February 9, 2018

Developing React Apps with Grails 3

ocitraining.com

oci training.com

OCI | HOME TO GRAILS

OCI | WE ARE SOFTWARE ENGINEERS.
☐ Team includes Grails co-founders and subject matter experts from around the globe

☐ Lead sponsor at G&G conferences around the globe

☐ 25+ updates and releases to the framework in the past 9 months

☐ Grails 3.3 GA released July 2017
Software Engineering Training

- 24+ years experience
- Over 50,000 trained
- 150 current courses
- More than 40 instructors on staff
- All training delivered by practitioners and SMEs in their respective fields
- Customized to fit your specific needs
- Flexible training delivery
- Training assessments
Introductions

☐ About your Instructor

• Web developer since 2010
• Joined OCI Grails team in 2015
• Developing with React since 2015
• Author of the React, React-Webpack, and Vue profiles for Grails

@zacharyaklein
https://guides.grails.org
https://start.grails.org
Create An Application Using The **Latest Release**

`curl -O start.grails.org/myProject.zip`

Create A **Plugin** Using The Latest Release

`curl -O start.grails.org/myPlugin.zip -d type=plugin`

Specify A Grails **Version**

`curl -O start.grails.org/myProject.zip -d version=3.3.2`

Specify A Grails **Profile**

`curl -O start.grails.org/restProject.zip -d profile=react`
Webinar Overview

- What is React?
  - Overview of React and front-end dev tools

- Approaches to Developing React with Grails
  - Asset Pipeline
  - Hybrid Web App
  - Multi-Project Build

- Using the React Profile
  - Spring Security REST

- Unified Deployment
What is React?
import React from ‘react’
import ReactDOM from ‘react-dom’

ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
What is React?

```javascript
import React from 'react'
import ReactDOM from 'react-dom'

ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
```
import React from 'react'
import ReactDOM from 'react-dom'

class Greeting extends React.Component {
    render() {
        return <h1>Hello, {this.props.name}!</h1>;
    }
}

ReactDOM.render(<Greeting name='G3 Summit' />, document.getElementById('root'));
import React from ‘react’
import ReactDOM from ‘react-dom’

class Greeting extends React.Component {
    render() {
        return <h1>Hello, {this.props.name}!</h1>;
    }
}

ReactDOM.render(<Greeting name=‘G3 Summit’ />, document.getElementById('root'))
class **Greeting** extends React.Component {
    render() {
        return(
            React.createElement('div', {},
                React.createElement('h1', {}, "Greetings, \${this.props.name}\")
            )
        );
    }
}

ReactDOM.render(React.createElement(Greeting, {name: 'G3 Summit'}, null), document.getElementById('root'));
class **Greeting** extends React.Component {
    render() {
        return (<div>
            <h1>Hello, {this.props.name}!</h1>
            <ul>
                <li><a href='edit'>Edit this greeting</a></li>
                <li><a href='reset'>Reset this greeting</a></li>
            </ul>
        </div>);
    }
}

ReactDOM.render(<Greeting name='G3 Summit' />, document.getElementById('root'));
What is React?

“React is only the view layer. We're only in one concern. React only knows how to render markup. It doesn't know where your data came from, how it's stored, or how to manipulate it. What concerns are being violated?”

-Andrew Ray

Source: [http://blog.andrewray.me/youre-missing-the-point-of-jsx/](http://blog.andrewray.me/youre-missing-the-point-of-jsx/)
render() {
  return <div className="list-container">
    <ul>JSX Examples</ul>
    <li>Here’s an expression: {3 + 3}</li>
    <li>State property: {this.state.myProperty}</li>
    <li>Function call: {myFunc()}</li>
    <li><button onClick={this.myEventHandler}>
      Function reference: </button></li>

    {myItemList.map(item => <li>{item}</li>)}

  </ul>
</div>;
class Greeting extends React.Component {
constructor() {
  super();

  this.state = {
    greetings: ['Hello', 'Salutations', 'Ho there']
  }
}

render() {
  const greetings = this.state.greetings;
  const randomGreeting = greetings[Math.floor(Math.random() * greetings.length)];

  return(
    <h1>{randomGreeting}, {this.props.name}</h1>
  );
}
}

ReactDOM.render(<Greeting name='G3 Summit' />, document.getElementById('root'));
Component State

```js
this.state = { ... }
```

