



WEBINAR

# Single Page Apps with Grails & Vue.js

Zachary Klein, Senior Software Engineer

## Speaker


---



Zachary Klein is a Senior Software Engineer at OCI. He has been practicing web development since 2010 and frontend development since 2015. He's a contributor to both the Grails and Micronaut frameworks, a conference speaker and an instructor in OCI's training practice. Zachary's home base is in St Louis, MO, along with his wife, Beth, and their three children.



# Grails Application Forge: <http://start.grails.org>



**GRAILS APPLICATION FORGE**

**Project Type:**  
Application

**Name your application:**  
myapp

**Version:**  
3.3.5

**Profile:**  
web


**Pick your Features:**

- ☒ asset-pipeline
- ☒ events
- ☒ geb
- ☐ geb2
- ☒ gsp
- ☐ hibernate4
- ☒ hibernate5
- ☐ json-views
- ☐ less-asset-pipeline
- ☐ markup-views
- ☐ mongodb
- ☐ neo4j
- ☐ rx-mongodb

**Generate**

# Grails Guides: <http://guides.grails.org>

GUIDES



LATEST GUIDES

TWITTER OAUTH WITH GAILS 3 AND SPRING SECURITY REST  
Apr 30, 2018 - Advanced Grails  
[Read More](#)

GORM LOGICAL DELETE  
Apr 09, 2018 - GORM  
[Read More](#)

JAVAMELODY MONITORING WITH GAILS 3  
Apr 02, 2018 - Grails + Devops  
[Read More](#)

BUILDING A VUE.JS APP WITH GAILS  
Mar 26, 2018 - Grails + Vue.js  
[Read More](#)

SEARCH

GRAILS TRAINING

Course	Date(s)	Instructor(s)	Hour(s)
<a href="#">Micronaut Deep Dive at GR8Conf US</a>	Jul 23 - Jul 24	Brown	12
<a href="#">Introduction to Grails Security</a>	May 17 - May 18	del Amo Caballero	6
<a href="#">GROOVY 2.5 UPDATE</a>	May 18	King	1

# Introduction to Vue.js

# What is Vue?



- Small, performant view library
- Mature ecosystem
- Powerful devtools
- Active community
- vs Angular vs React...



# Getting Started with Vue

- Install vue-cli (v3):
  1. `npm install -g @vue/cli`
  2. `vue create my-project`



# Getting Started with Vue

---

```
<script src="https://unpkg.com/vue"></script>
```

```
<div id="app">  
  {{ text }}  
</div>
```

```
<script>
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    text: 'Hello World!'  
  }  
})
```

```
</script>
```



# The Vue Instance

---

- Ground zero for a Vue.js app
- Accepts an object containing the Vue **Instance definition**
- Most definitions shared between instance and **components**

## Vue Instance

```
new Vue({  
  el: '#app',  
  data: {...},  
  computed: {...},  
  methods: {...},  
  template: '<App/>'  
})
```

## Vue Component

```
export default {  
  name: 'component',  
  data () { return {...} },  
  methods: {...}  
}
```

# The Vue Instance

---

- `el` - Specifies the DOM element where the app will be rendered

```
new Vue({  
  el: '#app'  
})
```

```
<div id="app"></div>
```

## The Vue Instance

---

- **data** - An object describing the state of the app/component
  - An object describing the state of the app/component
  - Properties are subject to reactive rendering within the app/component (one-way & two-way)
  - New properties can be added, but will not be subject to reactive behavior
  - Components must use a **data()** function that returns state

# The Vue Instance

---

- **data** - An object for standalone Vue instances, function for components

Vue Instance

```
new Vue({  
  data: {  
    myValue: 3,  
    myObject: {  
      prop: 'abc'  
    }  
  }  
})
```

Vue Component

```
export default {  
  data () {  
    return {  
      myValue: 3,  
      myObject: {  
        prop: 'abc'  
      }  
    }  
  }  
}
```

## The Vue Instance

---

- **methods** - Arbitrary functions that can access/manipulate data, and be called from templates or other methods.
- **computed** - Functions that return dynamic (cacheable) values and are accessed as data variables.
- **Lifecycle Methods** - Functions that are called at specific points in the app/component lifecycle (e.g, `created`)

# Templates

---

```
<template>
  <div class="form">
    <div class="title cell">
      <label>Title</label>
      <input ref="bookTitle" v-model="book.title" type="text"/>
    </div>
    <div class="pages cell">
      <label>Pages</label>
      <input v-model="book.pages" type="text"/>
    </div>
    <div class="author cell">
      <label>Author</label>

      <select v-model="book.author">
        <option disabled selected value="">Choose Author</option>
        <option v-if="author !== null"
          v-bind:value="{ id: author.id }"
          v-for="author in authors">{{author.name}}</option>

      </select>
    </div>
    <div class="save cell">
      <button @click="submitNewBook()" >Add Book</button>
    </div>
  </div>
</template>
```

## Components (\*.vue files)

- Nested component hierarchy
- Each component renders either a template or returns createElement() calls (JSX is supported)
- Components typically defined as **Single File Component** with `<template>`, `<script>`, and `<style>` sections

```
<template>
  <div class="hello">
    <h1>{{ msg }}</h1>
  </div>
</template>
```

