Single Page Apps with Grails & Vue.js

Zachary Klein, Senior Software Engineer
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 Guides: http://guides.grails.org
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

```html
<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**

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

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

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

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

```html
<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

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

```javascript
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`)
<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
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

- Asset Pipeline
- Hybrid Webapp
Multi-project Build Approach

GRAILS APP

VUE APP

REST

JSON VIEWS

client

server
Vue Profile for Grails


> grails create-app myapp -profile vue
> curl -O start.grails.org/myapp.zip -d profile=vue
Vue Profile for Grails

- REST-API Profile
- CORS Enabled

server
Vue Profile for Grails

- Standardize Node versions
- Project-local installation
- Handles npm/yarn tasks

```shell
> npm start
// start dev server
> npm test
// run tests
> npm build
// build dist.
> gradle client:start
> gradle client:test
> gradle client:build
```

- Vue CLI
- REST client
- Custom UI
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
Combined JAR Deployment

```
java -jar myapp.jar
```
Vue with JWT (Spring) Security
Spring Security REST

• 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](https://jwt.io)

- Open, industry-standard method for representing claims securely between two parties

- Consist of a header, payload, and signature
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
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

Training:
- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email 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.