Web User Interfaces Review

Andrea Bolognesi, EN/ACE/AMM
andrea.bolognesi@cern.ch

Contents

- JSF
  - https://gitlab.cern.ch/abologne/dev-jsf.git
- Angular
  - https://gitlab.cern.ch/abologne/dev-a.git
- React
  - https://gitlab.cern.ch/abologne/dev-r.git
- Polymer
  - https://gitlab.cern.ch/abologne/dev-p.git
- Comparison

Example

Demo Grid Data

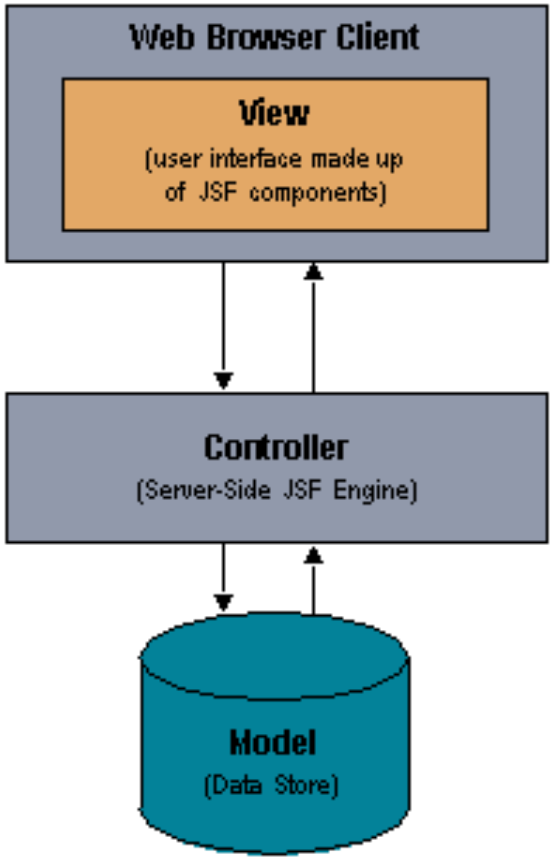
