JPA – Backend

Query, mit der man einen List-Parameter einer Entität mitjoinen kann

```
return em.createQuery("SELECT b from Bill b LEFT JOIN FETCH b.lines", Bill.class).getResultList();
```

- **Dao**
  - Named und Scope Annotations festlegen

```
@Named
@RequestScoped
public class TicketDao {
```

  - EntityManager als PersistenceContext festlegen

```
@PersistenceContext
EntityManager em;
```

  - GetAll

```
public List<Ticket> getAll() {
    return em.createNamedQuery("Ticket.findAll", Ticket.class).getResultList();
    //return em.createQuery("Select t from Ticket t LEFT JOIN FETCH t.assignments", Ticket.class).getResultList();
}
```

  - GetOne

```
public Ticket get(int id) {
    return em.find(Ticket.class, id);
}
```

  - Create → Transactional Annotation wichtig!

```
@Transactional
public void create(Ticket toInsert) {
    em.persist(toInsert);

    System.out.println(toInsert.getId());
}
```

  - Update → Transactional

```
@Transactional
public void update(Ticket toUpdate) {

    Ticket ticket = get(toUpdate.getId());
    System.out.println(toUpdate.getDescription() + " " + ticket.getDescription());

    if(ticket != null) {
        ticket.setDescription(toUpdate.getDescription());
        ticket.setSubmittedOn(toUpdate.getSubmittedOn());
        ticket.setPriority(toUpdate.getPriority());
        ticket.setState(toUpdate.getState());
        ticket.setUser(toUpdate.getUser());


        em.merge(ticket);
    }

}
```

    Objekt vorher vom EntityManager holen, um das ManagedObject zu erhalten und bei diesem dann die neuen Werte setzen – nicht das übergebene Mergen!

  - Delete → Transactional

```
@Transactional
public void delete(int id) {
    Ticket toRemove = get(id);
    if(toRemove != null) {
        em.remove(toRemove);
    }
}
```

- CORS
  - Filter am Server, der die Requests intercepted
    Unbedingt @Provider Annotation einfügen!

```
@Provider
public class CORSFilter implements ContainerResponseFilter {
    @Override
    public void filter(final ContainerRequestContext requestContext, final ContainerResponseContext cres)
            throws IOException {
        cres.getHeaders().add("Access-Control-Allow-Origin", "*");
        cres.getHeaders().add("Access-Control-Allow-Headers", "origin, content-type, accept, authorization");
        cres.getHeaders().add("Access-Control-Allow-Credentials", "true");
        cres.getHeaders().add("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE, OPTIONS, HEAD");
        cres.getHeaders().add("Access-Control-Max-Age", "1209600");
    }
}
```

Angular Client

- Routing
  - Routen definieren

```
const routes: Routes = [
  { path: "home", component: ListComponent },
  { path: "edit/:id", component: EditComponent },
  { path: "", redirectTo: "/home", pathMatch: "full" }
];
```

  - Parameter aus Route → Activated Route und paramMap.map → Observable!

```
constructor(private route: ActivatedRoute, private http: HttpClient
  private router: Router, private dataService: DataService) { }

ngOnInit() {
  this.editID = this.route.paramMap.map(param => param.get('id'));
```

  - Routerevent manuell auslösen

```
this.router.navigate(["home"]);
```

  - Routerevent über <a> auslösen

```
<a routerLink="/edit/{{ticket.id}}">Edit</a>
```

-

```
<tr *ngFor="let ticket of ticketList | async">
  <td>{{ticket.description}}</td>
```

- Dropdown

```
<select (change)="setState($event.target.value)" name="stateName">
  <option [selected]="toInsert.description.length == 0">Select State</option>
  <option *ngFor="let state of stateList | async">
    {{state.stateName}}
  </option>
</select>
```

  $event ist das ausgelöste Event
  So kann man mittels event.preventDefault() im Code z.B. einen Pagereload vermeiden