1. **Component is rendered**
   - State
   - Rendered output

2. **State**
   - Rendered output

3. **Component is re-rendered**
   - State
   - Rendered output
Virtual DOM

State changed!
let value = "hello"; //could also pass in a literal value
<MyComponent prop={value} /> //this.props.prop = "hello"

let a = 1;
let b = 2;
<MyComponent prop={a + b} /> //this.props.prop == 3

const myFunc = (a, b) => a + b;
<MyComponent prop={myFunc} /> //this.props.myFunc(1, 2) == 3
Component Props

Component is rendered

Props

Rendered output
this.props.myProp = 3

Props changed

Props

a = 3

Rendered output
this.props.myProp = 3

Component is re-rendered

Props

Rendered output
this.props.myProp = 5

Copyright © 2018 by Object Computing, Inc. (OCI). All rights reserved.
Component Hierarchy
Component Hierarchy
Component Hierarchy

React will re-render all affected child components
Component Hierarchy

State/Props ➔ Rendered

1 ➔
Component Hierarchy

Child components re-rendered
React is a small, focused library by design, but there's plenty of options for augmentation.
React Ecosystem

- [https://medium.freecodecamp.org/how-to-navigate-the-react-js-ecosystem-without-getting-lost-43db14b00e08](https://medium.freecodecamp.org/how-to-navigate-the-react-js-ecosystem-without-getting-lost-43db14b00e08)
- [https://github.com/enaqx/awesome-react](https://github.com/enaqx/awesome-react)
- [https://github.com/xgrommx/awesome-redux](https://github.com/xgrommx/awesome-redux)

* Somewhat out of date, but most of the libraries mentioned are still popular options
Build Tooling

React can be used standalone but is more commonly used in a node.js environment.
Why Use React & Grails?

Many features make Grails ideally suited to serve as a backend for Single Page Applications.

- GORM
- Convention/Configuration
- Spring Boot
- Profiles
- Plugins
- Gradle
- URL mappings
- JSON views
- Hypermedia
- GORM for GraphQL
- WebSockets
Approaches Using React with Grails

- Asset Pipeline
- Hybrid Web App
- Multi-Project Build
Approaches Using React with Grails

- Asset Pipeline
- Hybrid Web App
- Multi-Project Build
Asset Pipeline

- Core Grails Plugin
- Extensible with many plugins available
- Works well with `client-dependencies` plugin
- Natively processes React/JSX (no node.js tooling)
- Efficient and performant
- Familiar to many Grails developers
- Provides minification, source maps, compression
- Strong Grails community support
Results:

- **angular-annotate-asset-pipeline** craigburke
  AngularJS Annotate for Asset Pipeline 2.0+
  com.craigburke.angular:angular-annotate-asset-pipeline:2.4.1

- **angular-template-asset-pipeline** craigburke
  AngularJS Templates for Asset Pipeline 2.0+
  com.craigburke.angular:angular-template-asset-pipeline:2.4.0

- **asset-pipeline-grails** bertramlabs
  The Asset-Pipeline is a plugin used for managing and processing static assets in Grails applications. Asset-Pipeline functions include processing and minification of both CSS and JavaScript files. It is also capable of being extended to compile custom static assets, such as CoffeeScript.
  com.bertramlabs.plugins:asset-pipeline-grails:2.14.6

- **asset-pipeline-servlet** bertramlabs
  Generic servlet filter for serving static assets processed by the asset-pipeline
  com.bertramlabs.plugins:asset-pipeline-servlet:2.14.6

- **ember-asset-pipeline** bertramlabs
  Compiles hbs or handlebars files for the asset-pipeline into the Ember.TEMPLATES cache

http://asset-pipeline.com
Asset Pipeline

GRAILS APP

grails-app/assets

A powerful Groovy-based web application framework for the JVM built on top of Spring Boot