Filtering table with:

Filter

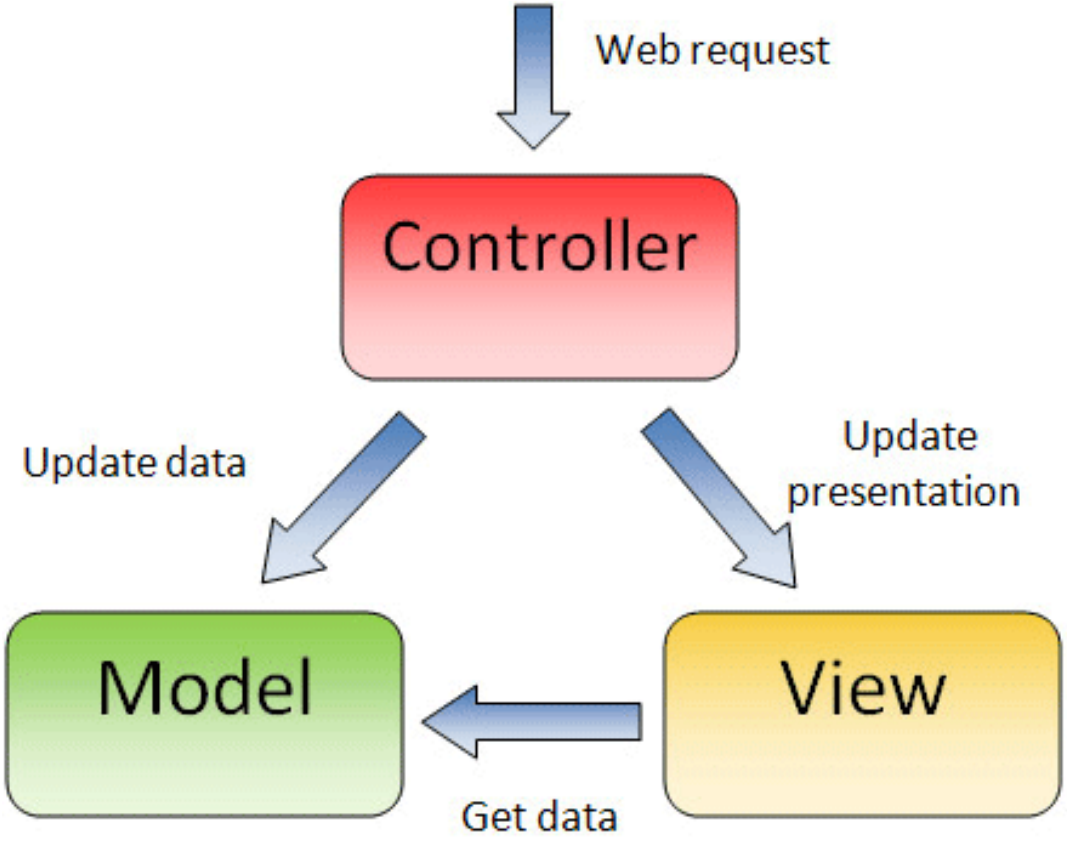
Code	Description
QLI-EX14=180	SVB1-PA box (SVB-1 / WAT-QLI)
QLI-EX24=180	SVB2-PA box (SVB-2 / WAT-QLI)
QLI-EX34=180	SVB3 - Profibus PA box (SVB-3 / WAT-QLI)
QLI-EX43=180	(DEWAR VB / QLDH-WAT)
QLI-EX44=180	CVB-ProfibusPAbox (CVB / WAT-QLI04)
QLI-EX45=180	DVB-ProfibusPAbox (DVB / WAT-QLIM)