Template Section

```
<script>
export default {
  name: 'HelloWorld',
  data () {
    return {
      msg: 'Welcome to Your Vue.js App'
    }
  }
}
</script>
```

Script Section

```
<style scoped>
h1 {
  font-color: blue;
}
</style>
```

Style Section

# Demo



## Vue vs Angular

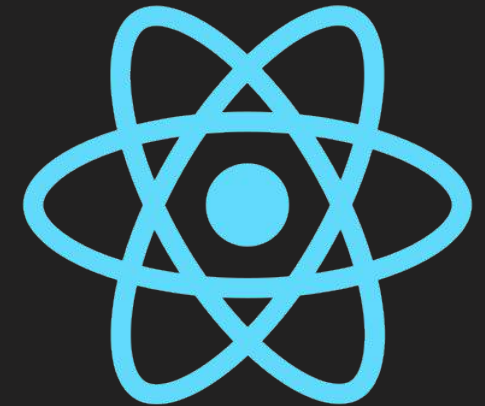
- A cleaner, simpler way to build apps
- Similar concepts/syntax: directives, templates, v-model
- More understandable, readable code
- A rich, featureful API that you can grow into
- Not a framework - you'll still need to add packages



## Vue vs React

---

- Adds a bit more magic
- Similar conceptually: components, virtual DOM, lifecycle methods
- Supports JSX, createElement API, render functions
- Less code... sometimes
- Favors pragmatism over “pure” simplicity



