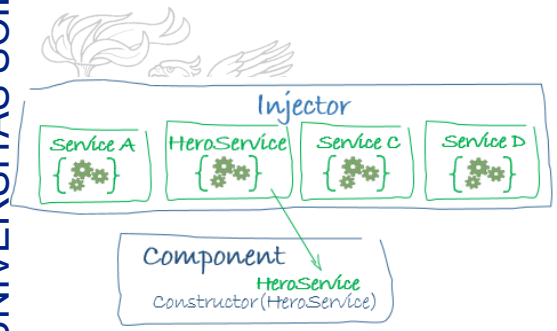
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

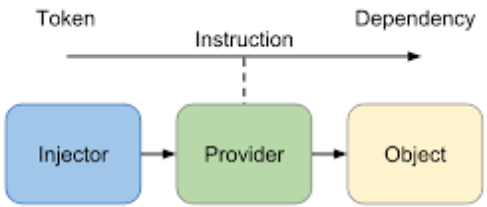
Angular Routing

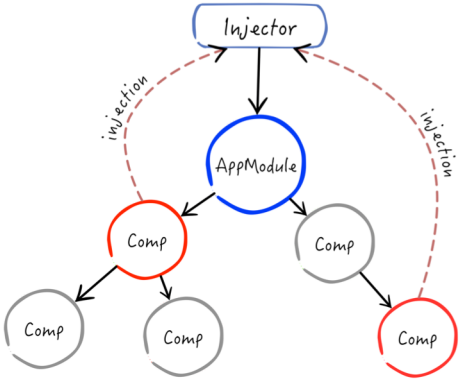
Dr. Bilicki Vilmos
Szoftverfejlesztés Tanszék

Az előző előadás összefoglalója

- DI
- Angular Dependency Injection
- Hierarchikus Injectorok
- DI Szolgáltatók
- DI működés közben
- Navigáció a komponens fán