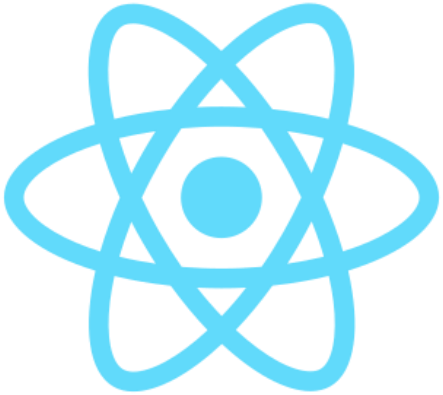
JavaServer Faces JSF

- Server side objects (managed backing beans) associated with UI components used in the page.
- Managed beans are JavaBeans components
  - Define Ui component properties bound to component's value
  - Define methods that perform functions associated with a component (validation, event handling, navigation processing).

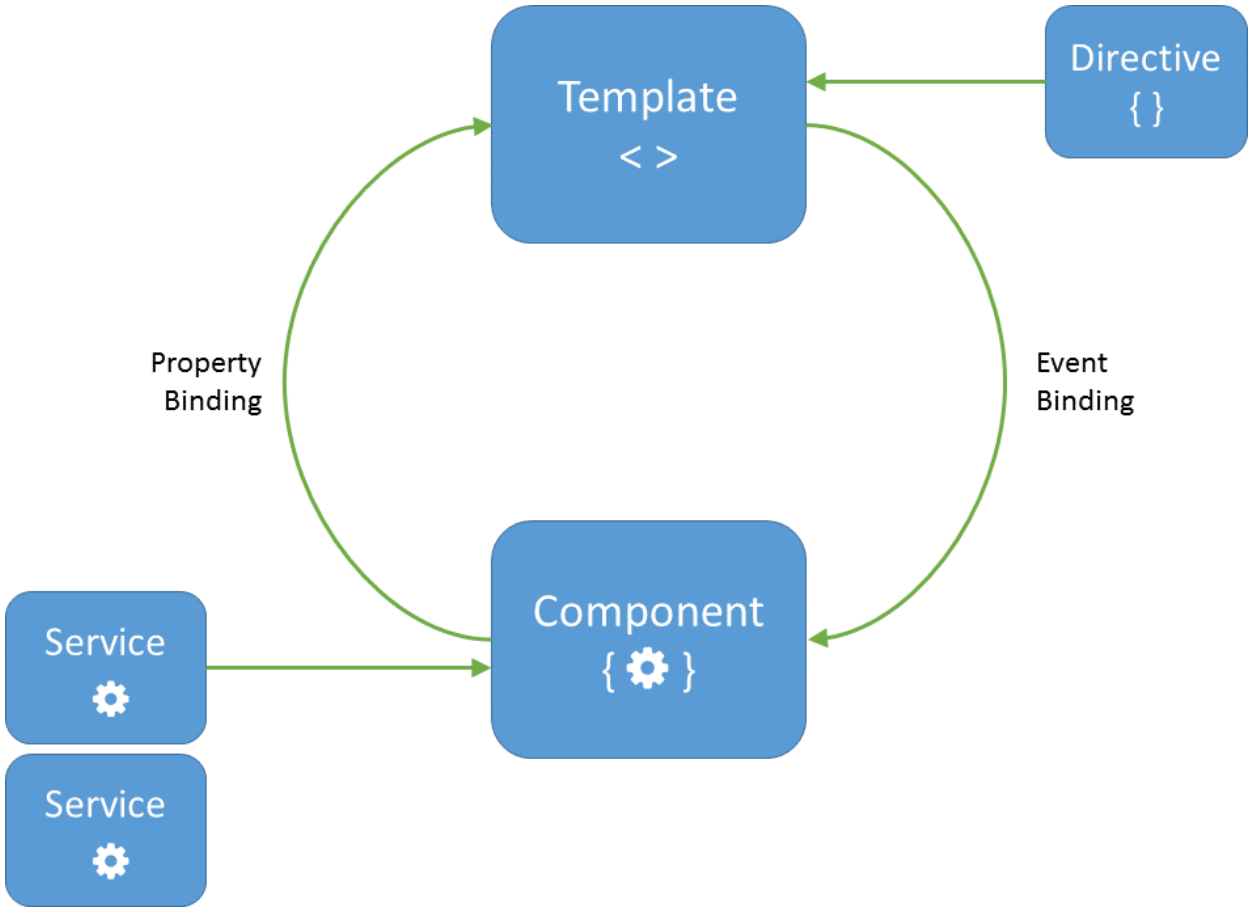


MVC Pattern


Web request
Controller
Update data
Update
presentation
Model
View
Get data



Components



- Taken from https://courses.edx.org/courses/course-v1:Microsoft+DEV216x+2T2017

Directives

- Directives allow the injection of custom behavior into existing HTML elements:
  - **Component directives**:

```
<h1>Demo Grid Data</h1>
<datagrid-component></datagrid-component>
```

  - **Attribute directives** change the behavior or appearance of an element.

```
<div [ngStyle]="{`color`:object.color}">
```

  - **Structural directives** show or hides an element.

```
<tr *ngFor="let object of objects">
```

Interpolation

Allows to weave HTML markup and dynamic data:

```
<tr *ngFor="let object of objects">
    <th scope="row">{{object.code}}</th>
    <td>{{object.desc}}</td>
</tr>
```

Weaves calculated strings into the text between HTML element tags and within attribute assignments:

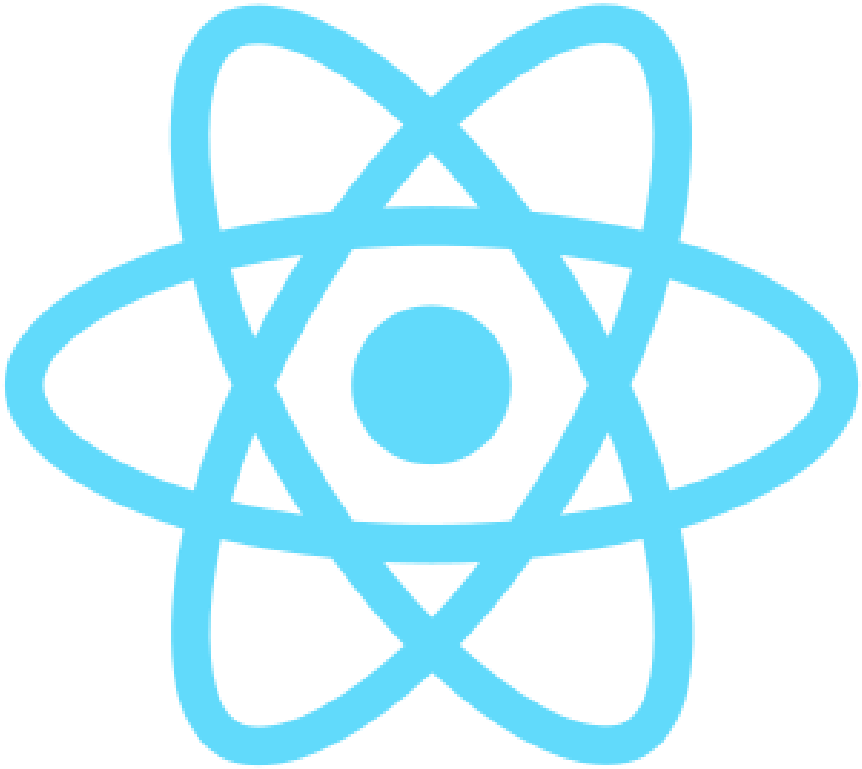
```
<img src="{{imageURL}}" >
```

Two-way binding

- The [(ngModel)] data binding syntax enables a two-way binding scenario.
- In a two-way binding scenario,
  - our property's value is updated in our component class whenever the user makes a change in the UI.
  - our UI is inversely updated if we change the value of the property in the component class.

Modular Application Design

- Architectural Goals:
  - Software components must be modular and reusable throughout your application
  - Modular components should be testable in isolation
  - Modules could be changed easily without rewriting the entire application



React and Composition

- React is a declarative, efficient, and flexible JavaScript library for building user interfaces.
  - https://reactjs.org/
- The key feature of React is **composition** of components.
  - Components written by different people should work well together.

JavaScript XML (JSX)

- Describes what the UI should look like.
- May remind you of a template language, but it comes with the full power of JavaScript.

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Interactive Component

```
<button onClick={() => alert('click')}>
  {this.props.value}
</button>
```

JavaScript XML (JSX)

- JSX is an XML-like syntax extension to ECMAScript.
- It's NOT intended to be implemented by engines or browsers.
- It's NOT a proposal to incorporate JSX into the ECMAScript spec itself.
- It's intended to be used by various preprocessors (transpilers) to transform these tokens into standard ECMAScript.

React Component

```
import React from 'react';
import DataGridRow from './DataGridRow';

const DataGrid = (props) => (

    <table className="table table-striped table-bordered"><tbody> {
            props.objects.map((object, index) => (
                <DataGridRow
                    key={object.code}
                    object={object}
                    count={index + 1}
                />
            ))
    }     </tbody></table>
);

export default DataGrid;
```

Polymer

- Google Polymer is a library that provides syntactic sugar and polyfills for building elements and applications with web components.
  - https://www.webcomponents.org
  - https://www.polymer-project.org
- Web components are reusable widgets that can be assembled like building blocks in web documents and apps.
- Good Reference to get started:
  - https://auth0.com/blog/build-your-first-app-with-polymer-and-web-components

Web Components

Web Components are a set of browser features that are being added to the W3C HTML and DOM specification.

- https://www.w3.org/standards/techs/components#w3c_all

2014-03-18	HTML Templates Describes a method for declaring inert DOM subtrees in HTML and manipulating them to instantiate document fragments with identical contents
2017-09-05	Shadow DOM Describes a method of establishing and maintaining functional boundaries between DOM subtrees and how these subtrees interact with each other within a document tree.
2016-10-13	Custom Elements This document describes the method for enabling the author to define and use new types of DOM elements in a document.
2016-02-25	HTML Imports This document defines a way to include and reuse HTML documents in other HTML documents.

