Storing router state

Vojtech Mašek

Head dev @FlowUp

/ vmasek

/ VojtechMasek

/ vmasek

What are the benefits?

- Overview of router state
- Abstraction over route layer
- One source of truth for route and all params

What are the costs?

- Little shift in mindset needed
- Works best when Typed correctly

What else should we prepare?

- Route typings
- Parameter object typings

Where to start?

Routing & Types

```typescript
export enum RoutePath {
  // superviews
  Landing = 'landing',
  Platform = 'platform',

  // views
  Comparison = 'comparison',
  Settings = 'settings',
  UserProfile = 'profile',
  ItemDetail = 'detail',
}

export enum RouteParamKey {
  ComparisonIds = 'comparisonIds',
}

export type RouteParamMap = {
  [P in RouteParamKey]?: string;
}

export interface RouteModel {
  path: RoutePath[];
  params: RouteParamMap;
  queryParams: {[key: string]: string};
}
```

```typescript
const platformRoutes: Routes = [
  {path: RoutePath.Settings, component: OverviewSettingsComponent},
  {path: RoutePath.UserProfile, component: UserProfileComponent},
  {path: `${RoutePath.ItemDetail}/:id`, component: ItemDetailComponent},
  {path: RoutePath.Comparison, component: ComparisonComponent},
  {path: '**', redirectTo: RoutePath.Platform}
];

const appRoutes: Routes = [
  {path: RoutePath.Landing, component: LandingComponent},
  {
    path: RoutePath.Platform,
    component: PlatformComponent,
    children: platformRoutes
  },
  {path: '**', redirectTo: RoutePath.Landing}
];

@NgModule({
  imports: [RouterModule.forRoot(appRoutes)],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```

How to sync the route state?

Action

```
export class UpdateRouteAction implements Action {
  static readonly type: 'UpdateRoute' = 'UpdateRoute';
  readonly type = UpdateRouteAction.type;

  constructor(public readonly route: RouteModel) {}
}

export type RouteActions
  = UpdateRouteAction;
```

Effects

```typescript
@Injectable()
export class RouteEffects {

  @Effect() storeRoute$ = this.router.events.pipe(
    filter(event => event instanceof NavigationEnd),
    map((event) => {
      const urlTree = this.router.parseUrl(event.urlAfterRedirects);
      const {segments} = urlTree.root.children.primary;

      return new UpdateRouteAction({
        path: segments.map(segment => segment.path as RoutePath),
        params: segments.reduce((acc, segment) => ({ ...acc,  ...segment.parameters}), {}),
        queryParams: urlTree.queryParams,
      });
    }),
  );

  constructor(private readonly router: Router) {}
}
```

Reducer

```
export function routeReducer(
  state: RouteState = {path: [], params: {}, queryParams: {}},
  action: RouteActions,
): RouteState {
  switch (action.type) {
    case UpdateRouteAction.type:
      return action.route;

    default:
      return state;
  }
}
```

Selectors

```
export const $route = ({route}: AppStateModel) => route;

export const $routePath = createSelector(
  $route,
  (route: RouteState) => route.path
);

export const $routeParams = createSelector(
  $route,
  (route: RouteState) => route.params
);

export const $isLanding = createSelector(
  $routePath,
  (path: string[]) => path[0] === RoutePath.Landing
);

export const $comparisonIds = createSelector(
  $routeParams,
  (params: RouteParamMap) => params[RouteParamKey.ComparisonIds]
);
```

@ngrx/router-store

Demo time!

flow^up

github.com / vmasek / router-store-demo

Vojtech Mašek
Head dev @FlowUp

/ vmasek

/ VojtechMasek

/ vmasek