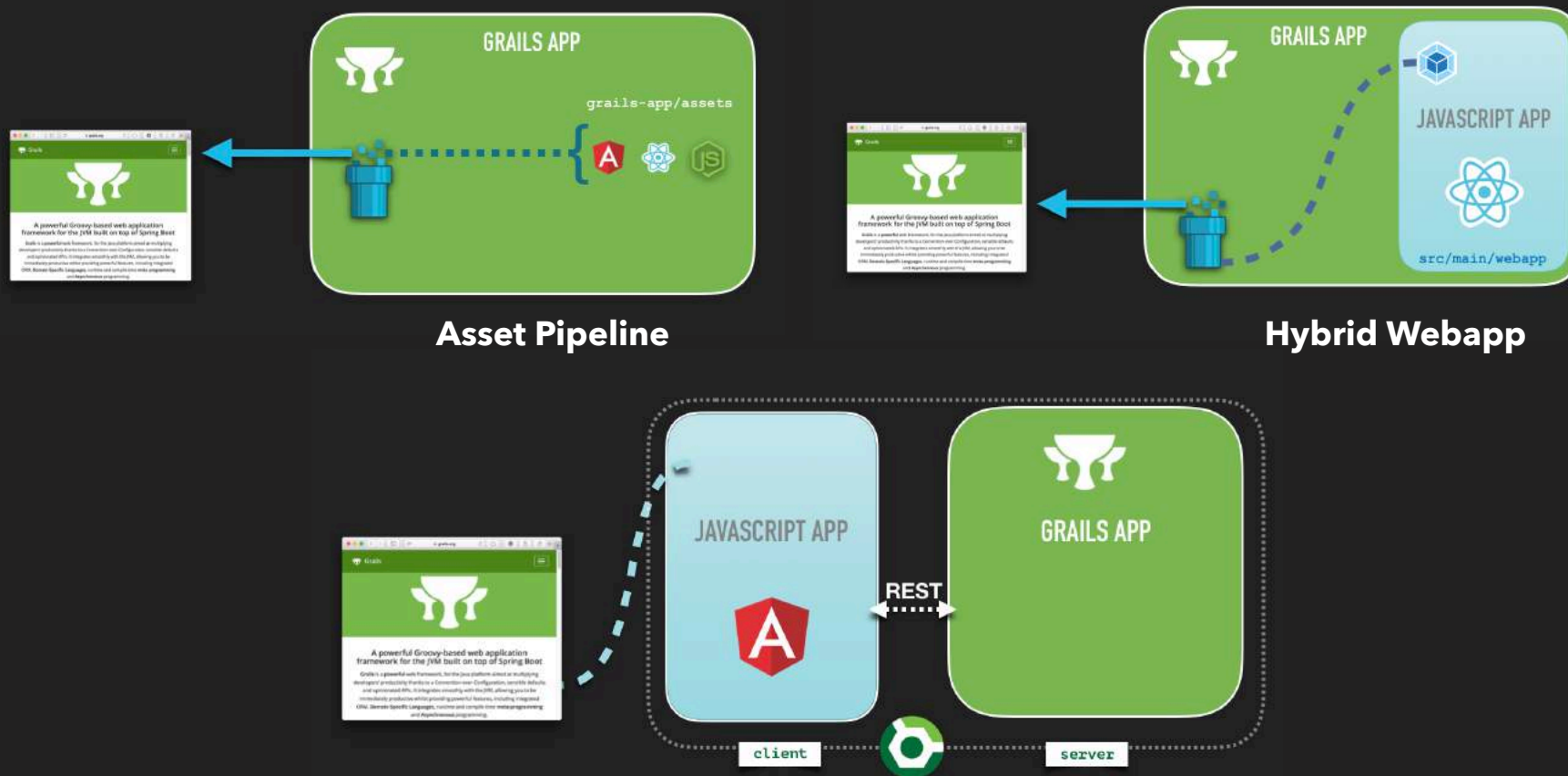
# Vue Profile for Grails

# Why Use Vue (and other JavaScript frameworks) with Grails?

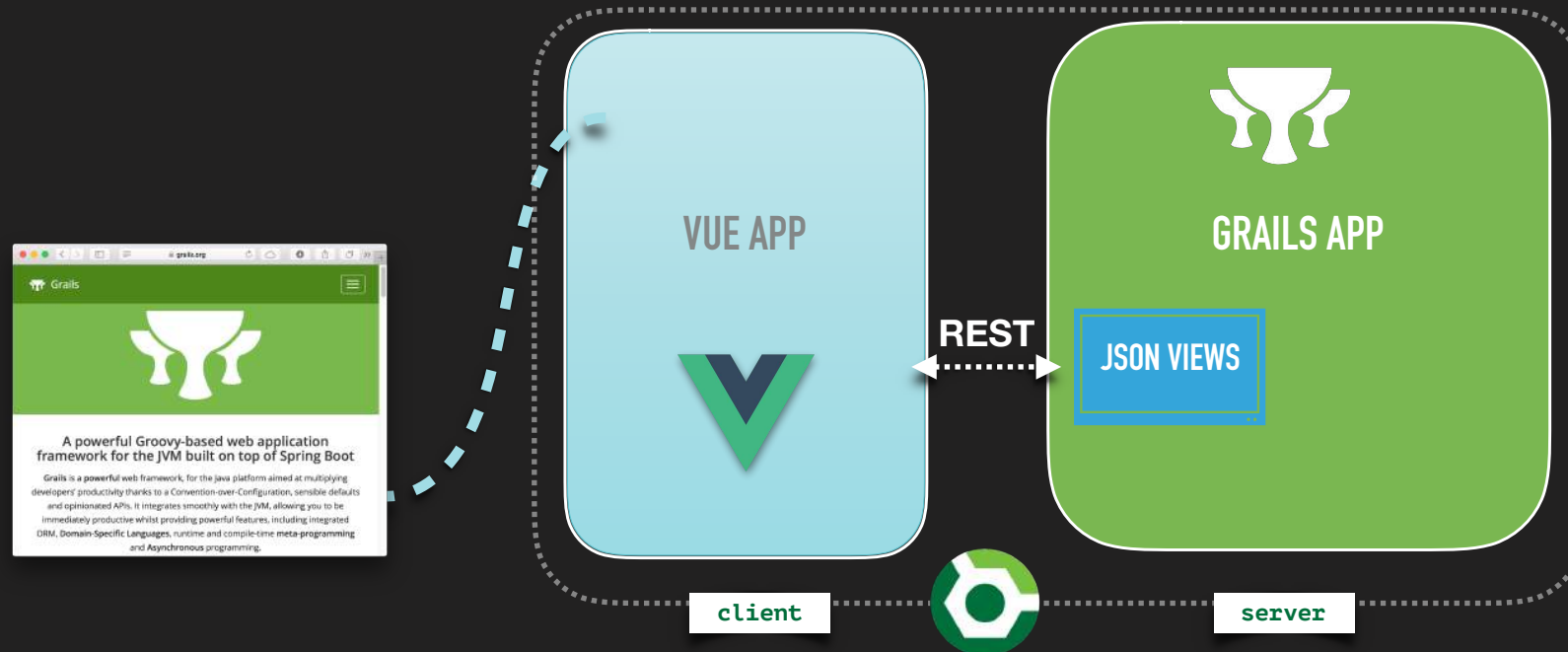
- Convention over Configuration
- Spring Boot compatibility
- Profiles & Plugins
- Gradle
- RESTful controllers & URL mappings
- JSON Views
- GORM (including GraphQL)



