Plan de test

Programmation de composants Angular pour la mise en œuvre de tests utilisateurs

Benjamin Mitton

Cédric Pinard

Université Grenoble Alpes

Table des matières

Introduction

Ce document décrit l'ensemble des tests à réaliser sur le produit de notre projet. Dans un premier temps nous présenterons les tests unitaires et dans un deuxième temps les tests d'intégration. L'aspect visuel de notre projet, implique que les tests soient réalisés et validés manuellement.

Objectifs et méthodes

Le but de ces tests est de pouvoir vérifier que l'ensemble des fonctionnalités attendues par le client soit non seulement conforme visuellement mais aussi fonctionnellement.

L'environnement de test est un projet Angular utilisant le produit de notre projet, la librairie basic-linechart, disponible sur npm. Pour tester chaque fonctionnalité, nous allons créer une linechart avec la libraire en précisant les paramètres nécessaires pour l'observer. Les données utilisées sont disponibles en annexe.

Description des tests unitaires

Test : Afficher une simple linechart sans paramètres

Mise en œuvre

On utilise la balise d'appel à la librairie avec comme seul paramètre les data à afficher :

<basic-linechart data=data></basic-linechart>

Données en entrée

Data contient les valeurs, le type d'interpolation, la couleur, le nom du composant ainsi que le style d'affichage.

Ici les valeurs sont celles du capteur PC6 (cf annexe 4), la couleur est #123568, le nom PC6, le style both et l'interpolation step.

Résultats attendus

Il doit donc s'afficher une linechart avec une seule courbe avec le nom PC6 (les autres paramètres seront regarder plus tard) cette ligne doit représenter des données booléennes.

Test : Afficher une multi linechart sans paramètres

Mise en œuvre

On utilise la balise HTML de la librairie avec comme seul paramètre, les données (data) à afficher :

<basic-linechart data=data></basic-linechart>

Données en entrée

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

Du capteur PC6 (cf annexe 4), la couleur est #123568, le nom PC6, le style both et l’interpolation step.

Du capteur Présence_Salon (cf annexe 4), la couleur est 'pink', le nom Présence_Salon, le style line et l’interpolation step.

Du capteur PC5 (cg annexe 4), la couleur est 'pur', le nom est PC5, le style est line et l’interpolation step.

Résultats attendus

Il doit donc s’afficher un composant de linechart avec trois courbes, de bonne couleur et de type de données correspondant. La courbe du capteur PC5 doit s'afficher en noir car la couleur 'pur', passé en paramètre, n'est pas reconnue comme un code couleur.

Test : Afficher une linechart avec comme style : line | aréa | both

Mise en œuvre

On utilise la balise HTML de la librairie avec comme paramètre, les données (data) à afficher. On ajoute 3 boutons qui, au clic, appellent une fonction qui change la valeur de data. Chaque bouton permet d'afficher un style de courbe : ligne, aire ou les deux.

<basic-linechart data=data></basic-linechart>

Données en entrée

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

Du capteur Température_Cuisine, la couleur est 'purple', le nom est Température_Cuisine, l’interpolation est step. Le style sera donc modifié, on retrouve donc line, area ou both.

Résultats attendus

Il doit donc s’afficher un composant contenant une courbe de données de couleur violette avec le titre Température_Cuisine et à chaque clic sur un des boutons le style de la courbe doit changer sans modifier les données celle-ci.

Test : Afficher une linechart avec comme interpolation : step | linear

Mise en œuvre

On utilise la balise HTML de la librairie avec comme paramètre, les données (data) à afficher. On ajoute 2 boutons qui, au clic, appellent une fonction qui change la valeur de data. Chaque bouton permet d'afficher une interpolation de courbe : linéaire ou en escalier.

<basic-linechart data=data></basic-linechart>

Données en entrée :

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

Du capteur Température_Cuisine, la couleur est 'yellow', le nom est Température_Cuisine, le style est line, l’interpolation est donc soit step soit linear

Résultats attendus :

Il doit donc s’afficher un composant contenant une courbe de données de couleur jaune avec le titre Température_Cuisine et a chaque clic sur un des boutons l’interpolation de la courbe doit changer sans modifier les données de celle-ci.