Grails is a powerful web framework for the Java platform aimed at multiplying developers’ productivity thanks to a Convention-over-Configuration, sensible defaults and opinionated APIs. It integrates smoothly with the JVM, allowing you to be immediately productive whilst providing powerful features, including integrated ORM, Domain-Specific Languages, runtime and compile-time meta-programming and Asynchronous programming.
DEMO
Approaches Using React with Grails

- Asset Pipeline
- Hybrid Web App
- Multi-Project Build
Approaches Using React with Grails

- Asset Pipeline
- Hybrid Web App
- Multi-Project Build
Hybrid Web App

- Merge front-end project with Grails project
- Leverage native JavaScript build tools (webpack)
- Leverage Gradle Node Plugin
- Serve JavaScript via Asset Pipeline
- Can be used alongside GSPs
- Best approach for server-side rendering
- Analogous to the JHipster stack (Spring Boot + Angular)
Webpack

- JavaScript Module bundler
- Supports hundreds of asset types
- Links/bundles dependencies in JavaScript apps
- Extremely configurable (sometimes too much)
- Builds a dependency graph from one or more “entries”
- Outputs a (optionally minified, “chunked”) bundle
- Supports hot-reloading
modules with dependencies

webpack MODULAR BUNDLER

static assets

JavaScript Module bundler
Supports hundreds of asset types
Links/bundles dependencies in JavaScript apps
Extremely configurable (sometimes too much)
Builds a dependency graph from one or more “entries”
Outputs a (optionally minified, “chunked”) bundle
Supports hot-reloading
Hybrid Web App Approach

A powerful Groovy-based web application framework for the JVM built on top of Spring Boot

Grails is a powerful web framework, for the Java platform aimed at multiplying developers' productivity thanks to a Convention-over-Configuration, sensible defaults and opinionated APIs. It integrates smoothly with the JVM, allowing you to be immediately productive whilst providing powerful features, including integrated ORM, Domain-Specific Languages, runtime and compile-time meta-programming and Asynchronous programming.
Application Profiles

- Include default config, plugins, project structure
- Custom commands/scripts for code-generation
- Optional “features” allow further control of the generated project
- Profile inheritance
- Packaged/resolved as Maven dependencies

grails create-profile myProfile
Application Profiles

```
~ grails create-app myApp -profile rest-api
~ cd myApp
~ grails create-controller MyRestfulController
```
React-Webpack Profile

- React, ReactDOM etc., installed via npm
- Webpack configured to process React code and output to `grails-app/assets/javascripts`
- React code lives in `src/main/webapp`
- gradle-node plugin installed, custom tasks to run webpack on app startup/packaging
- Sample React code and unit tests included

```
grails create-app myReactApp --profile react-webpack
OR
curl -O start.grails.org/myapp.zip -d profile=react-webpack
```
React-Webpack Profile

$ ls -l
grails-app/
-  assets/
-  javascripts/
-  bundle.js
node_modules/
package.json
src/
  main
  webapp/
-  app/
-  about.js
-  test/
-  js/
-  about.spec.js
webpack.config.js
React-Webpack Profile

A powerful Groovy-based web application framework for the JVM built on top of Spring Boot

Grails is a powerful web framework for the Java platform aimed at multiplying developers' productivity thanks to a Convention-over-Configuration, sensible defaults and opinionated APIs. It integrates smoothly with the JVM, allowing you to be immediately productive whilst providing powerful features, including integrated ORM, Domain-Specific Languages, runtime and compile-time meta-programming and Asynchronous programming.

src/main/webapp

GRAILS APP

JAVASCRIPT APP
Approaches Using React with Grails

☐ Asset Pipeline

☐ Hybrid Web App

☐ Multi-Project Build
Approaches Using React with Grails

- Asset Pipeline
- Hybrid Web App
- Multi-Project Build
Multi-Project Build

- Separate projects for front-end app and Grails app
- Optionally leverage Gradle’s multi-project builds
- Depends on CORS (native support in Grails 3.2+)
- Minimal coupling between front and back-end apps
- Client can point to any server
- Allows separate front- and back-end teams to work independently
- Allows use of all JavaScript build tools and community support

```
grails create-app myReactApp —profile react
```
create-react-app

“Create React App is a new officially supported way to create single-page React applications. It offers a modern build setup with no configuration.”