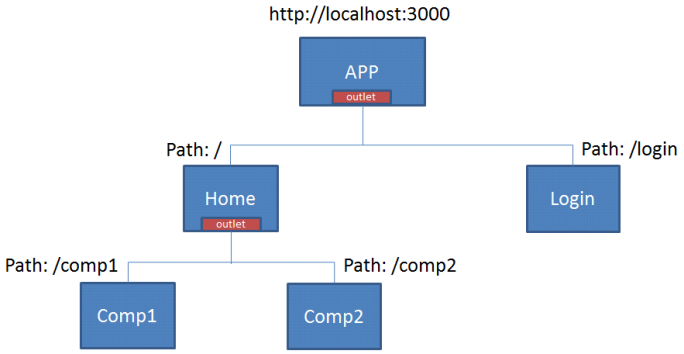


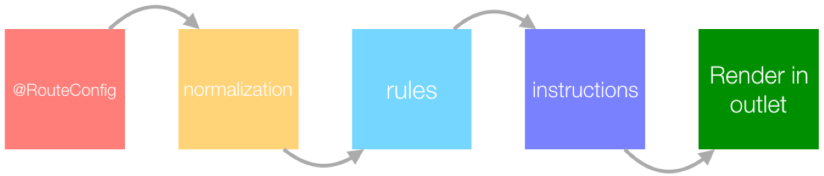




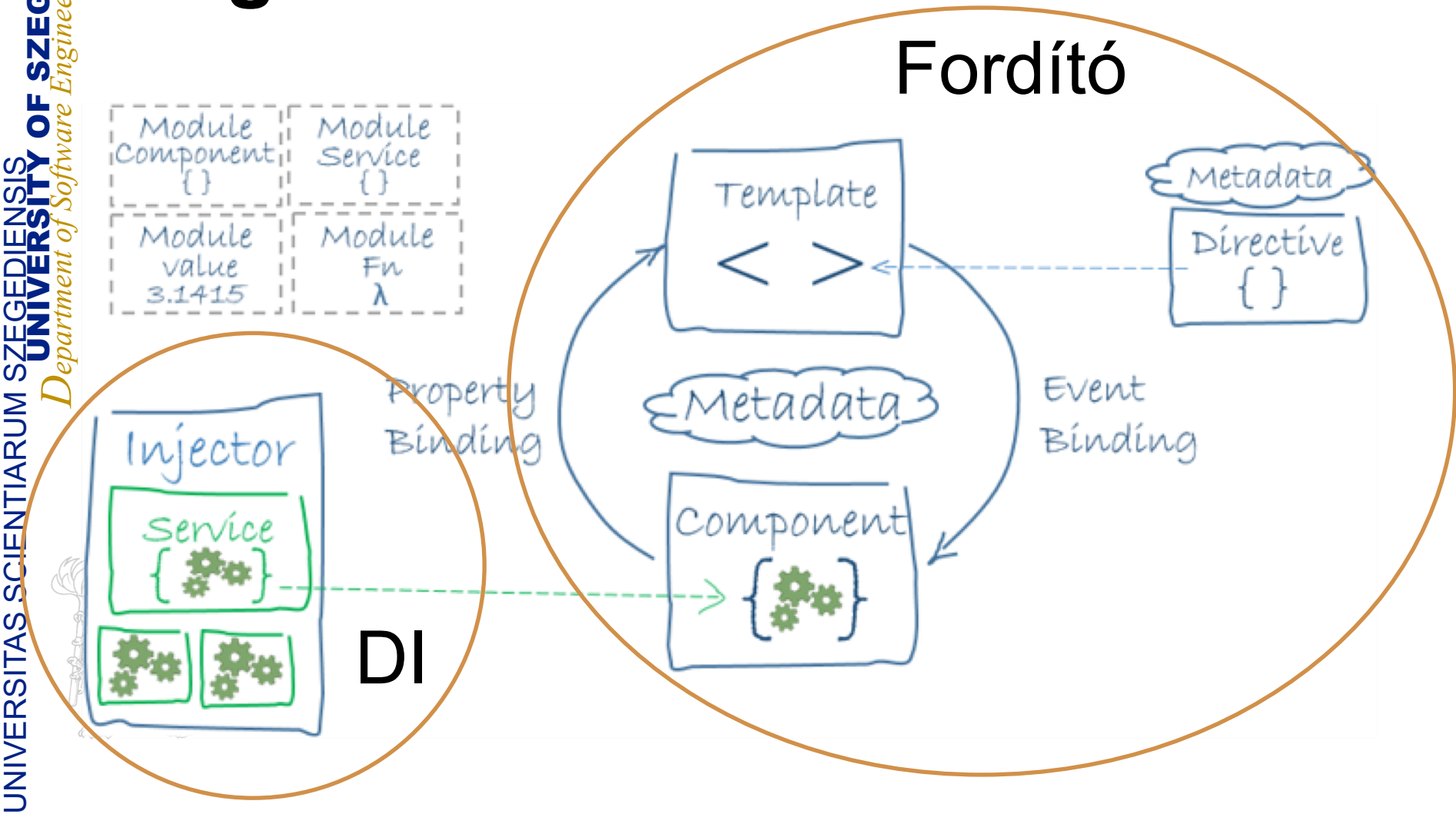
Áttekintés

- Angular Routing, Navigation
  - Router import
  - Konfiguráció
  - Router outlet
  - Router link
  - Aktív router link
  - Router állapot
  - Aktivált útvonal
  - Router események



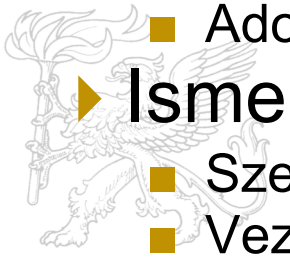


Angular architektúra



Forgalomirányítás - Routing

- Kliens oldalon vagyunk:
  - Egy URL is elegendő lenne
  - De
    - Nem tudunk úgy firssíteni, hogy megmaradjon az állapotunk
    - Nem tudjuk lemeneteni az elérési címet
    - Nem tudjuk megosztani az URL-t
- Az alkalmazást különböző területekre bontjuk
- Ezen területeket külön URL érjük el
- Segítségével:
  - Elkölönítjük az alkalmazás területeit
  - Az alkalmazás állapotát megfelelő eszköztárral kezeljük
  - Adott területeket adott szinten védhetünk
- Ismert minta alapján
  - Szerver (server)
  - Vezérlő (controller)
  - Akció (action)



Forgalomirányítás

- Első megoldások belső hivatkozások:

```
1 <!-- ... lots of page content here ... -->
2 <a name="about"><h1>About</h1></a>
```

- # alapú forgalomirányítás (Hash based routing)

```
http://something/#/about
```