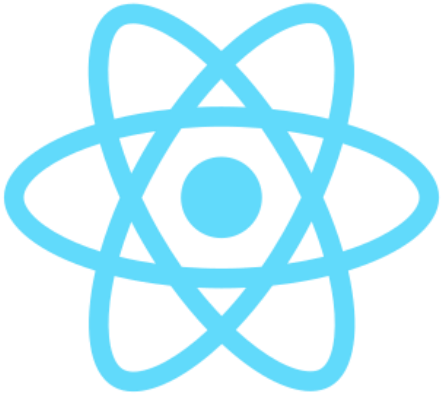
Web Components

- Allow us to architect and import custom elements that automatically associate JS behavior with templates and can utilize shadow DOM to provide CSS scoping and DOM encapsulation.
- Could be used natively without any additional libraries or toolsets.
  - However, not all features are supported by all browsers.
  - Need library like Polymer or polyfills, such as webcomponents.js, to bridge the gap between the current state of browser support and the future.

Web Components

- Shadow DOM is difficult and costly to polyfill, so Polymer uses Shady DOM to implement the features of Shadow DOM in browsers that lack support.
  - https://github.com/webcomponents/shadydom

Comparison



Adoption

- Angular, Polymer, and React are supported and used by big companies.
- Facebook, Instagram and Whatsapp are using React for their pages.
- Google uses Angular in a lot of projects: for example, the new Adwords UI was implemented using Angular & Dart.
- Google uses Polymer in Google Keep, YouTube.

Tools and Languages

- In Angular TypeScript is the de-facto language for building Angular apps.
- React focuses on the use of Javascript ES6.
- Polymer is based on Web standards, components are built with HTML and JavaScript.
  - In Polymer 2 and later, the default language level is ES6, which will get transpiled down to ES5 by the CLI to support older browsers.

Passing state between component

- Angular 2+ keeps the state in a shared service, available to components that need it through dependency injection.
- React has a single central store using Redux that components can bind to.
- With Polymer, the component-based development model offered by Web Components is flexible enough that you can build complex applications using the browser as your framework.

Templates and Data Binding

- Angular
  - Templates are enhanced HTML with special Angular language
  - One-way and two-way data binding
  - Dependency Injection (DI) system that makes it easier to hook up services, especially when testing.
  - RxJS and Observables for handling data, especially for asynchronous HTTP communication.
- React
  - Uses JSX is an optional preprocessor for HTML-like syntax which will be compiled in Javascript later.
- Polymer
  - On top of the Web Components standards
  - simple data binding system that supports both one-way and two-way binding of data.

Debugging

- Debugging Polymer applications can be done using the browser developer tools.
- The Angular CLI build creates source maps that allow you to debug the TypeScript code in the browser.

Application Routing

- Since Angular, React and Polymer are component-based, the general application structure is similar.
- Angular has a feature-rich router that supports arbitrarily deeply nested routes.
  - lazy load code for modules until needed.
- Polymer has an optional router (app-route) that can be used to map between URLs and components.

Styling

- Polymer
  - harder to customize in terms of look and feel because Web Components are designed to encapsulate their implementations.

Testing

- Angular
  - Karma
  - Jasmine
- React
  - Jest
  - Enzyme

Framework vs. library

- Angular is a framework rather than a library.
- React and Polymer are libraries, although with the right extension can be seen as a framework (especially for React).

Mobile Support

- Angular and React
  - Web applications and native applications (although native support requires you to write a separate implementation of the view).
    - Ionic Framework (Angular)
    - React-native (React)
- Polymer
  - PWA is the only mobile strategy. This is in line with Polymer’s goal of building on Web standards to expand what you can do on the Web.

Maintainability

- Angular and Polymer did not show a stellar API stability:
  - Polymer had a very rough transition period leading from version 0.5 until 1.0.
  - Angular 2 went through an extended period of API changes during development.
  - The good news is that both teams have taken note and are planning for smoother upgrade paths in the future. Polymer 2,
- Polymer and the W3C standards it builds on will provide a more stable foundation for apps that need to be maintained for longer periods.

References

- https://reactjs.org
- https://angular.io
- https://www.polymer-project.org
- https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176
- https://vaadin.com/blog/comparing-polymer-and-angular-from-a-developer-s-perspective
- https://courses.edx.org/courses/course-v1:Microsoft+DEV216x+2T2017
- https://completereactcourse.com