Source: https://facebook.github.io/react/blog/2016/07/22/create-apps-with-no-configuration.html

- Generates fully standalone React app
- Pre-configures webpack and Babel
- Provides scripts for starting app, building public bundle, running tests
- Simple, rapid dev workflow with an easy “exit strategy”

~ npm install -g create-react-app
~ create-react-app myApp
~ cd myApp; npm start
React Profile

React Profile generates a multi-project Gradle build with:

- A React app (generated via `create-react-app`) as client project
- A Grails 3 app (`rest-api` profile) as server project
- Gradle-node tasks defined within client project to run npm/yarn scripts (start, build, test, and eject)
- Client app serves default React index page built with `react-bootstrap`, with app data retrieved from server via REST call

```
grails create-app myReactApp --profile react
OR
curl -O start.grails.org/myapp.zip -d profile=react
```
React Profile

**JAVASCRIPT APP**

**GRAILS APP**

Client

Server

---

A powerful Groovy-based web application framework for the JVM built on top of Spring Boot

Grails is a powerful web framework, for the Java platform aimed at multiplying developers' productivity thanks to a Convention-over-Configuration, sensible defaults and opinionated APIs. It integrates smoothly with the JVM, allowing you to be immediately productive whilst providing powerful features, including integrated ORM, Domain-Specific Languages, runtime and compile-time meta-programming and Asynchronous programming.
React Profile

$ ls -l

client/
  - build.gradle
  - node_modules/
  - package.json
  - public/
  - src/
    - App.js
    - App.test.js
server/
settings.gradle
DEMO
Approaches Using React with Grails

- Asset Pipeline
- Hybrid Web App
- Multi-project Build
Approaches Using React with Grails

- Asset Pipeline
- Hybrid Web App
- Multi-project Build
Grails Quickcast: Multi-project Build

Grails Quickcast #3 - Multi Project Builds

https://www.youtube.com/watch?v=yNA0ce5fG9s
Approaches Using React with Grails

- Asset Pipeline
- Hybrid Web App
- Multi-project Build
Spring Security REST

- Extends Spring Security Core
- OCI-supported
- Adds support for stateless, token-based, auth
- Designed to secure restful APIs
- Supports multiple token types/storage strategies, including JWT, Memcache, GORM, Redis, and custom

compile 'org.grails.plugins:spring-security-rest:2.0.0.M2'
**Authentication Sequence**

1. GET /api/v1/myResource
   - 401 Unauthorized

2. Redirect user to login form

3. POST /api/login
   - validate credentials
   - generate and store token
   - 
   - "token": "a1b2c3d4e5f6g7h8"

4. Store token internally

5. GET /api/v1/myResource
   - X-Auth-Token: a1b2c3d4e5f6g7h8
   - validate token
   - 
   - {"myListofThings": [...]}

Browser

API
DEMO
Combined Deployment

```
java -jar myapp.jar
```
Online Grails Training Workshops

We offer a variety of live, instructor-led workshops, conveniently delivered as short sessions by Grails co-founder, Jeff Scott Brown and other members of the Groovy and Grails team.

Keep your eyes open for announcements of upcoming courses!

- Introduction to REST Services with Grails 3
- GORM Deep Dive
- Groovy Metaprogramming
- Grails Security
- Grails Testing

Visit grailstraining.com for more information
Grails Community Information

grailsblog.objectcomputing.com
grails.org
github.com/grails
grails.signup.team
objectcomputing.com/products/grails
grailstraining.com
@grailsframework
@objectcomputing
#grailsfw
Groovy Community Information

groovy.apache.org

groovy-lang.org

dev@groovy.apache.org

groovycommunity.com

github.com/apache/groovy

users@groovy.apache.org

objectcomputing.com/training/catalog/groovy-programming

@ApacheGroovy

#groovylan

groovylang
Contact Us

Zachary Klein
Software Engineer, Grails Team
info@ocitraining.com

Jen Wiese
Manager, Training and Workforce Development
wiesej@objectcomputing.com
(314) 579-0066