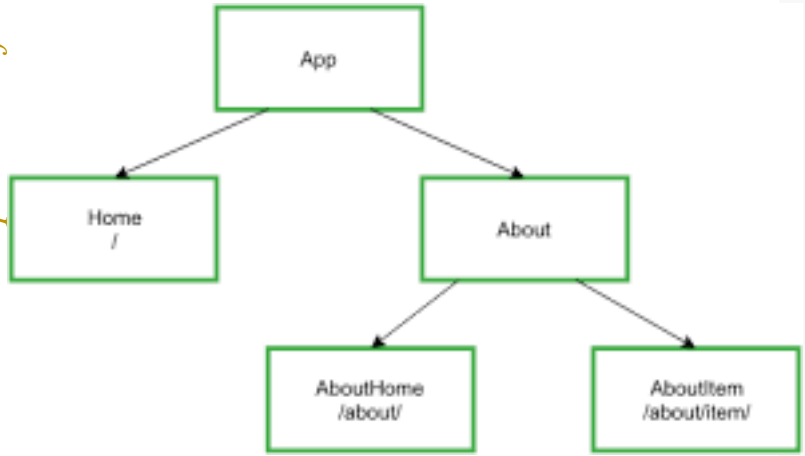
- HTML5 - a böngésző módosíthatja a böngészés múltat szerver oldali lekérdezés nélkül (ezt a böngészőnek és a szevernek is támogatnia kell)
  - pushState

base-href

- A forgalomirányítónak meg kell adni a kezdeti URL-t
  - index.html – fejléc részbe: <base href="/">
  - dinamikusan:
    - ng build --base-href /myUrl/

Angular router

- Nem része a Core csomagnak
  - import { RouterModule, Routes } from '@angular/router';
- Konfigurálása




```
const appRoutes: Routes = [
  { path: 'crisis-center', component: CrisisListComponent },
  { path: 'hero/:id',      component: HeroDetailComponent },
  {
    path: 'heroes',
    component: HeroListComponent,
    data: { title: 'Heroes List' }
  },
  { path: '',
    redirectTo: '/heroes',
    pathMatch: 'full'
  },
  { path: '**', component: PageNotFoundComponent }
];

@NgModule({
  imports: [
    RouterModule.forRoot(
      appRoutes,
      { enableTracing: true } // <-- debugging purposes only
    )
    // other imports here
  ],
  ...
})
export class AppModule { }
```

RouterOutlet

- Ide fogja a tartalmat behelyezni

```
<router-outlet></router-outlet>
<!-- Routed components go here -->
```

Router links

- <a> megfelelői
  - A routerLink elemek egyszeri kötés elvet követik (nem változnak meg)
- routerLinkActive
  - CSS
  - Jogosultságkezelés

```
<h1>Angular Router</h1>
<nav>
  <a routerLink="/crisis-center" routerLinkActive="active">Crisis Center</a>
  <a routerLink="/heroes" routerLinkActive="active">Heroes</a>
</nav>
<router-outlet></router-outlet>
```

Fontosabb fogalmak

Router elem	Jelentése
Router	Megjeleníti az aktív URL-hez tartozó alklamazás komponst. Menedzseli a komponensek közötti navigációt.
RouterModule	Egy külön NgModule amely a megfelelő szolgáltatók és direktívák segítségével navigációt biztosít az alkalmazás nézetek között.
Routes	Útvonalak tömbje, mindegyik egy-egy URL útvonalat köt egy komponenshez.
Route	Megadja, hogy a router hogyan navigáljn az adott komponenshez az adott URL segítségével. A legtöbb útvonal egy URL és egy komponens kötését adja meg.
RouterOutlet	Egy direktíva (<router-outlet>) amely megadja a view-ban a komponens helyét.
RouterLink	Egy direktíva amely kattintható HTML elemként segít navigálni.
RouterLinkActive	Segítségével lehet a HTML elemekből osztályokat elvenni/hozzáadni.
ActivatedRoute	A komponens megkapja kérés adatait.
RouterState	Az router állapota, beleértve az aktivált útvonalakat, ….
Link parameters array	Egy útvonalakat tartalmazó tömb.
Routing component	Egy angular komponens amelyet a RouterOutlet adott view-ban megjelenít a navigáció hatására.

