Social Component Framework – Getting social with AEM Communities

Siddharth Palaniswami | Senior Software Engineering Lead

Customer Pain Points – where it began

- Too much to do to get components working out of the box
- Extending functionality means large IT projects
- Long time to go live with AEM Communities
- Difficult to customize look and feel and UX
- Poor Re-usability
- Hard to integrate with other systems

Configuration > Customization > Extension

- **Easy to integrate** Out-of-the-Box with minimum to no customization for 80% of the use cases

- **Easy to customize and extend** functionality both on the server size and on the client side for the rest

SCF – Goals - Tenets

- Skinnable Markup and Stylesheets
  - Easily customize look and feel of components
- Simple Logic-less Templates
- Server side extensibility
  - Easy to add simple incremental business logic to existing components
  - Easy to customize/override OOTB functionality
- Client Side extensibility
  - Easy to customize component UX
- HTTP API
  - Any client, Mobile apps
- Client side rendering – fast, rich user experience
- SEO

What is the Social Component Framework

Social Component Framework (SCF) refers to the **collection of framework, patterns and conventions** that the new social communities components use. SCF aims at making social components easy to **customize**, **extend** and **reuse**.

Typical use-cases when customizing/extending components and SCF's solutions

ACME vs SCF - Use Case – Level 0

- Level 0 – add AEM Communities components to a page

- Solution
  - Drag and drop the components with ease
  - Simple setup and configuration

  - Powerful configuration options to maximize re-usability
  - Modular and well defined clientlibraries

SCF

SCF based Social Components

Forum Calendar Ratings

Reviews Voting Comments

…and more to come

ACME vs SCF - Use Case – Level 1

- Level 1 - the components just don't look right! …. **Blues** are not **reds**!!!

- Solution:
  - Add an SCF skin
    - Define a stylesheet that overrides the default CSS styles for SCF elements

SCF – CSS Contracts

- Well defined CSS classes for OOTB markups
- Semantic CSS class names
- Well scoped CSS
- CSS classes used for styles separate from those used as JS selectors

.scf button .scf-link .scf-composer-block .scf-comment-author

Easy to customize

ACME vs SCF - Use Case – Level 2

- Level 2 – I see Foo, Bar, Cat… but what I really want is Foo++ …
  - I want an ideation component, can I use or extend your component to build it?

- Solution:
  - Create custom components
  - Configure resource super types to point to OOTB resource types
  - Edit the templates to match the new component
  - Connect javascript models and views as necessary.

SCF – Simple/Reusable Templates

SCF based Social Components

- Handlebars Templates
- Stylesheets

JSP	Handlebars
Can get very complex very fast	Simple
Non reusable, java code seeps in	Logic less
Server side only	Server side and Client side
Overlays are hard because of complexity	Easy to overlay and customize
Tough to bind to javascript UX when overlaying	Naturally binds with client UX with client side rendering

Client Side Rendering | Skinnable | SEO Friendly | Easy to customize

SCF – JS Framework

- Backbone.js based Model/View framework for SCF Components
- Extensible and Reusable
- Modular
- HTTP API Client

data-scf-component = "acme/components/ideation/forum"

SCF.registerComponent('acme/components/ideation/forum', SCF.Forum, SCF.ForumView);

SCF.Model SCF.View

Easy to customize

Client Side Rendering

Client Side Extensible

Adobe

ACME vs SCF - Use Case – Level 2.1

- Level 2.1 –
  - Add Voting on Ideas
  - Remove voting from replies

- Solution
  - Use the “include” helper to add voting to topics
  - Edit the replies template to remove voting

```
{{include this.votes resourceType='social/tally/components/hbs/voting'}}
```

ACME vs SCF - Use Case – Level 3

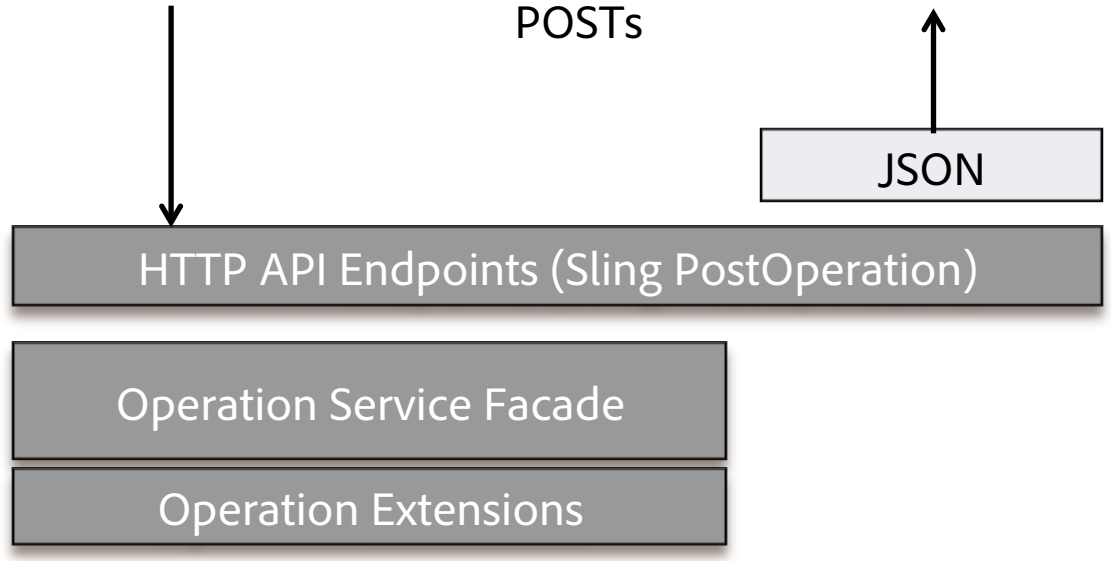
- Level 3 – Ideas need status that track the different states an idea can be in
  - Automatically add a "new" state to every idea

- Solution
  - Implement an OperationExtension (ForumOperationExtension)
  - Hook into any operation supported by SCF Components (ForumOperations.CREATE)
  - Add code to the beforeAction or afterAction to perform any custom logic
  - Use OperationServices when needed to perform any OOTB business logic

```
@Component(name = "Ideation Status Extension", immediate = true, metatype = true)
@Service
public class IdeationStatusExtension implements ForumOperationExtension {
@Override
   public List<ForumOperation> getOperationsToHookInto() {
      return Arrays.asList(ForumOperation.CREATE);
   }
@Override
   public void beforeAction(op, sessionUsed, requestResource, props)
      throws OperationException {..... }
}
```

SCF – Operation Services and OperationExtensions

- Define an endpoint by implementing a Sling PostOperation and extending AbstractSocialOperation.
  - :operation -> each SCF Operation
- Operations Services
  - Façade that wraps business logic and maintenance code for every operation
  - Accessible via servlet/operation or from anywhere else as an OSGi service
- Operation Extensions
  - Special hooks to plug into operations
    - Before and after actions



Easy to customize

Server Side Extensible

Adobe

ACME vs SCF - Use Case – Level 4

- Level 4 – Moderators should be able to move ideas between through its states

- Solution
  - Implement the AbstractSocialOperation to an HTTP endpoint that accepts a request and performs the action
  - Use the ForumOperationService to update the idea
  - Edit the template to add special inputs for enabling a moderator to perform the action
  - Extend the Javascript view and model to accept the user input, post it to the server and update the view.

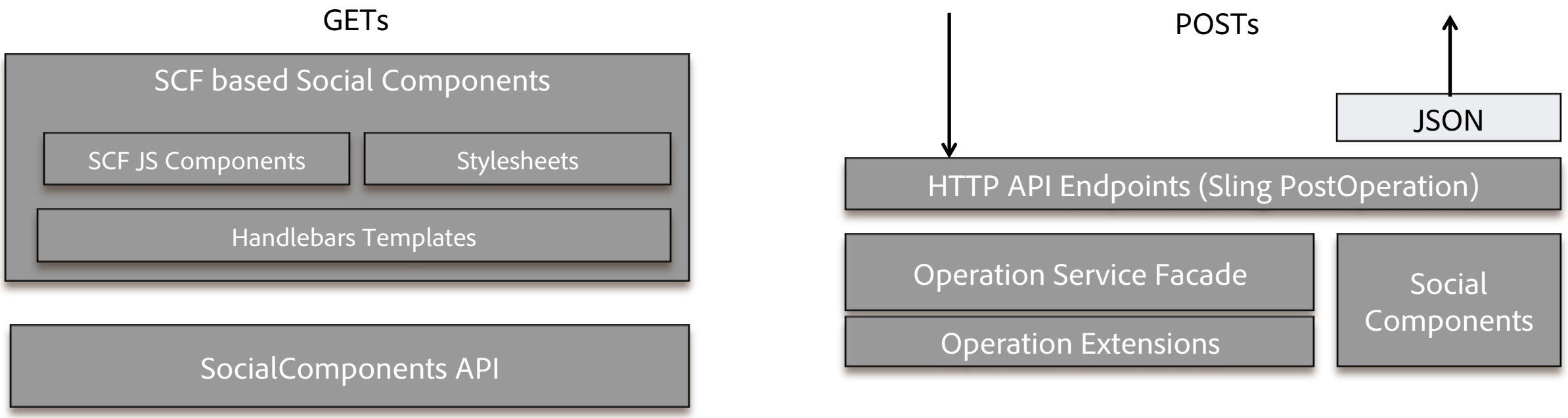
```
evt="click=setStatus"
var IdeaView = SCF.TopicView.extend({…})
var Idea = SCF.Topic.extend({…})
SCF.registerComponent('acme/components/ideation/topic', SCF.Idea, SCF.IdeaView);
```

ACME vs SCF - Use Case – Level 5

- Level 5 – but wait, I don't want the states showing up with the user tags. We need to filter them out!

- Solution
  - Implement a SocialComponent to represent an Idea
  - Extend provided abstract classes (AbstractPost) to inherit default behavior
  - Override methods that need to be customized(getTags())
  - Add a SocialComponentFactory that registers the IdeaSocialComponent with SCF so that it can be used every time a resource of type "idea" is requested for.

SCF – SocialComponent and SocialComponentFactory



- Client/template friendly POJOs that represents a resource
- One SocialComponent per resource type
- Automagically bound to template at render time (client/server)

- SocialComponentFactories register SocialComponents with the framework.
- Enables overriding/extending OOTB SocialComponents

Server side extensibility

Easy to create custom templates

ACME vs SCF - Use Case – Level 5.1

- Level 5.1 – Let's display the state of each idea when listing ideas.

- Solution
  - Add a few more getters to the IdeaSocialComponent so that the status of the idea is exposed to the clients
  - Edit the templates to display the status of an idea.

```
{{#if status}}<span class="idea-status">{{status}}</span>{{/if}}

public class IdeaSocialComponent extends AbstractPost implements Post {
...
public String getStatus() { .... }
...
}
```

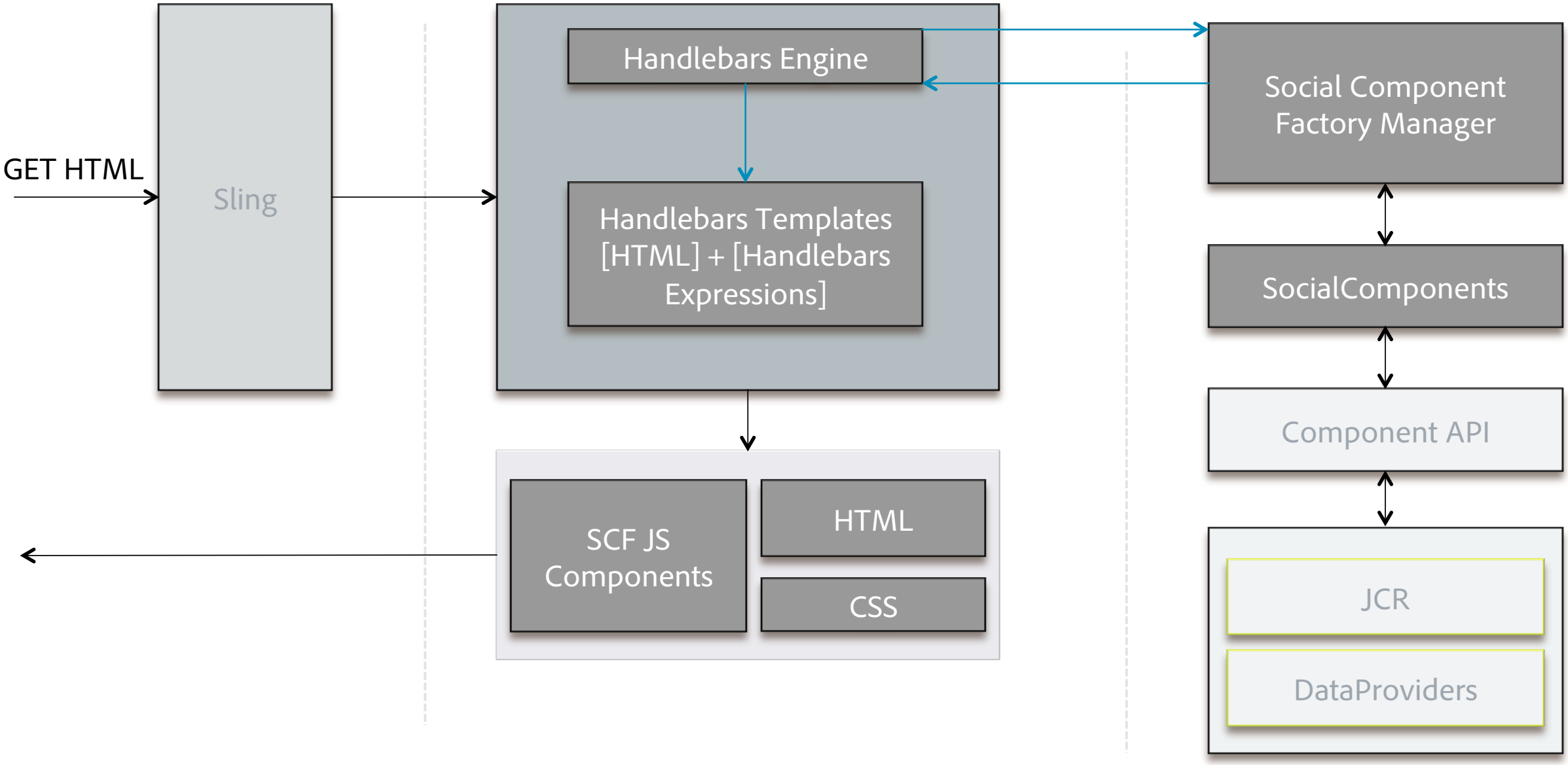
ACME vs SCF - Use Case – Level 6

- Level 6 - I would like to add filters to the idea lists to filter ideas by category

- Solution
  - Integrate with inbuilt search endpoints
    - Most components come with a well defined query able API
  - Edit the ideation forum template to include a list of filters that link to a filtered page

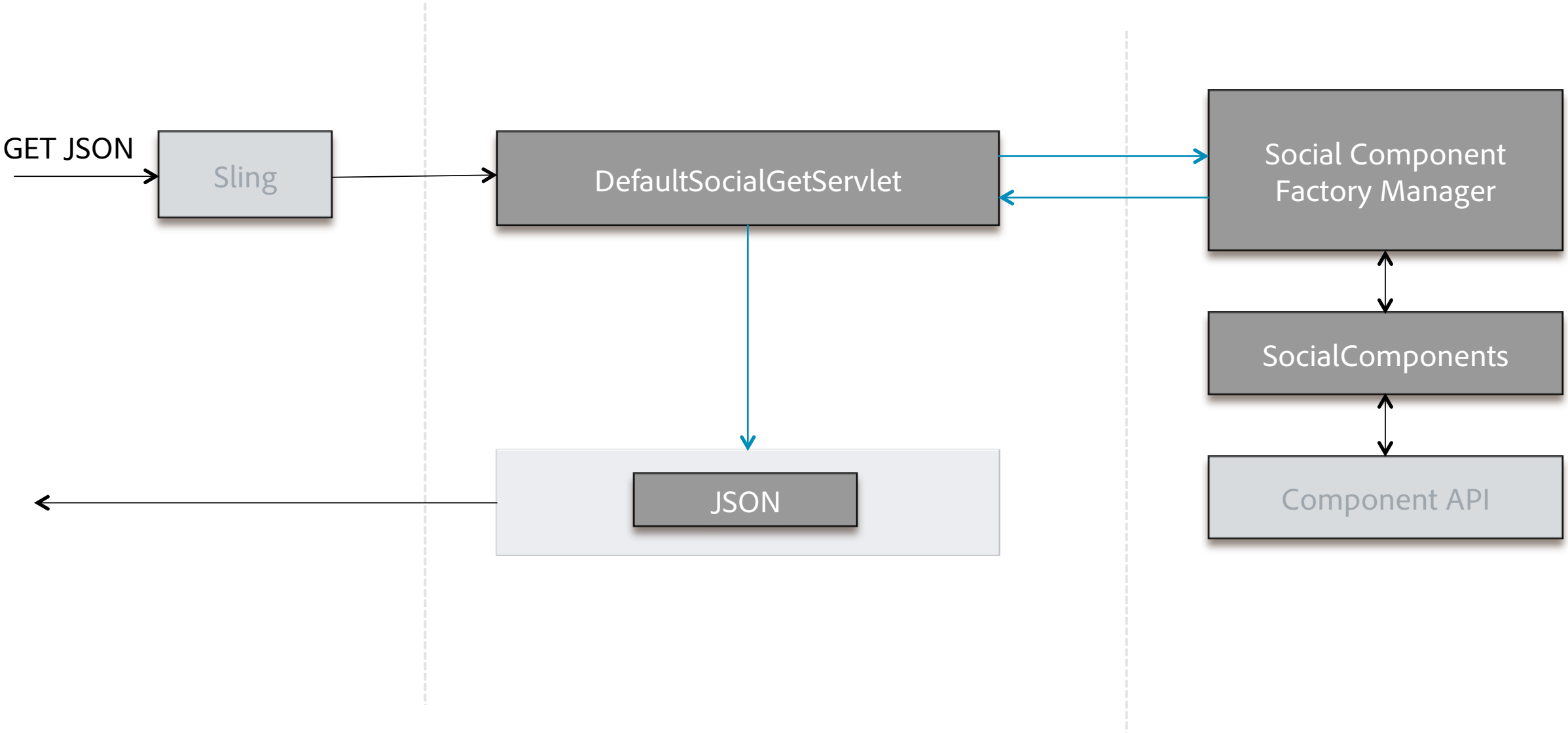
```
<a href='{{pageInfo.basePageURL}}.html?filter=tag like "products"' class="idea-filter">Products</a>
```

