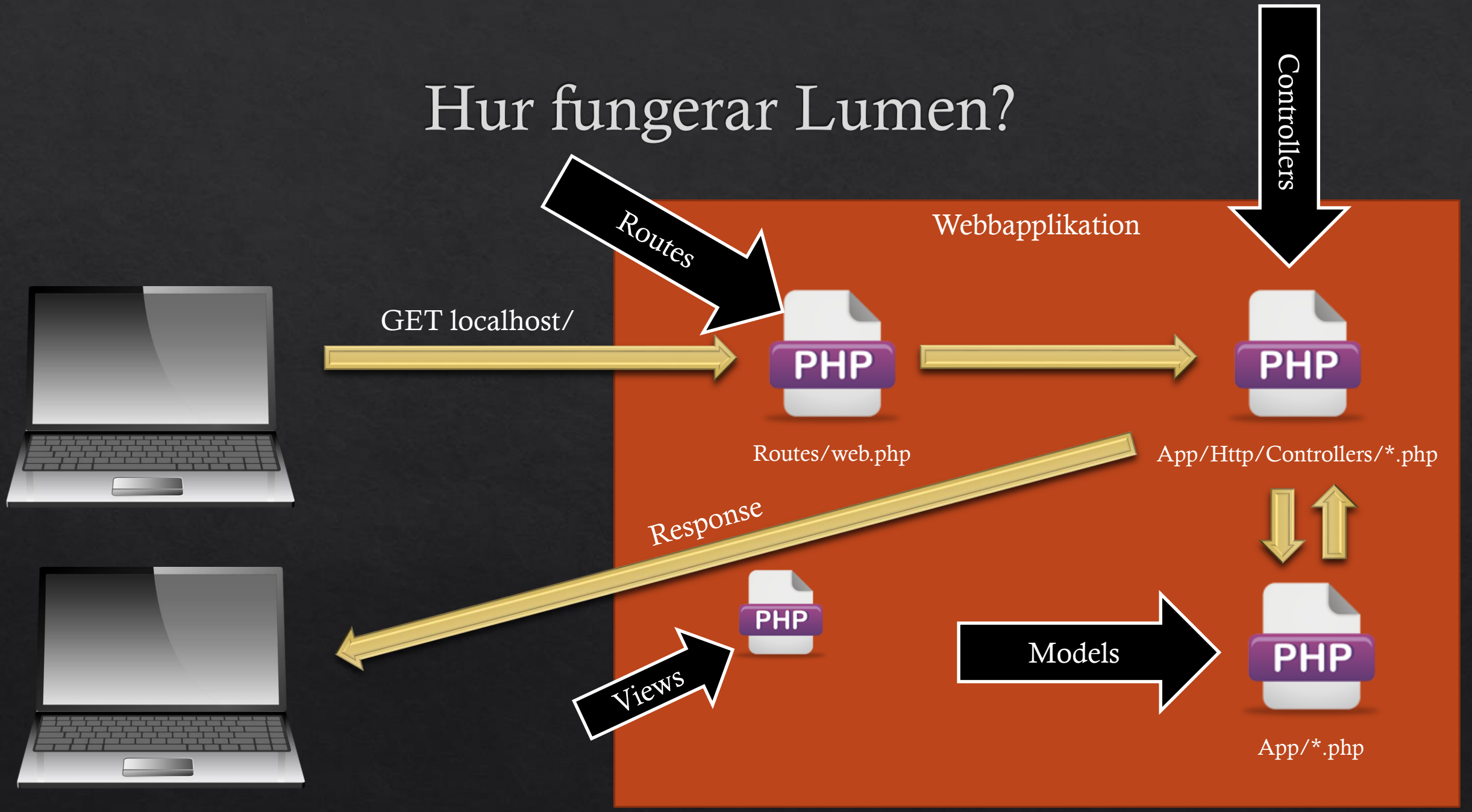
Middleware

m.m.