Aktivált útvonal

Tulajdonság	Leírás
url	An Observable of the route path(s), represented as an array of strings for each part of the route path.
data	An Observable that contains the data object provided for the route. Also contains any resolved values from the resolve guard.
paramMap	An Observable that contains a map of the required and optional parameters specific to the route. The map supports retrieving single and multiple values from the same parameter.
queryParamMap	An Observable that contains a map of the query parameters available to all routes. The map supports retrieving single and multiple values from the query parameter.
fragment	An Observable of the URL fragment available to all routes.
outlet	The name of the RouterOutlet used to render the route. For an unnamed outlet, the outlet name is *primary*.
routeConfig	The route configuration used for the route that contains the origin path.
parent	The route's parent ActivatedRoute when this route is a child route.
firstChild	Contains the first ActivatedRoute in the list of this route's child routes.
children	Contains all the child routes activated under the current route.

Router események

Router Event	Description
NavigationStart	An event triggered when navigation starts.
RouteConfigLoadStart	An event triggered before the Router lazy loads a route configuration.
RouteConfigLoadEnd	An event triggered after a route has been lazy loaded.
RoutesRecognized	An event triggered when the Router parses the URL and the routes are recognized.
GuardsCheckStart	An event triggered when the Router begins the Guards phase of routing.
ChildActivationStart	An event triggered when the Router begins activating a route's children.
ActivationStart	An event triggered when the Router begins activating a route.
GuardsCheckEnd	An event triggered when the Router finishes the Guards phase of routing successfully.
ResolveStart	An event triggered when the Router begins the Resolve phase of routing.
ResolveEnd	An event triggered when the Router finishes the Resolve phase of routing successfuly.
ChildActivationEnd	An event triggered when the Router finishes activating a route's children.
ActivationEnd	An event triggered when the Router finishes activating a route.
NavigationEnd	An event triggered when navigation ends successfully.
NavigationCancel	An event triggered when navigation is canceled. This is due to a Route Guard returning false during navigation.
NavigationError	An event triggered when navigation fails due to an unexpected error.
Scroll	An event that represents a scrolling event.

Routing modul

- Nagyobb alkalmazásnál nem célszerű mindent az app.module.ts-be rakni
- Előnyök:
  - Elkülöníti az útvonalválasztást az egyébb feladatoktól.
  - Egy a tesztelés során könnyen eltávolítható modulba szerverzi az útvonalakat.
  - Jól ismert helyként összefogja az útvonalválasztás és a joogosultság területét.
  - Nem deklarál komponenseket.

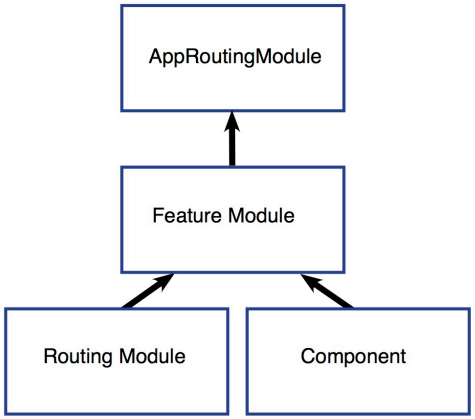
app-routing.module.ts

```typescript
import { NgModule }             from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { CrisisListComponent }   from './crisis-list/crisis-list.component';
import { HeroListComponent }     from './hero-list/hero-list.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

const appRoutes: Routes = [
  { path: 'crisis-center', component: CrisisListComponent },
  { path: 'heroes',        component: HeroListComponent },
  { path: '',   redirectTo: '/heroes', pathMatch: 'full' },
  { path: '**', component: PageNotFoundComponent }
];

@NgModule({
  imports: [
    RouterModule.forRoot(
      appRoutes,
      { enableTracing: true } // <-- debugging purposes only
    )
  ],
  exports: [
    RouterModule
  ]
})
export class AppRoutingModule {}
```

```typescript
import { NgModule }       from '@angular/core';
import { BrowserModule }  from '@angular/platform-browser';
import { FormsModule }    from '@angular/forms';

import { AppComponent }     from './app.component';
import { AppRoutingModule } from './app-routing.module';

import { CrisisListComponent }   from './crisis-list/crisis-list.component';
import { HeroListComponent }     from './hero-list/hero-list.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
  declarations: [
    AppComponent,
    HeroListComponent,
    CrisisListComponent,
    PageNotFoundComponent
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Képesség modulok

- Egy átlagos alkalmazás több képesség területre bontható, más-más funkcionalitással, céllal.
- Ezeket kezelhetjük együtt is az src/app/ mappában, de ez így nem less átlátható, karbantartható. Célszerűbb ezeket a területeket sajét mappákba szervezni.
  - A mappákon túl érdemes minden modulnak saját útvonal konfigurációs fájlt is létrehozni..
  - RouterModule.forRoot()
  - RouterModule.forChild()



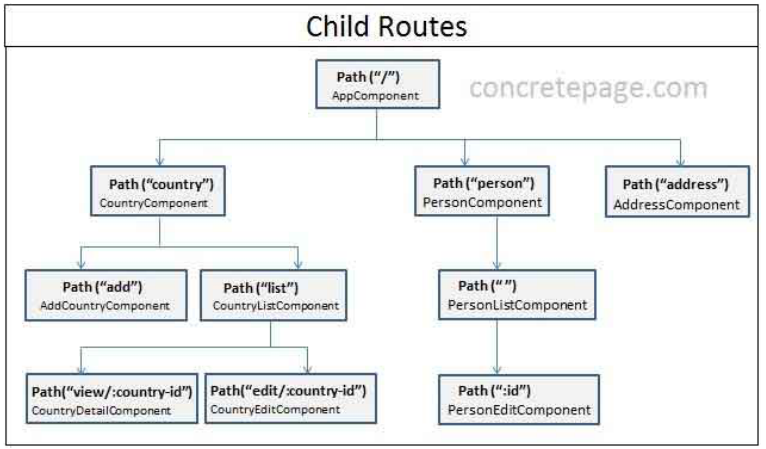
```typescript
import { NgModule }             from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { HeroListComponent }    from './hero-list/hero-list.component';
import { HeroDetailComponent }  from './hero-detail/hero-detail.component';