Test : Afficher une linechart qui réagit dynamiquement au changement de type de données

Mise en œuvre

On utilise la balise HTML de la librairie avec comme paramètre les données (data) à afficher. On ajoute 3 boutons qui, au clic, appellent une fonction qui change la valeur de data. Chaque bouton permet d'afficher une type de données : booléen, flottant ou entier positif.

<basic-linechart data=data></basic-linechart>

Données en entrée :

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

Du capteur Température_Cuisine contenant des valeurs flottantes, la couleur est 'yellow', le nom est Température_Cuisine,le style est line, l’interpolation est linear

Du capteur Présence_Cuisine contenant des valeurs booléennes, la couleur est 'purple', le nom est Présence_Cuisine, le style est line, l’interpolation est step.

Des valeurs aléatoires entières, la couleur est 'black', le nom est PC5, le style est line et l’interpolation est step.

Résultats attendus :

Il doit s’afficher un composant contenant une courbe de données flottantes de couleur jaune si le bouton flottant est cliqué, une courbe de données booléennes de couleur violette si le bouton booléen est cliqué et une courbe de données positives discrètes de couleur noir si le bouton int est cliqué.

Test : Afficher une linechart avec un domaine

Mise en œuvre

On utilise la balise HTML de la librairie avec comme paramètres, les données (data) à afficher et le paramètre domain.

<basic-linechart data=data domain=[min,max]></basic-linechart>

Données en entrée :

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

Du capteur Température_Salon contenant des valeurs flottantes, la couleur est 'purple', le nom est Température_Salon , le style est line, l’interpolation est step.

Domain est un tableau de deux entier un pour chaque borne.

Ici les valeurs sont [0,30]

Résultats attendus :

Il doit s’afficher une linechart contenant une seule courbe de couleur violette. Cette courbe doit être adapter à l'échelle de l’axe, déterminée par le domaine en entrée. L'échelle doit donc aller de 0 à 30.

Test : Afficher une linechart avec une taille différente

Mise en œuvre

On utilise la balise HTML de la librairie avec comme paramètres, les données (data) à afficher, la largeur (width) et la hauteur (height).

<basic-linechart data=data width=a height=b></basic-linechart>

Données en entrée :

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

Du capteur Température_Salon contenant des valeurs flottantes, la couleur est 'purple', le nom est Température_Salon, le style est line, l’interpolation est step.

Width et height prennent comme valeur un entier qui représente une taille en pixel (px).

Ici les valeurs sont :

width = 1500

heigth = 500

Résultats attendus :

Il doit s’afficher une linechart contenant une seule courbe de couleur violette. Cette fois le composant doit faire une taille de 1500px par 500px vérifiable avec l’option inspecteur de code du navigateur.

Test : Vérifier la bulle d'information (tooltip)

Mise en œuvre

On utilise la balise HTML de la librairie avec comme paramètre, les données (data) à afficher. On ajoute 3 boutons qui, au clic, appellent une fonction qui change la valeur de data. Chaque bouton permet d'afficher une type de données : booléen, flottant ou entier positif. Ensuite, on doit déplacer la souris au sein du composant pour voir le tooltip s’afficher.

<basic-linechart data=data></basic-linechart>

Données en entrée :

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

De str2

De str3

De str4

Ces données sont construites de manières a être facilement lisible pour permettre de réparer une erreur du tooltip facilement.

Résultats attendus :

Il doit s’afficher une linechart ou les données correspondent à celles affichées dans le tooltip. C’est à dire que si le curseur de la souris pointe sur la quatrième valeur de la courbe, c’est bien la quatrième valeur qui est affichée dans le tooltip et ceci peu importe le type de données qui est affiché. On doit aussi vérifier que les données s'affichent correctement dans le tooltip.