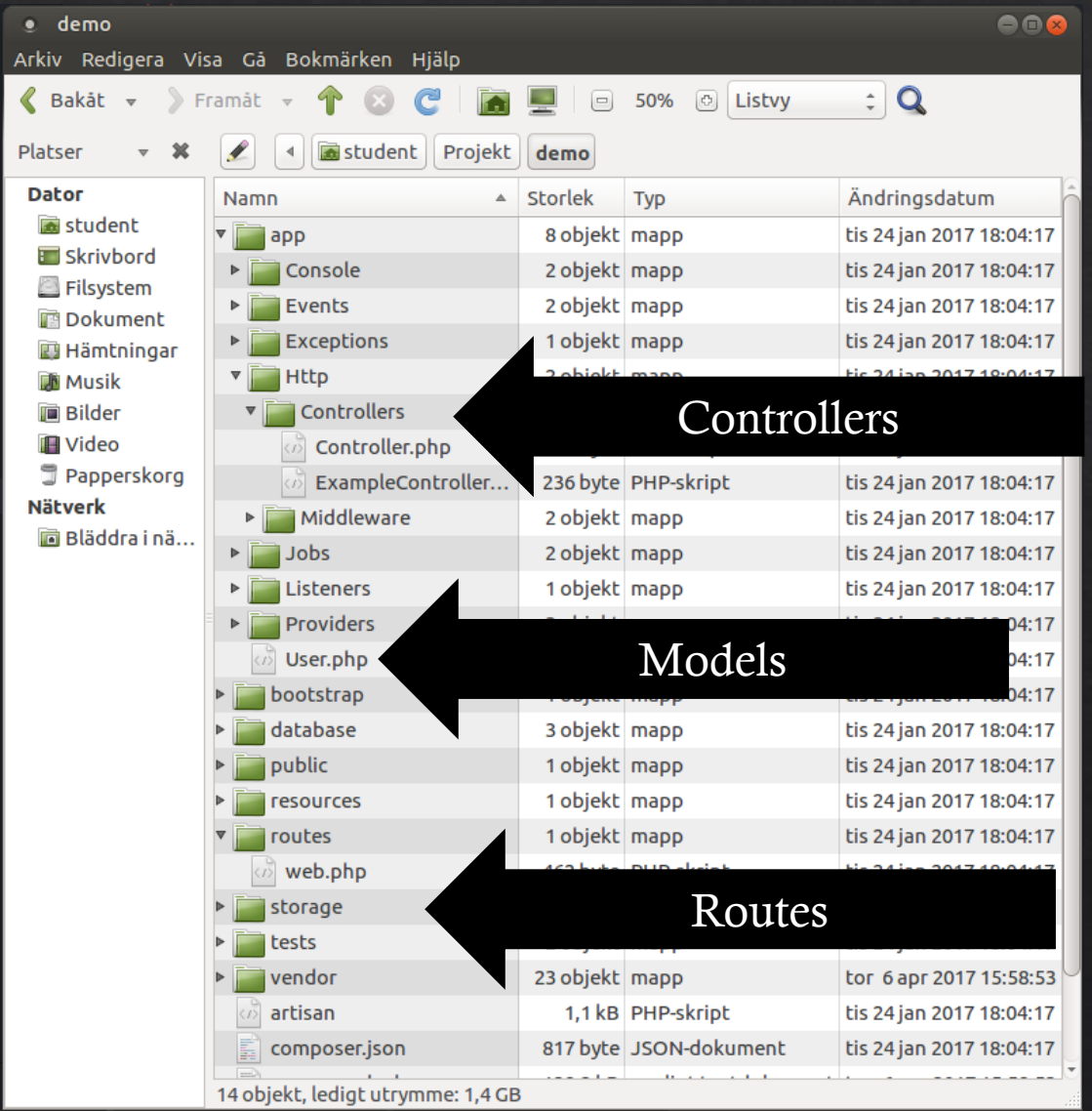
Hur fungerar Lumen?