# Approaches to using JavaScript Frameworks with Grails

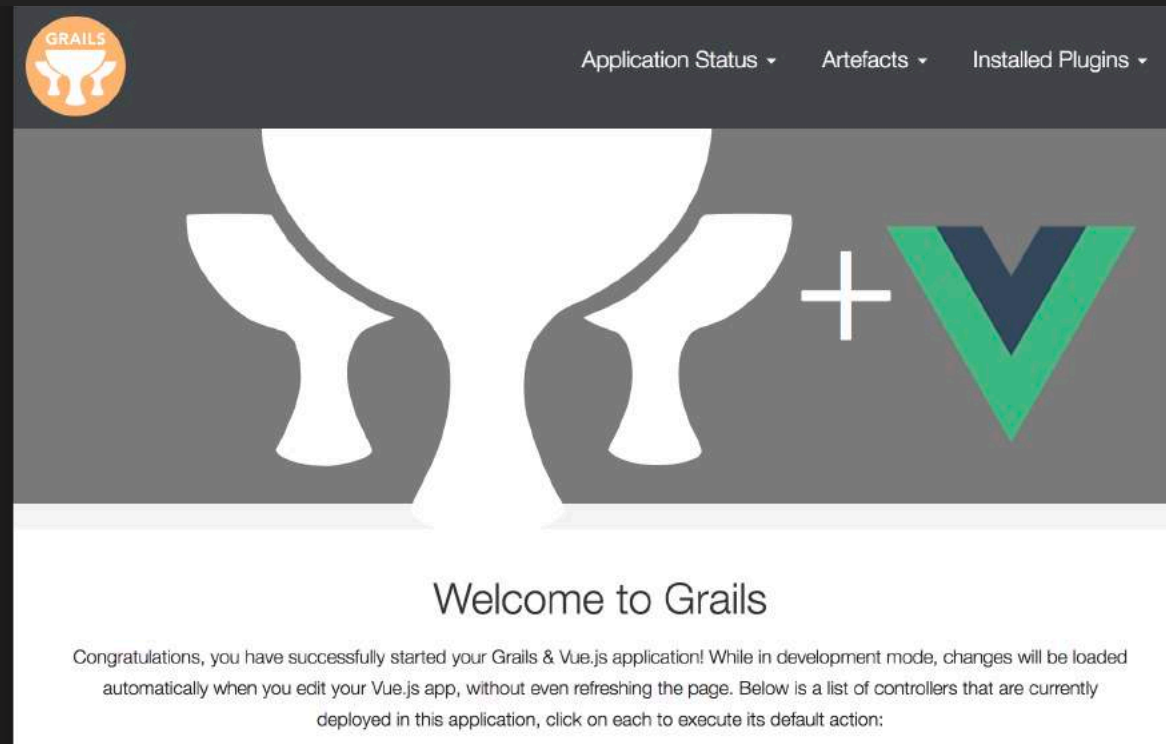


# Multi-project Build Approach



# Vue Profile for Grails

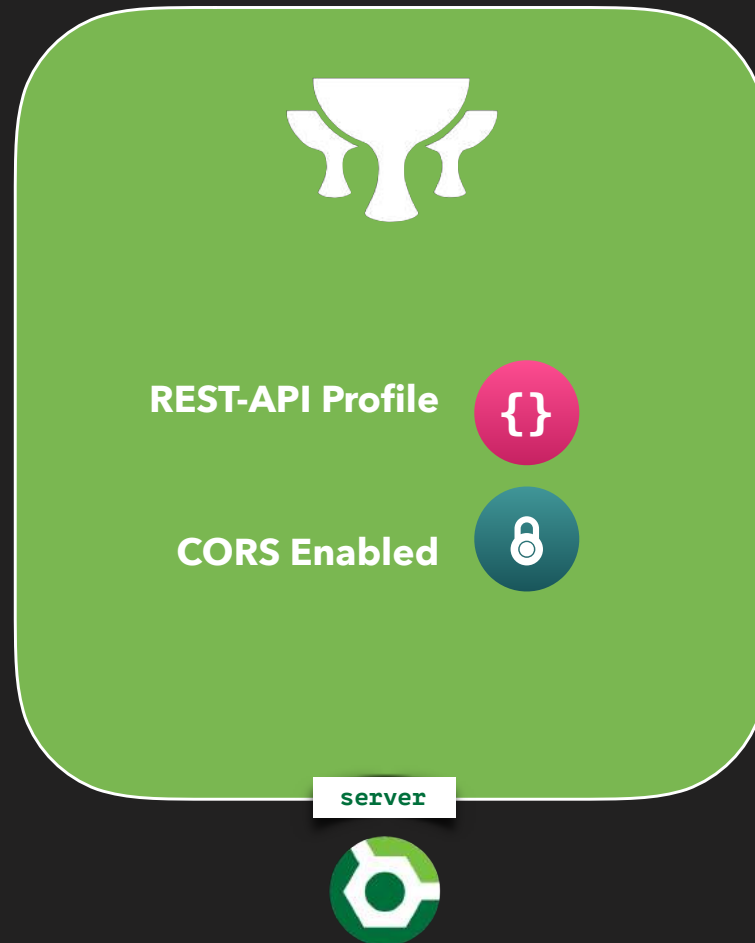
<https://grails-profiles.github.io/vue/latest/guide/index.html>



```
> grails create-app myapp -profile vue  
> curl -O start.grails.org/myapp.zip -d profile=vue
```