const heroesRoutes: Routes = [
  { path: 'heroes',  component: HeroListComponent },
  { path: 'hero/:id', component: HeroDetailComponent }
];

@NgModule({
  imports: [
    RouterModule.forChild(heroesRoutes)
  ],
  exports: [
    RouterModule
  ]
})
export class HeroesRoutingModule { }
```



```typescript
import { NgModule }       from '@angular/core';
import { BrowserModule }  from '@angular/platform-browser';
import { FormsModule }    from '@angular/forms';

import { AppComponent }     from './app.component';
import { AppRoutingModule } from './app-routing.module';
import { HeroesModule }     from './heroes/heroes.module';

import { CrisisListComponent }   from './crisis-list/crisis-list.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    HeroesModule,
    AppRoutingModule
  ],
  declarations: [
    AppComponent,
    CrisisListComponent,
    PageNotFoundComponent
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Fontos az importálási sorrend

- Amikor az összes útvonal az AppRoutingModule-ban található akkor az általános útvonalak a vége felé vannak.
- Mivel nem egy fájlban vannak az útvonalak ezért az összefűzésnél fontos ezen fájlok sorrendje.
- Adott modul az importálás sorrendjében egészíti ki a konfigurációs fájlt




```
imports: [
  BrowserModule,
  FormsModule,
  HeroesModule,
  AppRoutingModule
],
```

Útvonal paraméterek

```
{ path: 'hero/:id', component: HeroDetailComponent }
```

```
localhost:4200/hero/15
```

```
<a [routerLink]="['/hero', hero.id]">
```

Az útvonal elérése

```
import { ActivatedRoute } from "@angular/router";

constructor(private route: ActivatedRoute) { }
```

URL paraméterek elérése

- Snapshot
  - Egy alkalmas esemény (egyszeri)
  - Nem használható, ha a paraméter változik (és az oldal nem lesz megsemmisítve) animals/dog-tól animals/cat-hoz nem semmisíti meg az animals komponenst


```
ngOnInit() {
  this.animal = this.route.snapshot.paramMap.get("animal")
}
```

Lekérdezés paraméterek elérése

```
/users?id=123
```

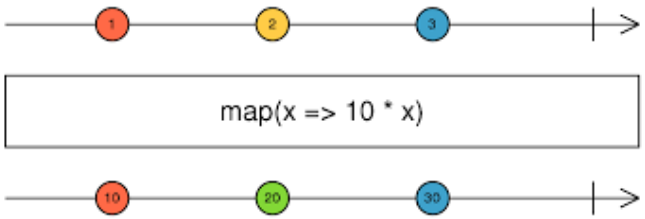
```
ngOnInit() {
  this.name = this.route.snapshot.queryParamMap.get("paramName")
  this.route.queryParamMap.subscribe(queryParams => {
    this.name = queryParams.get("paramName")
  })
}
```