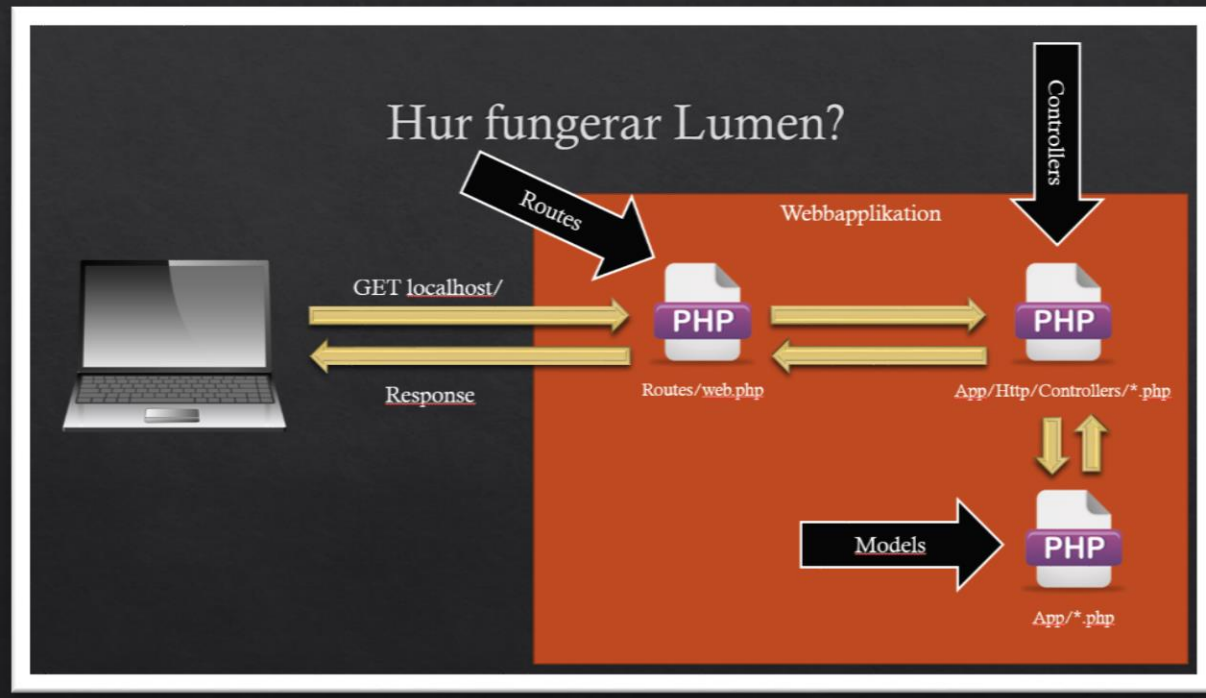


Hur fungerar Lumen?



Hur fungerar Lumen? (2)





Viktiga koncept att kunna inom Lumen!

- Routing
- Controllers
- Models
- (Middleware)
- (Authentiation)
- (ORM)
- (m.m.)

Routes (Routes/web.php)

Basic Routing

You will define all of the routes for your application in the `routes/web.php` file. The most basic Lumen routes simply accept a URI and a `Closure`:

```
$app->get('foo', function () {
    return 'Hello World';
});

$app->post('foo', function () {
    //
});
```

https://lumen.laravel.com/docs/5.4/routing

Routes (Routes/web.php)

Available Router Methods

The router allows you to register routes that respond to any HTTP verb:

```
$app->get($uri, $callback);
$app->post($uri, $callback);
$app->put($uri, $callback);
$app->patch($uri, $callback);
$app->delete($uri, $callback);
$app->options($uri, $callback);
```

https://lumen.laravel.com/docs/5.4/routing

Route Parameters

Required Parameters

Of course, sometimes you will need to capture segments of the URI within your route. For example, you may need to capture a user's ID from the URL. You may do so by defining route parameters:

```
$app->get('user/{id}', function ($id) {
    return 'User '.$id;
});
```

You may define as many route parameters as required by your route:

```
$app->get('posts/{post}/comments/{comment}', function ($postId, $commentId) {
    //
});
```

Route parameters are always encased within "curly" braces. The parameters will be passed into your route's `Closure` when the route is executed.

Note: Route parameters cannot contain the `-` character. Use an underscore (`_`) instead.

Mer avancerade routes

Regular Expression Constraints

You may constrain the format of your route parameters by defining a regular expression in your route definition:

```
$app->get('user/{name:[A-Za-z]+}', function ($name) {
    //
});
```