Test : Vérifier le fonctionnement du zoom et de la scrollbar

Mise en œuvre

On utilise la balise HTML de la librairie avec comme paramètre, les données (data) à afficher. On ajoute 2 boutons qui, au clic, appel une fonction qui change la valeur de data. Un bouton pour afficher une courbe seule et un pour afficher plusieurs courber. Ensuite, on doit naviguer au sein du composant, en utilisant la molette de la souris pour zoomer et en déplaçant (drag and drop) la barre horizontale (scrollbar) pour vérifier leur fonctionnement.

<basic-linechart data=data></basic-linechart>

Données en entrée :

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

De str2

De str3

De str4

Résultats attendus :

Il doit s’afficher une linechart où l’on peut zoomer sans détruire les données, c'est à dire agrandir ou réduire la plage de temps que l'on affiche. Où l'on peut déplacer la scrollbar pour naviguer horizontalement dans les données tout en gardant le même indice de zoom, c'est à dire déplacer la plage de temps sans l'agrandir ou la réduire. La scrollbar ne doit pas disparaître sur les côtés.

Test : Vérifier le déplacement de la barre verticale représentant le temps actuel (currentTime)

Mise en œuvre

On utilise la balise HTML de la librairie avec comme paramètre, les données (data) à afficher. On maintient le clique sur le sélecteur en haut de la ligne (cercle rouge) pour la déplacer (drag and drop).

<basic-linechart data=data></basic-linechart>

Données en entrée :

Data comporte les valeurs, le type d’interpolation, la couleur, le nom du composant ainsi que le style d’affichage.

Ici les valeurs sont celles :

Du capteur PC6 (cf annexe 4) , la couleur est #123568, le nom PC6, le style est both et l’interpolation step.

Du capteur Présence_Salon (cf annexe 4), la couleur est 'pink', le nom Présence_Salon, le style est line et l’interpolation step.

Du capteur PC5 (cg annexe 4), la couleur est 'pur', le nom est PC5, le style est line et l’interpolation step.

Résultats attendus :

On peut déplacer la barre dans les deux composants affichés. La barre disparaît si elle sort de la plage de temps affichée, lors d'un zoom ou d'un déplacement de la scrollbar, mais n'est pas détruite.

Description des tests d’intégrations

Test : Synchronisation de la plage de temps (range) et la barre verticale représentant le temps actuel (currentTime)

Description du test

Pour les tests d’intégrations, nous allons utiliser plusieurs balises <basic-linechart> qui interagissent entre elles afin de vérifier qu’elles communiquent correctement en entrée et en sortie. Il faut vérifier que les balises soient synchronisables sur l’affichage des plages de données et sur l'affichage de la barre verticale currentTime, lors de l'utilisation du zoom, de la scrollbar et du déplacement de la barre verticale currentTime.

But du test

On cherche à prouver que les différentes balises peuvent correctement s'intégrer au sein d’un projet et que les entrées et les sorties qui les définissent fonctionnent correctement.

Principe de réalisation

On utilise deux balise <basic-linechart> contenant les attributs suivants :

-data : les données sous le modèle présenté dans les tests unitaires

-range : un tableau de deux entiers (min et max) représentant les bornes de la plage de temps affichée

-rangechange : un output qui permet d’émettre la valeur de la plage de temps

-currentTime : un entier (timestamp) qui détermine la position de la barre verticale dans la plage de temps affichée

-currentTimeChange : un output qui permet d'émettre la valeur de la barre verticale

On utilise les mêmes paramètres pour les deux linechart c’est à dire :

Les données utilisées sont data : [Temperature_Salon,violet,line,linear]

On ajoute deux fonctions à chaque composant

-updateRange qui modifie la valeur de range du composant

-updateCurrentTime qui modifie la valeur de la barre verticale currentTime du composant

Il ne reste alors qu’à vérifier sur le composant, qu’en déplaçant la currentTime dans l’un il se passe la même chose dans l’autre, pareil pour la plage de temps lorsque l'on utilise le zoom et la scrollbar.

On change aussi les données pour avoir des données différentes entres les deux courbes et on refait les mêmes manipulations.