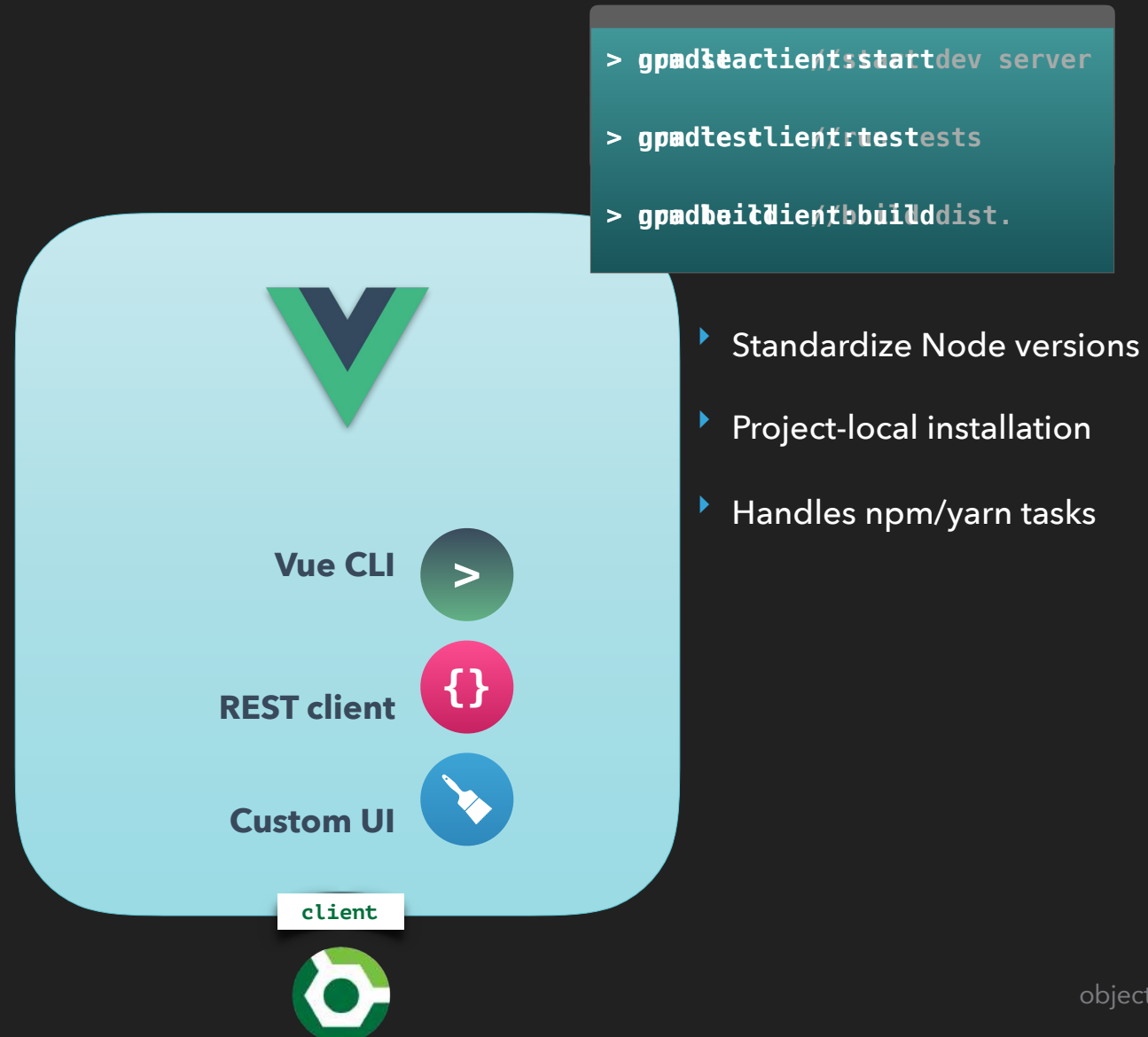
# Vue Profile for Grails

---





# Vue Profile for Grails



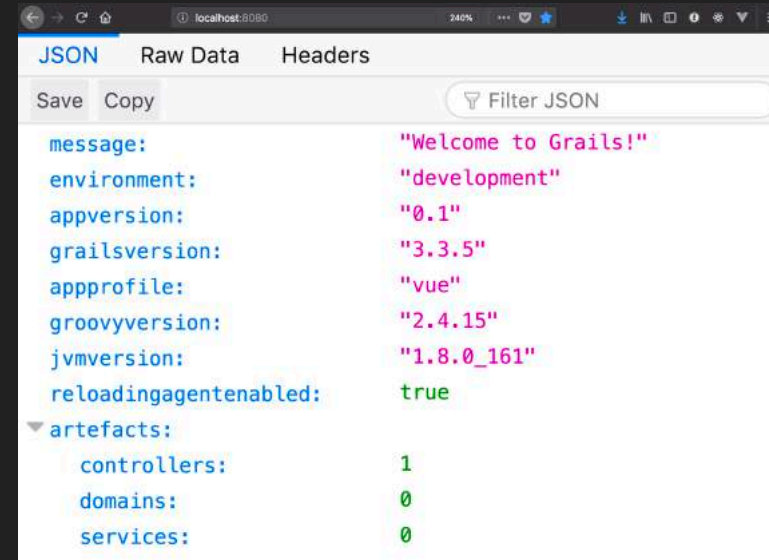
# Vue Profile for Grails

```
> gradle bootRun//or grails run-app
```

Grails application  
running at http://localhost:8080

```
> gradle start //or npm start
```

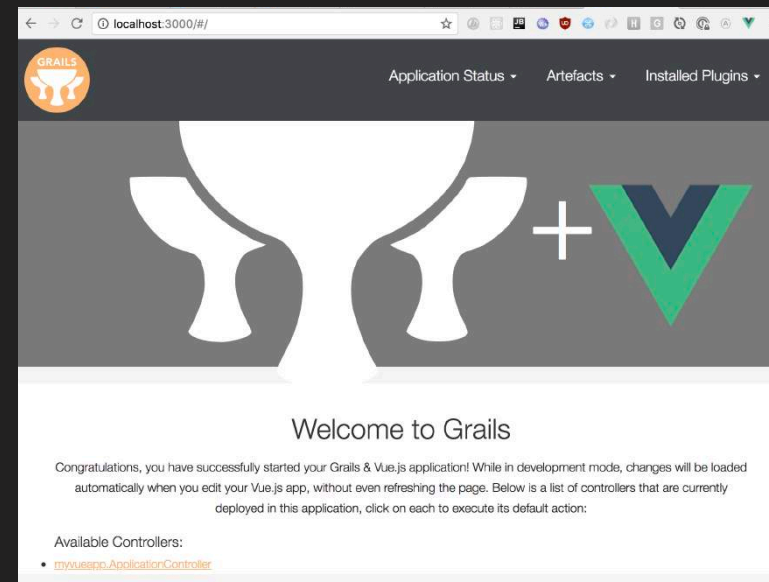
Your application is running here:  
http://localhost:3000