https://lumen.laravel.com/docs/5.4/routing

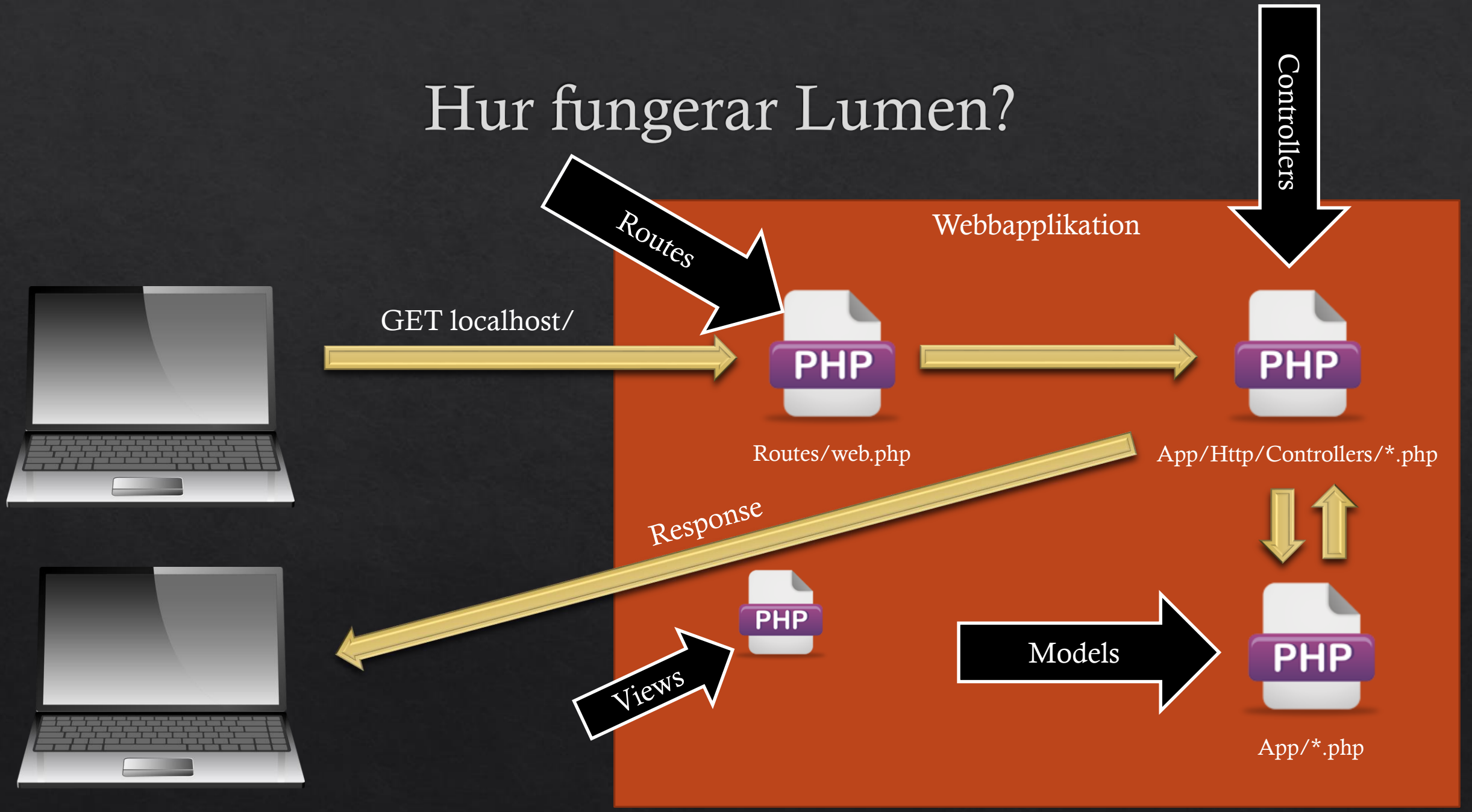
Middleware

Middleware

To assign middleware to all routes within a group, you may use the `middleware` key in the group attribute array. Middleware will be executed in the order you define this array:

```
$app->group(['middleware' => 'auth'], function () use ($app) {
    $app->get('/', function ()    {
        // Uses Auth Middleware
    });

    $app->get('user/profile', function () {
        // Uses Auth Middleware
    });
});
```

Hur fungerar Lumen?



Controllers

Basic Controllers

Here is an example of a basic controller class. All Lumen controllers should extend the base controller class included with the default Lumen installation:

```php
<?php

namespace App\Http\Controllers;

use App\User;

class UserController extends Controller
{
    /**
     * Retrieve the user for the given ID.
     *
     * @param  int  $id
     * @return Response
     */
    public function show($id)
    {
        return User::findOrFail($id);
    }
}
```

En filmsamling?