Test : Déplacement de la barre verticale de temps

Description du test

La position de la barre verticale currentTime est déterminée par un input, il est donc possible dans un plus gros projet de rendre automatique le déplacement de celle-ci. Il faut donc créer une fonction qui déplace la barre d’une certaine valeur en fonction du temps qui s’écoule.

But du test

Le but du test est de montrer qu’il est possible de faire réagir dynamiquement le composant par rapport à des données. Qu’il est donc possible de faire interagir le composant avec une simulation d'acteurs (capteurs et effecteurs).

Principe de réalisation

On utilise une seule balise <basic-linechart> qui contient les attributs range, currentTime. On ajoute deux boutons, lire et arrêter qui, au clic, appelleront une fonction.

La fonction lire() vient modifier la valeur de currentTime du composant à chaque t donnée.

La fonction arrêter() arrête la fonction lire().

Il suffit alors de regarder si la barre se déplace correctement sur la courbe par rapport au temps fixé et au pas de déplacement dans les données.

Test : Synchronisation et déplacement auto de la currentTime

Description du test

On va tester l’ensemble des possibilités d’intégrations du composant, il faut donc lier deux composants. Les deux composants affichent des données différentes. Il suffit ensuite de venir tester l’ensemble des fonctionnalités testées lors des test unitaires pour vérifier que les composants fonctionnent correctement.

But du test

Le but de ce test est de vérifier que toutes les interactions possibles, entre les linecharts, fonctionnent correctement même avec différents types de données. On peut ainsi observer la barre verticale currentTime, le zoom et la scrollbar synchronisés pendant que la currentTime se déplace aussi en automatique. Si le test est correct c’est que le produit remplit l’ensemble des fonctionnalités demandées par le client.

Principe de réalisation

On utilise deux balises <basic-linechart> l’une contient des données booléennes, l’autre des données flottantes. On rajoute les deux boutons lire et arrêter.

Les données d’entrées sont les suivantes :

data1 : [temperature_salon ,violet,line,linear]

data2 : [PC6,bleu,both,step]

On utilise les attributs suivants :

-range : un tableau de deux entiers (min et max) représentant les bornes de la plage de temps affichée

-rangechange : un output qui permet d’émettre la valeur de la plage de temps

-currentTime : un entier (timestamp) qui détermine la position de la barre verticale dans la plage de temps affichée

-currentTimeChange : un output qui permet d'émettre la valeur de la barre verticale