JSON Raw Data Headers

Save Copy Filter JSON

```
{  "message": "Welcome to Grails!",  "environment": "development",  "appversion": "0.1",  "grailsversion": "3.3.5",  "appprofile": "vue",  "groovyversion": "2.4.15",  "jvmversion": "1.8.0_161",  "reloadingagentenabled": true,  "artefacts": {    "controllers": 1,    "domains": 0,    "services": 0  }}
```



Application Status ▾ Artefacts ▾ Installed Plugins ▾

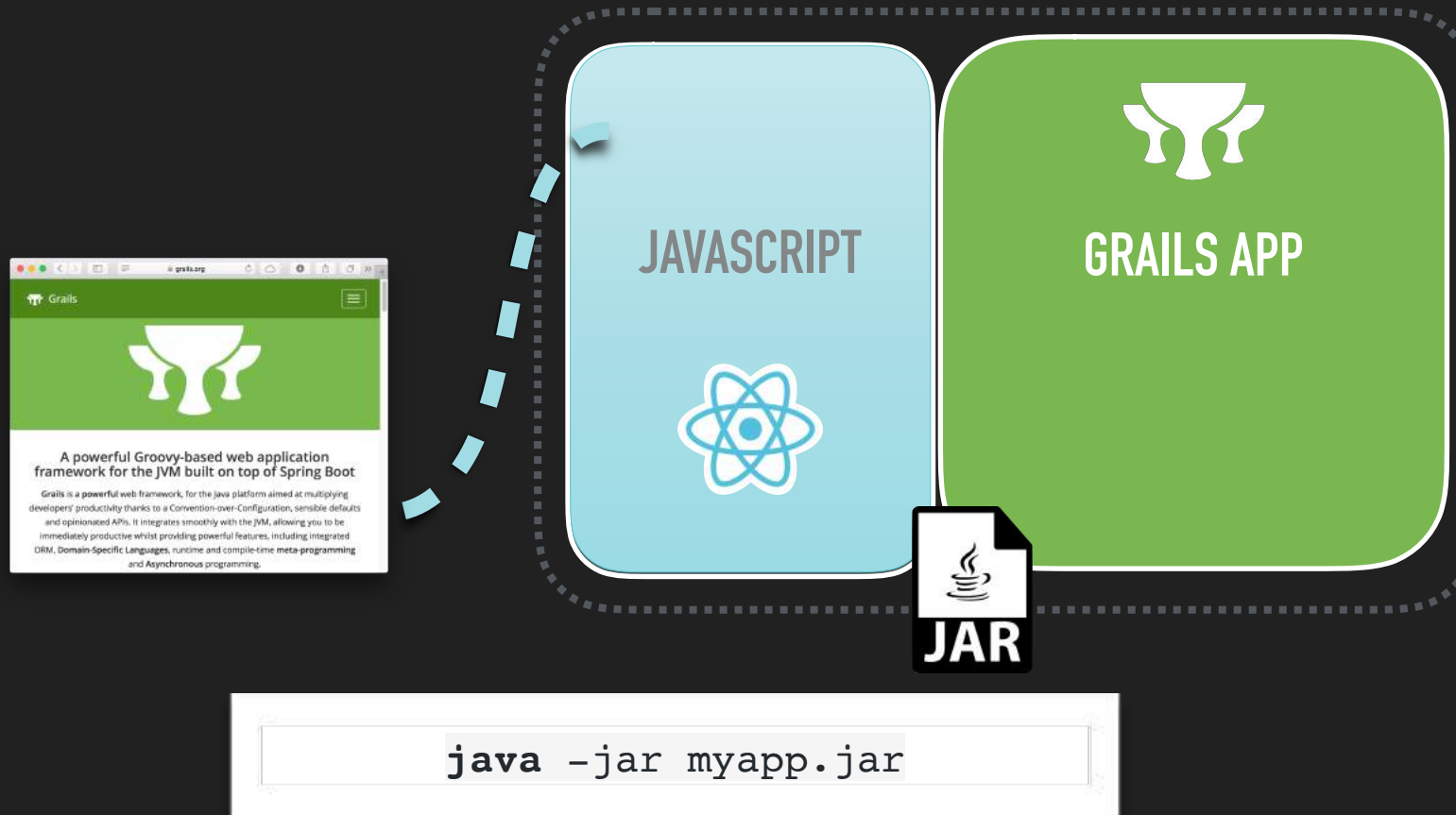
Welcome to Grails

Congratulations, you have successfully started your Grails & Vue.js application! While in development mode, changes will be loaded automatically when you edit your Vue.js app, without even refreshing the page. Below is a list of controllers that are currently deployed in this application, click on each to execute its default action:

Available Controllers:

- [myvueapp.ApplicationController](#)