Aktivált útvonal


```
import { switchMap } from 'rxjs/operators';

constructor(
  private route: ActivatedRoute,
  private router: Router,
  private service: HeroService
) {}

ngOnInit() {
  this.hero$ = this.route.paramMap.pipe(
    switchMap((params: ParamMap) =>
      this.service.getHero(params.get('id')))
  );
}
```

map, mergeMap, switchMap

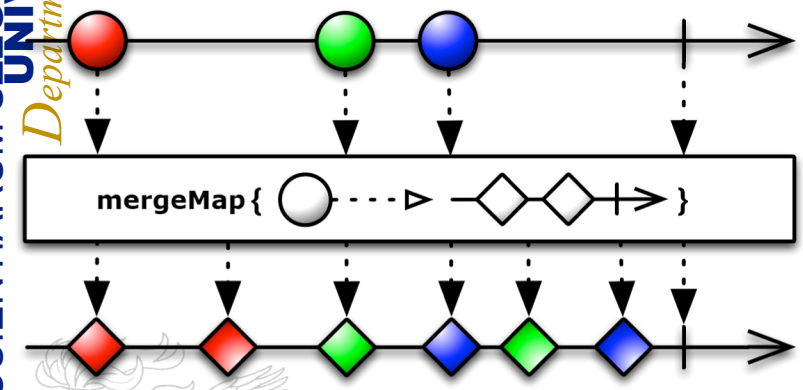
- map - Observable -» adat -» Observable




```
const data = of([
  {
    brand: 'porsche',
    model: '911'
  },
  {
    brand: 'porsche',
    model: 'macan'
  },
  {
    brand: 'ferarri',
    model: '458'
  },
  {
    brand: 'lamborghini',
    model: 'urus'
  }
]);
data
  .pipe(
    map(cars => cars.map(car => `${car.brand} ${car.model}`))
  ).subscribe(cars => console.log(cars))
data
  .pipe(
    map(cars => cars.filter(car => car.brand === 'porsche'))
  ).subscribe(cars => console.log(cars))
```

map, mergeMap, switchMap

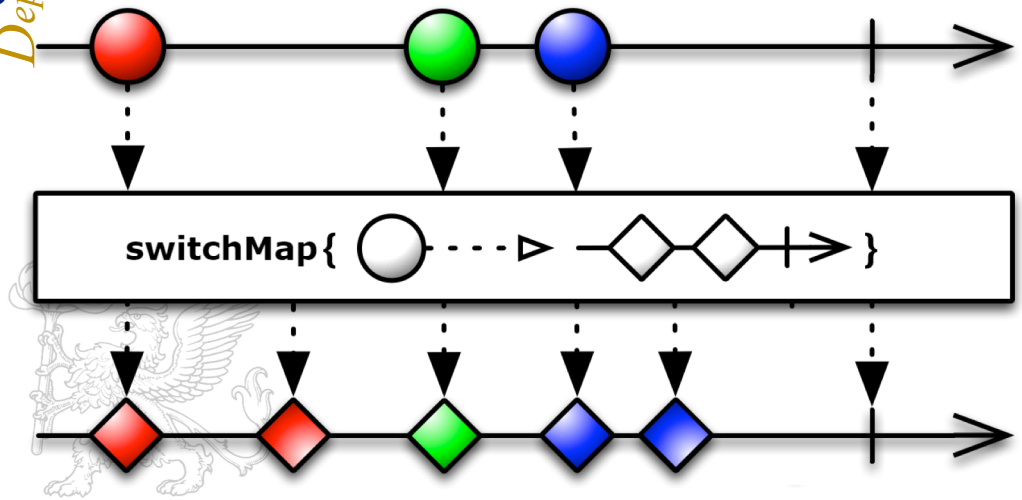
- mergeMap
  - Observable/Observable -» adat -» Observable




```
1   import { of, from } from 'rxjs';
2   import { map, mergeMap, delay, mergeAll } from 'rxjs/operators';
3
4   const getData = (param) => {
5     return of(`retrieved new data with param ${param}`).pipe(
6       delay(1000)
7     )
8   }
9
10  // using a regular map
11  from([1,2,3,4]).pipe(
12    map(param => getData(param))
13  ).subscribe(val => val.subscribe(data => console.log(data)));
14
15  // using map and mergeAll
16  from([1,2,3,4]).pipe(
17    map(param => getData(param)),
18    mergeAll()
19  ).subscribe(val => console.log(val));
20
21  // using mergeMap
22  from([1,2,3,4]).pipe(
23    mergeMap(param => getData(param))
24  ).subscribe(val => console.log(val));
25
```

