Rapport – **Från koncept till prototyp**

Mitt bibliotek

Jag har använt mig av det externa biblioteket React Router (react-router-dom). Användningsområdet är att kunna skapa specifika url:er för olika sidor i SPA-applikationen. I normala fall ändrar inte en SPA-applikation url:en när en användare upplever att sidan ändras, utan det är bara koden som uppdaterar elementen på sidan (DOM:en). Med React Router kan jag välja att skapa en url, för en specifik react-komponent och vy.

Om extern datalagring och React

Det viktiga med att använda externa lagringskällor utanför React, är att koppla dem till komponenternas tillstånd. Att på det viset kontrollera när data ska hämtas beroende på komponentens livscykel. Ibland vill vi att data ska hämtas innan komponenten ritats ut och ibland kanske efter.

Det är också viktigt att förstå att React jobbar asynkront. Det innebär att React inte automatiskt väntar på att extern data har hämtats, utan vi måste styra det själva. Gäller det API:er kan dat vara med async/await eller .then, och är det lokala datalagringsplatser kan det hanteras med ”conditional rendering” eller en ”looding boolean” för ett specifikt state.

Om URL:en

Url:en blir som bokmärken för en användare av applikationen. Url:en går att dela till andra användare som referens, men även för att själv spara sidor i t.ex. webbläsaren. Använder vi Url:en som lagringsplats är det därför viktigt att inte byta eller förändra url:er så att de inte längre innehåller rätt information. Det kan liknas vid en url-förfrågan till ett API. Har vi i api:et ändrat informationen för just den url:en, så får vi fel information.

Mitt val av bibliotek

För en webbapplikation så är url:en mycket viktig. Som ovan avsnitt beskriver, så visar den var vi befinner oss i en webbapplikation. Det behöver inte betyda att den måste finnas, då webbapplikationerna mer och med liknar lokala applikationer på olika enheter. Men oftast så förknippar en användare en hemsida/webb-app med olika url-adresser för olika sidor, och att de går att använda som bokmärken eller att det går att använda ”bläddra” pilarna i webbläsaren för att ta sig fram och tillbaka i applikationen.

Eftersom React är ett verktyg för att skapa SPA-applikationer så sker det inte automatiskt att varje ny sida (som användaren upplever det) också har en specifik url. Av det skälet blev det ett självklart val att använda mig av React Router som just hjälper mig att skapa specifika url:er till olika React-komponenter.
Paketet har många användbara funktioner, t.ex. nästlade routes och dynamiska segment. Vilket gör att det går att skapa url:er där bara den sista delen renderas och byts ut i React. Paketet är väl genomtänkt och implementeras lätt i koden. Det ger en strukturerad uppbyggnad, skapar bra kodstruktur samt tydlighet för användaren.
Paketet är otroligt populärt (8 milj nedladdningar i veckan) och visar därmed att det är ett bra bibliotek, även om jag inte vet och kan allt om innehåll ännu.
Licensen är MIT vilket innebär att det går att använda i princip fritt både privat och kommersiellt. Paketet är utgivet av Remix som verkar vara en stor organisation, vilket också ger förtroende för att det kommer fortsätta utvecklas, och hållas uppdaterat.

Min kodstruktur

Jag har använt mig av designmönstret MVC. Den logiska delen av koden ligger i mappen **Model**. Här finns t.ex. kommunikationen med det externa REST-apiet. Jag har också separerat url:erna till apiet i en egen fil så blir det enklare att byta url, om koden skulle växa och samma url används på flera ställen. Det finns en Data-mapp, men den innehåller bara filer för det simulerade api:et (json-server), men i detta fallet passar den också bra där.
Jag har två mappar som representerar **Views.** Det är Components och Pages. Båda innehåller React komponenter, men för att fördela sidor och komponenter med de olika routes som jag använder, så har jag separerat de i två mappar. Möjligt att jag får tänka om när projektet växer, men så går mina tankar. Representationsmappar för **Controller** var svårare att definiera. Jag gjorde därför till en början en mapp som jag kallar Controller och som bara innehåller Reacts renderfunktion samt React Routers struktur med routes. Det var den komponent som jag tycker kommer närmast tankarna kring Controllers. Dock kan det säkert bli så att även det kommer ”ge sig” när projektet växer.
I övrigt så har jag en mapp ”Helpers” med filer som varken är komponenter eller tillhör Model. Det är moduler som kan återanvändas och justeras utan att man behöver leta och i alla komponenter. Sedan också en mapp (images) med bilder, men den skulle också kunna byta namn senare till t.ex. (resources) som i sin tur innehåller images och fler resursmappar.

Även om det är lite svårt att applicera Reacts arbetssätt på just MVC, så har det ändå många likheter. Det är också ett känt designmönster och försöker vi hålla oss till designmönster som många känner till, blir strukturen mer lättläst för andra utvecklare. Även om inte allt ligger på rätt ställe från början, finns MVC som ett mål i designtänket och kommer sakta formas till ett bra ock lättläst mönster. Skapar man en modul och funderar på vart den hör hemma i mönstret och lägger den där, så håller man en struktur från början.

Min prototyps externa datalagring

I min applikation finns det tre olika platser för extern datalagring. Jag använder Local Storage för att spara en inloggad användare och tänker också använda samma datalagringsplats för att spara ett pågående arbete med en layout, så den inte försvinner så fort komponenter ”unmountar” eller man uppdaterar webbläsaren.

Jag använder också url:en som lagringsplats för olika sidor, eller rättare sagt ”mountade” komponenter. Detta görs med React Router, och exakt på vilket sätt det funkar kan jag inte svara på. Lagringsplatsen skulle också kunna utvecklas till att visa olika layouter beroende på url-parameter. Då kan användaren dela enkelt med någon annan.

Min tredje datalagringsplats är API:et. Det är det största och det som används för långvarig förvaring. Här sparas just nu layouter samt information om de resurser som visas i content-browser, under knappen ”marknadsmaterial”. Eftersom jag i prototypen använder ett simulerat api med json-server och json-sever bara stödjer ett lager ner i ett nästat json-objekt, så kan jag inte specificera url:en för att hämta ett specifikt längre ner i det stora objektet. I detta fallet så använder jag därför ”.filter” och ”.map” för att få ut objektet som jag vill ha. Det är snarare ett krångligare sätt än att hämta rätt information direkt med url-parameter, men jag lyckades inte att lösa specifika routes för json-server inför deadline, så därför är det så just nu. Sedan är tanken att api:et är grunden för hela applikationen. Här kommer information om användaren sparas, inloggningar, företag, delat material och mycket mer.