METAL.JS: SOY TEMPLATES

WHAT ARE SOY TEMPLATES?

- *Soy Templates*, also known as Closure Templates, are developed by Google. They are a client-side and server-side templating system that helps you dynamically build reusable HTML and UI elements.
- They have a simple syntax, and you can customize them to fit your application's needs.
- In contrast to traditional templating systems, in which you must create one monolithic template per page, you can think of Soy Templates as small components that you compose to form your UI.
- You can also use the built-in message support to easily localize your applications.
- Metal.js is designed to work seamlessly with Soy Templates.

HOW IS THIS DIFFERENT?

- *Soy Templates* can be combined to build dynamic displays.
- This means we can build interface components and views piece by piece.
- Instead of having one big template for each "page" (*view*), we can break it out into components that can be reused:
  - Search bar
  - Table view
  - Data list
  - Paginator
  - Carousel
- Let's see what this looks like.

MAKING JAVA PRETTY

- A classic use case for templates is to build pretty views for an application.
- Java can use tools like FreeMarker to create views from templates.
- The Java application provides data to the template and compiles it into a final output that will become the HTML for the page.

FROM WHOLE TO PARTS

- In this approach, we have to construct the entire view in one template.
- To break it down for ease of development and flexibility, we might be able to include other templates.
- Including external templates is limited, and doesn't allow for dynamically modifying the component based on environment or data.
- Soy Templates are components, and can be run independently.
- Building one view might mean calling multiple Soy Templates to build a search bar, buttons, and table view.
- A Soy Template can be *called* — just like a JavaScript function.

SOY TEMPLATE EXAMPLE

- A Soy Template can be self-contained:

```
{namespace Modal}

/**
 * This renders the component's whole content.
 * Note: has to be called ".render".
 */
{template .render}
  <div>Hello World</div>
{/template}
```

SOY TEMPLATE COMPONENTS

- Or contain other Soy Template components:

```
{namespace Modal}

/**
 * This renders the component's whole content.
 * Note: has to be called ".render".
 */
{template .render}
  {call .header /}
  <div>Hello World</div>
  {call .button}
    {param label: "Okay"}
  {/call}
{/template}
```

WHY USE SOY TEMPLATES?

- **Convenience:** Soy Templates provide an easy way to build pieces of HTML for your application's UI and help you separate application logic from display.
- **Language-neutral:** Soy Templates work with JavaScript or Java. You can write one template and share it between your client-side and server-side code.
- **Client-side speed:** Soy Templates are compiled to efficiently run JavaScript functions for maximum client-side performance.
- **Easy to read:** You can clearly see the structure of the output HTML from the structure of the template code. Messages for translation are inline for extra readability.

BENEFITS OF USING SOY TEMPLATES

- **Designed for programmers:** Soy Templates are simply functions that can call each other. The syntax includes constructs familiar to programmers. You can put multiple templates in one source file.
- **A tool, not a framework:** Soy Templates work well in any web application environment in conjunction with any libraries, frameworks, or other tools.
- **Battle-tested:** Soy Templates are used extensively in some of the largest web applications in the world, including Gmail and Google Docs.
- **Secure:** Soy Templates are contextually autoescaped to reduce the risk of XSS.

RELATIONSHIP BETWEEN METAL.JS AND SOY TEMPLATES

- For the integration between Metal.js and Soy Templates to work, the Soy Template files need to be compiled via one of our available build tools.
- That's because they don't just compile the code, but also add information that helps with the integration (like export declarations).
- The available build tools that correctly compile Soy Templates for Metal.js are:
  - **Metal Tasks**: *http://npmjs.com/package/gulp-metal* (already included when creating project via generator-metal)
  - **metal-cli**: *http://npmjs.com/package/metal-cli*
  - **metal-tools-soy**: *http://npmjs.com/package/metal-tools-soy*

REGISTER SOY TEMPLATES FROM METAL.JS

- The only thing you need to do to use Soy Templates in your Metal.js component is to call Soy.register, passing it your component class and the Soy Templates you're going to use, like this:

```
import templates from './MyComponent.soy';
import Component from 'metal-component';
import Soy from 'metal-soy';

class MyComponent extends Component {
}
Soy.register(MyComponent, templates);

export default MyComponent;
```

CASE FOR SOY TEMPLATES

- Soy Templates provide reusable components to build your displays.
- Simple syntax is easy to learn.
- Advanced features like calling functions, manipulating data, and localizing messages are built in.
- Soy Templates is the recommended templating language for Metal.js.

Notes: