

UI Overview

DataHub



Tech Stack

- **Typescript** - like JS with types
- **Ember** - our SPA framework
- **Yarn Workspaces** - multiple npm packages in a single location

Ember Overview

- **app** - This is where most code is
 - **routes** - routes handlers for our application
 - index.ts
 - **templates** - templates for the routes*
 - index.hbs
 - **components** - ui reusable components*
 - component1.hbs
 - component1.ts
 - **routes.ts** - path for the routes

Ember Addon Overview

- **app** - the contents of this folder will be **merge** with your host app, no actual code is written under this folder. Only re-export for components and services.
- **addon** - this is where most of the code for addons will live
 - **components**
 - **services**
 - ...

Example

Addon

- app
 - component
 - component1.js
 - service
 - service1.js
- addon
 - utils
 - util.ts
 - component
 - component1.ts
 - component1.hbs
 - service
 - service1.ts

+

App

- app
 - component
 - component2.ts
 - service
 - service2.ts

=

Result

- app
 - component
 - component1.js
 - component2.js
 - service
 - service1.js
 - service2.js

DataHub Folder Structure

- **package/data-portal** - main ember app
- **@datahub** - group all the DataHub functionality
- **@nacho-ui** - UI framework (buttons, tables) components
- **@dh-tools** - group some tooling like *.pdl to *.ts transformer
- **@yourcompany** - potentially where you should add your custom entities and code

Package/Data-Portal

- Main ember application
- It should be almost *empty* as all components, routes will come from addons
- Imports all addons, including @yourcompany custom addons

Thoughts

- How can we make the app more flexible to include dynamically custom addons?
 - Mitigated by allowing forking this part as it shouldn't contain code

In progress

- Move out old components and routes

@nacho-ui/nacho-core

- Custom UI Framework that we use for DataHub
- It contains from simple components like buttons

pills	Pill/Tag components
table	Generic table that allows dynamic table rendering
buttons	Primary, secondary buttons, etc...
dropdown	Dropdown/Selects wrappers that allow easy option selections
avatar	User related components (like picture or name with picture)
animation	Related animation styles and components like loading status

In progress

- Some nacho components still live under data-portal and should be moved to this library

@datahub

Util	Util package that contains utility functions and simple multipurpose components
Metadata-Types	Originally was serving pure model type definitions, however, since we introduced automatic model type generation, most of the code in this package is generated (in progress)..
Data-Models	Model abstractions Api calls Models configuration
Shared	All common functionally, components, routes that makes DataHub... DataHub :)
Entities	Top level addon that contains components that are only related to an entity

What is a generated model? (metadata-types)

```
interface Dataset {
  description: string;
  removed: boolean;
  deprecation?: Com.Linkedin.Dataset.DatasetDeprecation;
  datasetUpstreamLineage?: Com.Linkedin.Dataset.DatasetUpstreamLineage;
  entityTopUsage?: Com.Linkedin.Common.EntityTopUsage;
  follow?: Com.Linkedin.Common.Follow;
  health?: Com.Linkedin.Common.Health;
  institutionalMemory?: Com.Linkedin.Common.InstitutionalMemory;
  likes?: Com.Linkedin.Common.Likes;
  ownership?: Com.Linkedin.Common.Ownership;
  ownershipSuggestion?: Com.Linkedin.Common.OwnershipSuggestion;
  upstreamLineage?: Com.Linkedin.Dataset.UpstreamLineage;
  status?: Com.Linkedin.Common.Status;
  tags: string[];
  properties?: { [id: string]: string };
}
```

- It is the data that comes from the backend (GMA)

What is a model abstraction? (data-models)

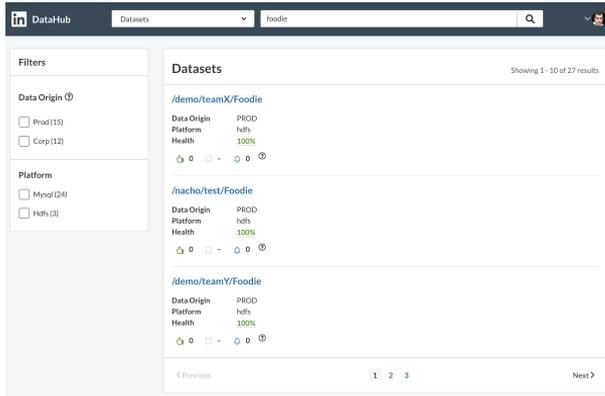
```
@statics<IBaseEntityStatics<Dataset>>()  
export class DatasetEntity extends BaseEntity<Dataset> {  
  
  static displayName: 'datasets' = 'datasets';  
  
  static get renderProps(): IEntityRenderProps {  
    return ...;  
  }  
  
  @oneWay('entity.description')  
  description?: string;  
}
```

- It is a wrapper around the backend data that allows abstractions on top and custom massaging
- It also defines configuration around this entity that will determine what to show in search, entity page and browsing (Render Props).

What is render-props? (data-models)

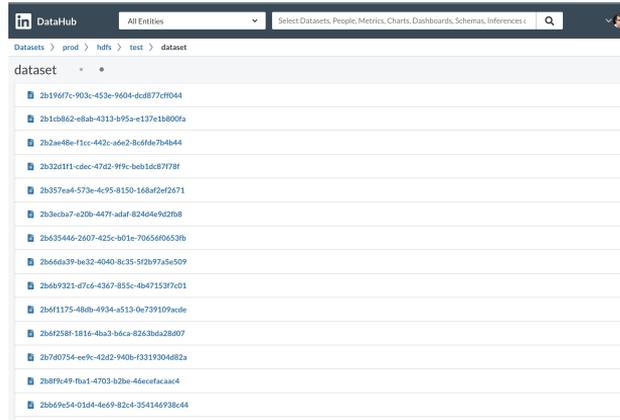
```
export interface IEntityRenderProps {  
  apiEntityName: string;  
  search: IEntityRenderPropsSearch;  
  browse?: IEntityRenderPropsBrowse;  
  entityPage?: IEntityRenderPropsEntityPage;  
}
```

- It defines how this entity should show in these sections:



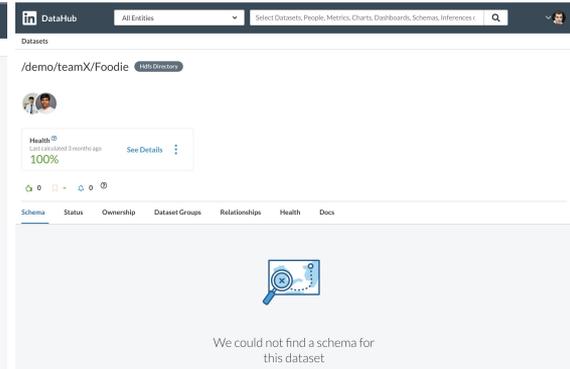
The screenshot shows the DataHub search interface. The search bar contains 'foodie'. On the left, there are filters for 'Data Origin' (Prod (13), Corp (12)) and 'Platform' (Mysql (24), Hdfs (3)). The main content area displays three dataset entries under the 'demo/teamX/Foodie' namespace. Each entry shows its Data Origin (PROD), Platform (hdfs), and Health (100%).

Search



The screenshot shows the DataHub browse view for the 'dataset' entity. The breadcrumb trail is 'DataHub > prod > hdfs > test > dataset'. The main content area displays a list of 15 dataset entries, each with a unique ID and a small icon.

Browse



The screenshot shows the DataHub entity page for 'demo/teamX/Foodie'. The breadcrumb trail is 'DataHub > All Entities > demo/teamX/Foodie'. The page displays a 'Health' indicator showing 'Last calculated 3 months ago' and '100%'. Below this, there are tabs for 'Schema', 'Status', 'Ownership', 'Dataset Groups', 'Relationships', 'Health', and 'Docs'. The main content area is currently empty, displaying a message: 'We could not find a schema for this dataset'.

Entity Page

What is render-props for search? (data-models)

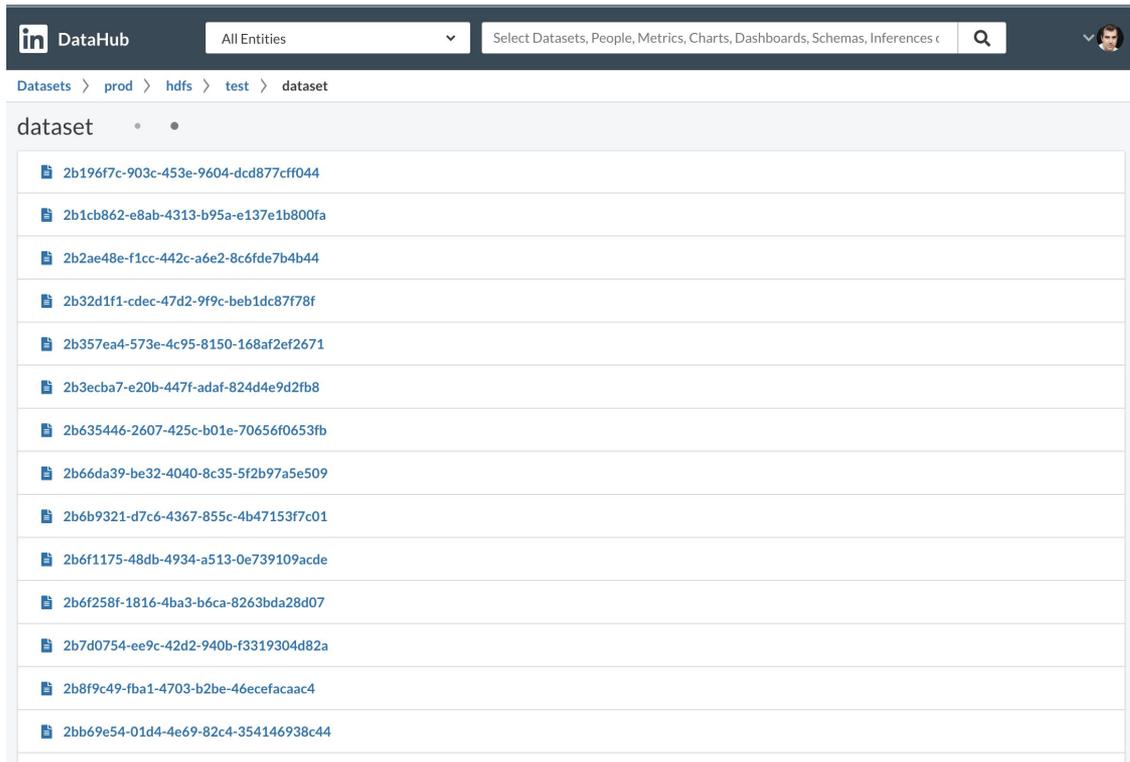
Desired filters

The screenshot shows the DataHub interface with a search for 'foodie'. The left sidebar contains filter sections: 'Data Origin' with 'Prod (15)' and 'Corp (12)' options, and 'Platform' with 'Mysql (24)' and 'Hdfs (3)' options. The main content area displays a list of datasets. Annotations highlight specific elements: a blue box around the 'Data Origin' filter section is labeled 'Desired filters'; a blue box around the 'Data Origin', 'Platform', and 'Health' properties for the first dataset is labeled 'Properties to show'; and a blue box around the interaction icons (thumbs up, bookmark, bell) for the first dataset is labeled 'Additional components you may want to render'. The dataset list includes entries like '/demo/teamX/Foodie', '/nacho/test/Foodie', and '/demo/teamY/Foodie', each with its own set of properties and interaction icons. The interface also shows a search bar, a user profile icon, and pagination controls at the bottom.

Properties to show

Additional components you may want to render

What is render-props for browse? (data-models)



The screenshot shows the DataHub interface. At the top, there is a navigation bar with the DataHub logo, a dropdown menu set to "All Entities", a search bar containing "Select Datasets, People, Metrics, Charts, Dashboards, Schemas, Inferences c", and a user profile icon. Below the navigation bar, a breadcrumb trail reads "Datasets > prod > hdfs > test > dataset". The main content area is titled "dataset" and displays a list of 15 dataset entries, each with a small icon and a long alphanumeric ID.

Dataset ID
2b196f7c-903c-453e-9604-dcd877cff044
2b1cb862-e8ab-4313-b95a-e137e1b800fa
2b2ae48e-f1cc-442c-a6e2-8c6fde7b4b44
2b32d1f1-cdec-47d2-9f9c-beb1dc87f78f
2b357ea4-573e-4c95-8150-168af2ef2671
2b3ecba7-e20b-447f-adaf-824d4e9d2fb8
2b635446-2607-425c-b01e-70656f0653fb
2b66da39-be32-4040-8c35-5f2b97a5e509
2b6b9321-d7c6-4367-855c-4b47153f7c01
2b6f1175-48db-4934-a513-0e739109acde
2b6f258f-1816-4ba3-b6ca-8263bda28d07
2b7d0754-ee9c-42d2-940b-f3319304d82a
2b8f9c49-fba1-4703-b2be-46ecefacaac4
2bb69e54-01d4-4e69-82c4-354146938c44

Not much to customize for now, but you can enable or disable search, as hierarchy comes from ES (elastic search)

What is render-props for entity page? (data-models)



The screenshot shows the DataHub interface for an entity named 'Health'. The page is annotated with several callouts:

- Breadcrumbs if browse is enabled:** A box highlights the breadcrumb path `/demo/teamX/Foodie`.
- Tags to show:** A box highlights a tag labeled `Hdfs Directory`.
- Additional custom header components for this entity:** A box highlights a 'Health' status card showing 'Last calculated 3 months ago' and '100%' with a 'See Details' link.
- Available tabs and its contents:** A large rounded box highlights the 'Schema' tab, which contains a message: 'We could not find a schema for this dataset'.

Available tabs and its contents. It can be a completely custom component or a existing component like a table

Components in @datahub/shared

Components that are shareable across the app. For example:

@datahub/shared/addon/components/social/social-action.ts



Entity components in @datahub/entities

Custom components that are only valid for specific entity. For example:

`@datahub/entities/addon/components/datasets/dataset-schema.ts`

Table

JSON

Last modified: 7/30/2020, 4:11:57 AM

Column	Data Type	Default Comments
key:	STRING	
firstName	STRING	
lastName	STRING	
age	INT	

How to customize DataHub?



Good topic for next session!

Sneak peak: Add your own package with your own:

- Custom components (if needed as we provide some generic ones)
- Models (what's the shape of the data)
- Configuration (what to show where)

What are we improving?



Finalize all migrations like moving data-portal code to @datahub

Automatic model generation from open source

Update APIs (mid tier)

Move to open source first

Thank you