Annexe

```
str3 : string  =`
"2016-07-25 15:46:53,000";"SPA";"10.5"
"2016-07-25 15:47:53,100";"SPA";"20.5"
"2016-07-25 15:48:53,200";"SPA";"30.5"
"2016-07-25 15:49:53,300";"SPA";"40.5"
"2016-07-25 15:50:53,400";"SPA";"50.5"
"2016-07-25 15:51:53,500";"SPA";"60.5"
"2016-07-25 15:52:53,600";"SPA";"70.5"
"2016-07-25 15:53:53,700";"SPA";"80.5"
"2016-07-25 15:54:53,800";"SPA";"90.5"
"2016-07-25 15:55:53,900";"SPA";"100.5"
"2016-07-25 15:56:54,000";"SPA";"110.5"
"2016-07-25 15:57:54,100";"SPA";"120.5"
"2016-07-25 15:58:54,200";"SPA";"130.5" `

str4 : string  =`
"2016-07-25 15:46:53,000";"SPA";"OFF"
"2016-07-25 15:47:53,100";"SPA";"OFF"
"2016-07-25 15:48:53,200";"SPA";"OFF"
"2016-07-25 15:49:53,300";"SPA";"ON"
"2016-07-25 15:50:53,400";"SPA";"OFF"
"2016-07-25 15:51:53,500";"SPA";"OFF"
"2016-07-25 15:52:53,600";"SPA";"OFF"
"2016-07-25 15:53:53,700";"SPA";"ON"
"2016-07-25 15:54:53,800";"SPA";"OFF"
"2016-07-25 15:55:53,900";"SPA";"OFF"
"2016-07-25 15:56:54,000";"SPA";"OFF"
"2016-07-25 15:57:54,100";"SPA";"ON"
"2016-07-25 15:58:54,200";"SPA";"OFF" `


str2 : string  =`
 "2016-07-25 15:46:53,000";"SPA";"1"
 "2016-07-25 15:47:53,100";"SPA";"2"
 "2016-07-25 15:48:53,200";"SPA";"3"
 "2016-07-25 15:49:53,300";"SPA";"4"
 "2016-07-25 15:50:53,400";"SPA";"5"
 "2016-07-25 15:51:53,500";"SPA";"6"
 "2016-07-25 15:52:53,600";"SPA";"7"
 "2016-07-25 15:53:53,700";"SPA";"8"
 "2016-07-25 15:54:53,800";"SPA";"9"
 "2016-07-25 15:55:53,900";"SPA";"10"
 "2016-07-25 15:56:54,000";"SPA";"11"
 "2016-07-25 15:57:54,100";"SPA";"12"
 "2016-07-25 15:58:54,200";"SPA";"13" `
```

```
str : string =  `
"2016-07-25 15:46:53,910";"ikettle_temp";"86.7"
"2016-07-25 15:46:56,302";"I1";"1.24482346"
"2016-07-25 15:46:56,302";"PF";"1.29077506"
"2016-07-25 15:46:56,302";"V1N";"237.304749"
"2016-07-25 15:46:56,318";"P_active";"0.209506989"
"2016-07-25 15:46:56,318";"Freq_totale";"49.9725494"
"2016-07-25 15:47:11,767";"ikettle_temp";"87.2"
"2016-07-25 15:47:16,286";"I1";"1.21982718"
"2016-07-25 15:47:16,301";"PF";"1.29786205"
"2016-07-25 15:47:16,301";"V1N";"237.287674"
"2016-07-25 15:47:16,317";"P_active";"0.203233793"
"2016-07-25 15:47:16,317";"Freq_totale";"49.9884491"
"2016-07-25 15:47:18,970";"Temperature_SDB";"27.78"
"2016-07-25 15:47:19,079";"Leds_CO2_SDB";"OFF"
"2016-07-25 15:47:19,204";"CO2_Cuisine";"298.88"
"2016-07-25 15:47:19,298";"Humidite_Cuisine";"66"
"2016-07-25 15:47:19,423";"Temperature_Cuisine";"26.7"
"2016-07-25 15:47:19,517";"Leds_CO2_Cuisine";"OFF"
"2016-07-25 15:47:19,642";"CO2_Salon";"654.72"
"2016-07-25 15:47:19,751";"Humidite_Salon";"66"
"2016-07-25 15:47:19,845";"Temperature_Salon";"26.34"
"2016-07-25 15:47:19,954";"Leds_CO2_Salon";"OFF"
"2016-07-25 15:47:20,079";"CO2_Chambre";"670.72"
"2016-07-25 15:47:20,173";"Humidite_Chambre";"57"
"2016-07-25 15:47:20,283";"Temperature_Chambre";"28.48"
"2016-07-25 15:47:20,392";"Leds_CO2_Chambre";"OFF"
"2016-07-25 15:47:20,533";"Presence_Cuisine";"OFF"
"2016-07-25 15:47:20,662";"Luminosite_Cuisine";"55.0"
"2016-07-25 15:47:20,787";"Presence_Table";"OFF"
"2016-07-25 15:47:20,912";"Luminosite_Table";"196.96"
"2016-07-25 15:47:21,021";"Presence_Salon";"OFF"
"2016-07-25 15:47:21,162";"Luminosite_Salon";"482.88"
"2016-07-25 15:47:21,271";"Presence_Lit";"OFF"
"2016-07-25 15:47:21,396";"Luminosite_Lit";"60.0"
"2016-07-25 15:47:21,537";"Presence_Bureau";"OFF"
"2016-07-25 15:47:21,662";"Luminosite_Bureau";"25.0"
```