# Combined JAR Deployment



# Vue with JWT (Spring) Security

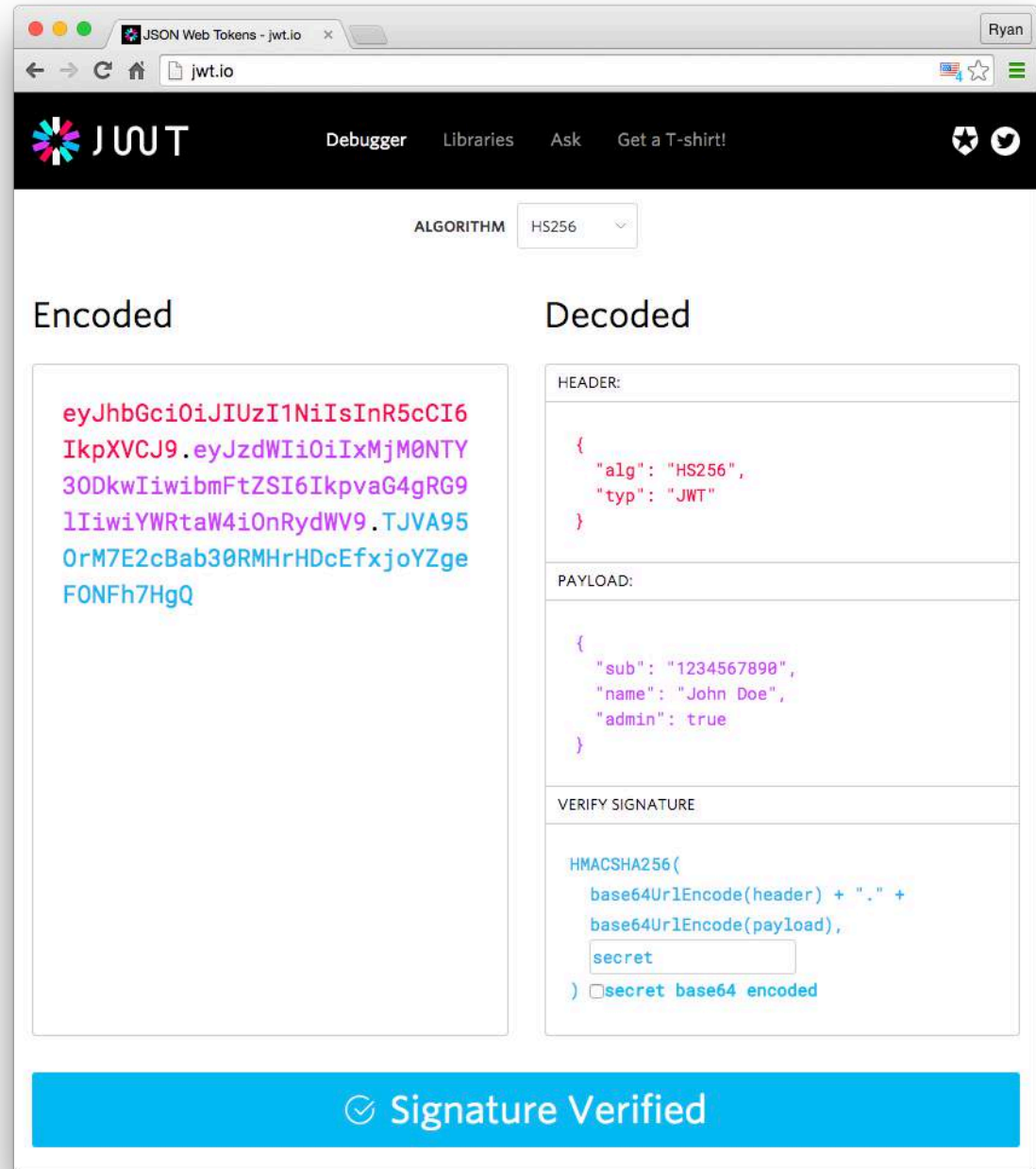


- Extends Spring Security Core
- OCI-supported
  - Adds support for stateless token-based authentication
  - Designed to secure restful APIs
  - Supports multiple token types/storage: JWT, GORM, Redis, etc

```
compile 'org.grails.plugins:spring-security-rest:3.0.0.RC1'
```

# JSON Web Token

- <https://jwt.io>
- Open, industry-standard method for representing claims securely between two parties
- Consist of a header, payload, and signature



The screenshot shows the JWT.io website interface. At the top, there's a navigation bar with the JWT logo, links for 'Debugger', 'Libraries', 'Ask', and 'Get a T-shirt!', and a user profile 'Ryan'. Below the navigation bar, the 'ALGORITHM' is set to 'HS256'. The 'Encoded' section displays a long string of base64-encoded characters. The 'Decoded' section shows the token's structure: a 'HEADER' with 'alg': 'HS256' and 'typ': 'JWT', and a 'PAYLOAD' with 'sub': '1234567890', 'name': 'John Doe', and 'admin': true. Below the payload, there's a 'VERIFY SIGNATURE' section showing the HMACSHA256 function being used to verify the token. At the bottom, a large blue banner indicates 'Signature Verified'.

JWT

Debugger Libraries Ask Get a T-shirt!