map, mergeMap, switchMap

- switchMap
  - Obeservable/Observable -» adat -» Observable
  - a belsőből csak egyet, a többit lemondja




```
1  import { of, from } from 'rxjs';
2  import { map, delay, switchAll, switchMap } from 'rxjs/operators';
3
4  const getData = (param) => {
5    return of(`retrieved new data with param ${param}`).pipe(
6      delay(1000)
7    )
8  }
9
10 // using a regular map
11 from([1,2,3,4]).pipe(
12   map(param => getData(param))
13 ).subscribe(val => val.subscribe(data => console.log(data)));
14
15 // using map and switchAll
16 from([1,2,3,4]).pipe(
17   map(param => getData(param)),
18   switchAll()
19 ).subscribe(val => console.log(val));
20
21 // using switchMap
22 from([1,2,3,4]).pipe(
23   switchMap(param => getData(param))
24 ).subscribe(val => console.log(val));
25
```

Opcionális paraméterek

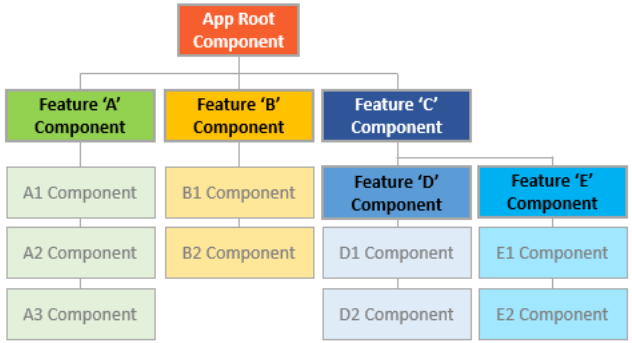
- Mátrix URL jelölés

```
this.router.navigate(['/heroes', { id: heroId, foo: 'foo' }]);
```

```
localhost:4200/heroes;id=15;foo=foo
```

```
ngOnInit() {
  this.heroes$ = this.route.paramMap.pipe(
    switchMap(params => {
      // (+) before `params.get()` turns the string into a number
      this.selectedId = +params.get('id');
      return this.service.getHeroes();
    })
  );
}
```

Gyermek útvonalak




```
const crisisCenterRoutes: Routes = [
  {
    path: 'crisis-center',
    component: CrisisCenterComponent,
    children: [
      {
        path: '',
        component: CrisisListComponent,
        children: [
          {
            path: ':id',
            component: CrisisDetailComponent
          },
          {
            path: '',
            component: CrisisCenterHomeComponent
          }
        ]
      }
    ]
  }
];
```

```
@NgModule({
  imports: [
    RouterModule.forChild(crisisCenterRoutes)
  ],
  exports: [
    RouterModule
  ]
})
export class CrisisCenterRoutingModule { }
```

Több útvonal megjelenítése

- Nevesített kivezetések
- Másodlagos útvonalak

```
<div [@routeAnimation]="getAnimationData(routerOutlet)">
  <router-outlet #routerOutlet="outlet"></router-outlet>
</div>
<router-outlet name="popup"></router-outlet>
```

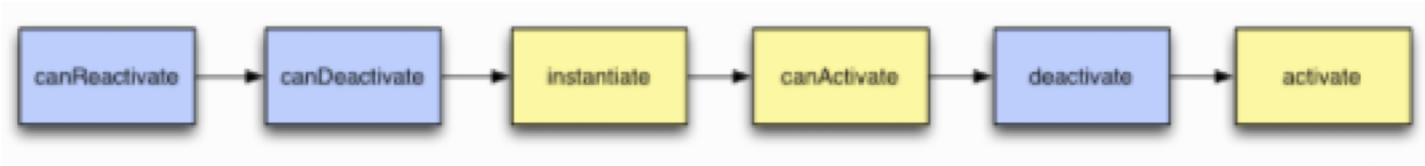


```
{
  path: 'compose',
  component: ComposeMessageComponent,
  outlet: 'popup'
},
```

```
<a [routerLink]="[{ outlets: { popup: ['compose'] } }]">Contact</a>
```