SCF– GET Request

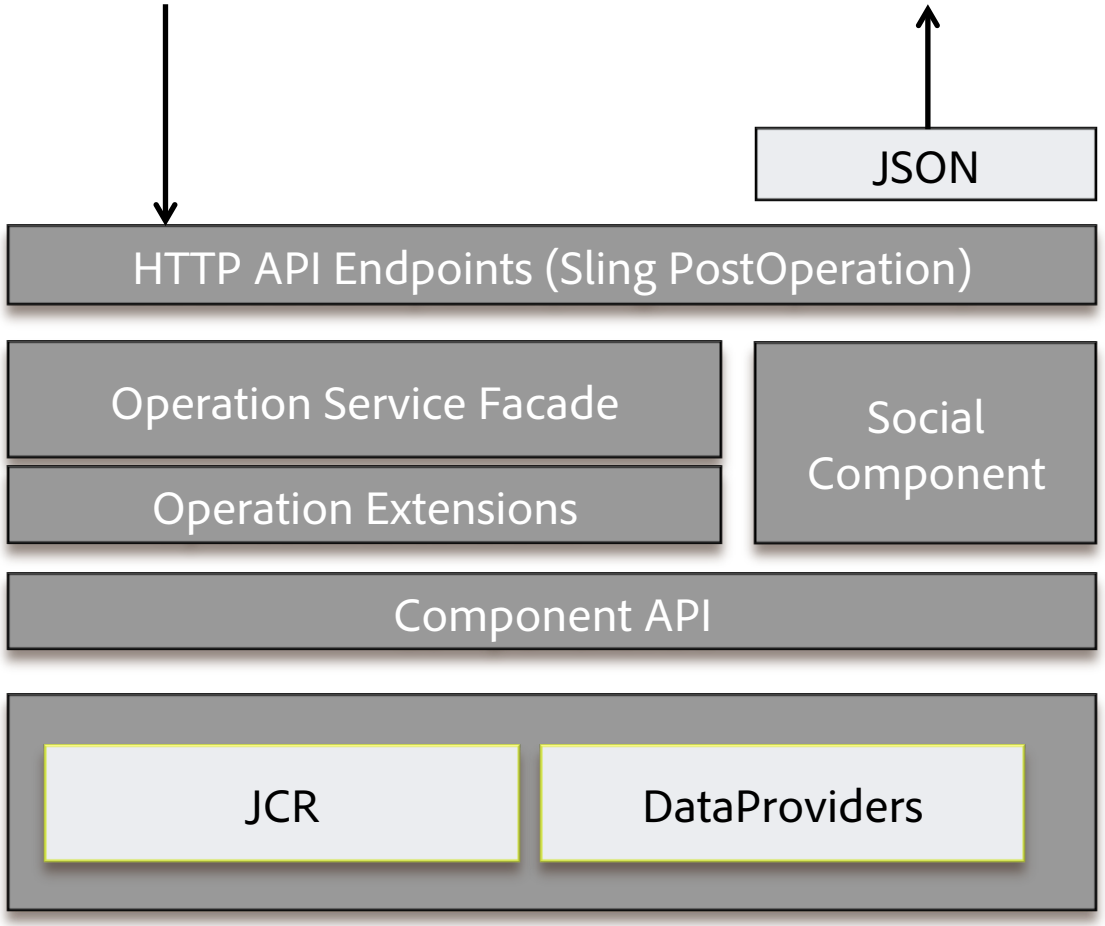


Adobe

SCF– GET HTTP API Request



SCF- POST Requests



SCF– Customizing/Extending OOTB Components - Recap

- Skinning – Level 1
  - Only need to change the CSS
- Look and Feel – Level 1-2
  - Change template and CSS
- Look and Feel and UX – Level 1,4
  - Change template, CSS and extend/override Javascript
- Make more/less information available to the template or to the GET endpoint – Level 5
  - Extend the OOTB SocialComponent (only need to add/subtract what you need)
- Add some custom processing during operations – Level 3
  - Write an OperationExtension
- Add a new custom operation – Level 4
  - Create new Sling Post Operation
  - Use existing Operation Services as needed
  - Extend Javascript Models and Views to invoke your operation when needed from the client side

Thank You!

Questions?

email: palanisw@adobe.com