ALGORITHM HS256

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0Ij0iMTYzNDU2Nzg5LjJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

Decoded

HEADER:

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD:

```
{  "sub": "1234567890",  "name": "John Doe",  "admin": true}
```

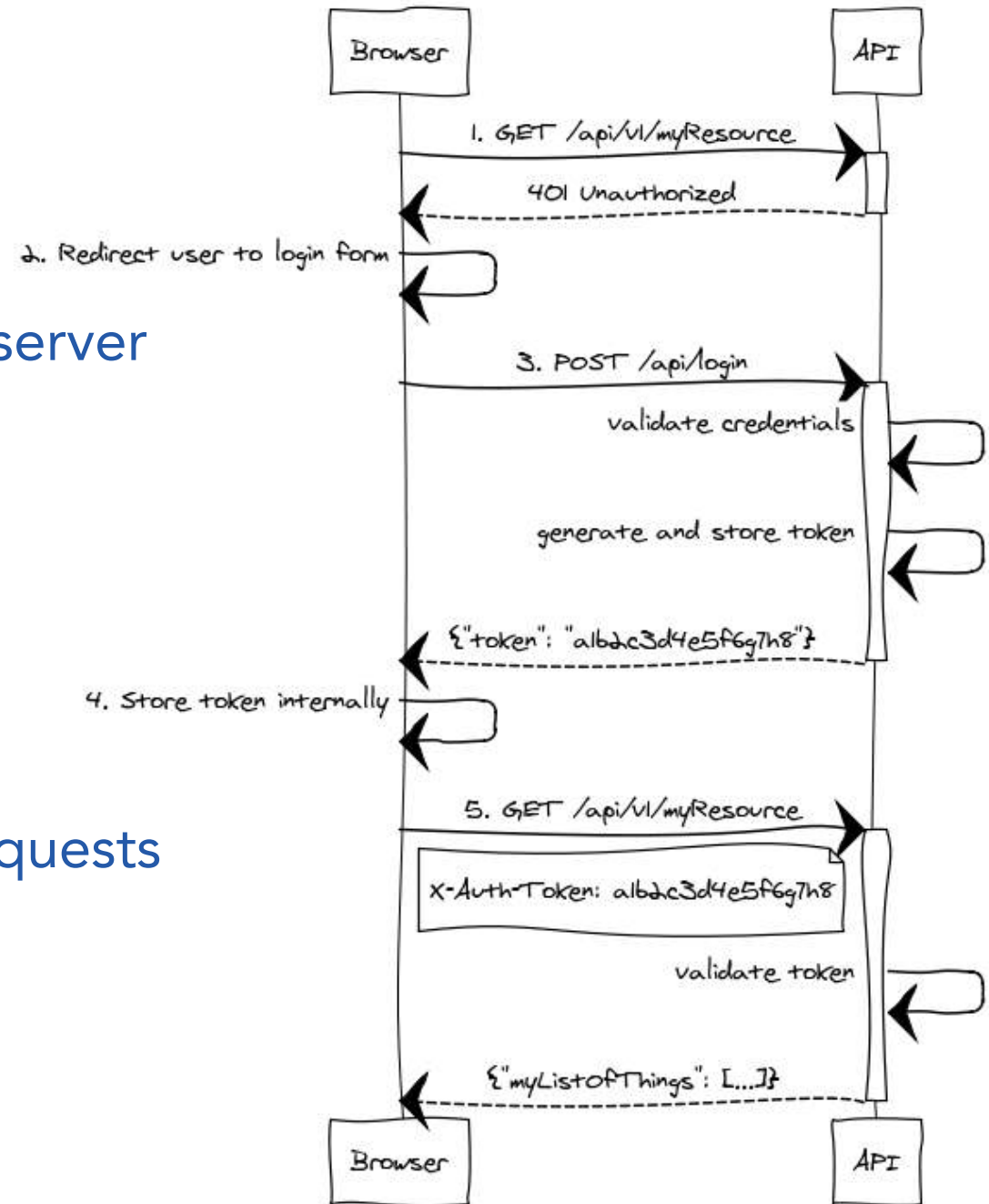
VERIFY SIGNATURE

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  secret  ) ☐ secret base64 encoded
```

Signature Verified

# JWT Authentication

- Client makes unauthorized request to server
- Server responds with 401
- Client POSTs to /login endpoint
- Server responds with token
- Client includes token in subsequent requests
- Server responds with resource



# Demo



# Grails & Vue Links

- Vue Profile docs: <https://grails-profiles.github.io/vue/latest/guide/>
- Building a Vue App: <http://guides.grails.org/building-a-vue-app/guide>
- Combined Build: <https://guides.grails.org/grails-vue-combined/guide>
- Using the Vue Profile: <http://guides.grails.org/using-the-vue-profile/guide>

# Thank you!

# LEARN MORE ABOUT OCI EVENTS AND TRAINING



## Events:

- [objectcomputing.com/events](https://objectcomputing.com/events)

## Training:

- [objectcomputing.com/training](https://objectcomputing.com/training)
- [grailstraining.com](https://grailstraining.com)
- [micronauttraining.com](https://micronauttraining.com)

Or email [info@ocitraining.com](mailto:info@ocitraining.com) to schedule a custom training program for your team online, on site, or in our state-of-the-art, Midwest training lab.



OBJECT  
COMPUTING

## CONNECT WITH US

---



1+ (314) 579-0066



@objectcomputing



objectcomputing.com