Forgalomirányítás

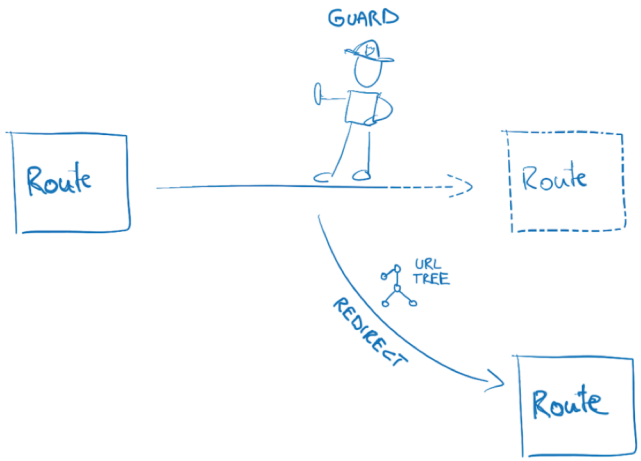
- Események/Életciklus



- Védett oldalak kezelése (UI élmény szempontjából, nem biztonság szempontjából)

Route guard

- A guard visszatérési értéke adja meg a hivatkozsá aktiválhatóságát:
  - Amennyiben igazzal tér vissza akkor a nevigációs folyamat folytatódhat.
  - Amikor hamissal akkor a folyamat leáll.
  - Amikor UrlTree-vel tér vissza akkor a mostani navigáció leáll és az UrlTree-vel megadott helyre navigál.



Guard interfészek

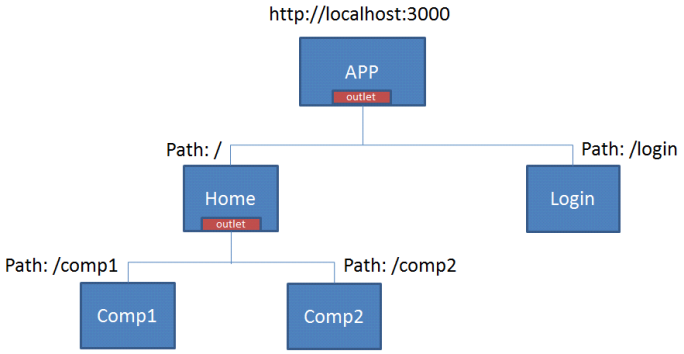
- CanActivate: továbbtja a navigációt a célhoz.
- CanActivateChild: továbbíthatja a navigációt egy gyerek útvonalhoz
- CanDeactivate: elnavigálhat az adott útvonaltól.
- Resolve: lekérheti az útvonal adatot a navgiáció előtt.
- CanLoad: aszinkron módon betöltött oldalra navigálhat.

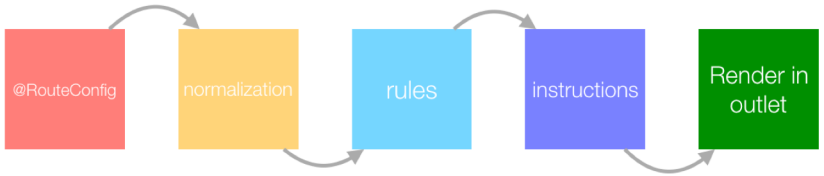
```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot }
from '@angular/router';

@Injectable({
  providedIn: 'root',
})
export class AuthGuard implements CanActivate {
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): boolean {
    console.log('AuthGuard#canActivate called');
    return true;
  }
}
```

```
const adminRoutes: Routes = [
  {
    path: 'admin',
    component: AdminComponent,
    canActivate: [AuthGuard],
    children: [
      {
        path: '',
        children: [
          { path: 'crises', component: ManageCrisesComponent },
          { path: 'heroes', component: ManageHeroesComponent },
          { path: '', component: AdminDashboardComponent }
        ],
      }
    ]
  }
];
```

Áttekintés

- Angular Routing, Navigation
  - Router import
  - Konfiguráció
  - Router outlet
  - Router link
  - Aktív router link
  - Router állapot
  - Aktivált útvonal
  - Router események





A következő előadás összefoglalója

- Angular HTTP
  - Szolgáltatás igénybevétele
  - JSON feldolgozása, ellenőrzése
  - Hiba kezelés
  - Hiba részletek
    - retry()
  - Observables és operátorok
  - Nem JSON adat kérése
  - POST, DELETE, PUT
  - A kérés konfigurálása - kérés, válasz elkapása
  - Előrehaldás események
  - XSRF
  - Saját süti, fejléc