SEW Cheatsheet 5BHITM

Backend

Model

Entity

```
@Entity
// OR PanacheEntity (No definition of ID requiered)
public class Model1 extends PanacheEntityBase {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "model_seq")
    @SequenceGenerator(name = "model_seq", sequenceName = "model_seq", initialValue =
100) // OPTIONAL
    public Long id;

    @Enumerated(EnumType.STRING) //OPTIONAL
    public Enum enumValue;

    @ManyToOne
    @JoinColumn(name = "otherModel_id") //OPTIONAL
    public OtherModel otherModel;

    @OneToMany(mappedBy = "model", fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    public List<Model2> model2List;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.PERSIST)
    @JoinTable(name = "model1_model2", joinColumns = @JoinColumn(name = "model1_id"),
inverseJoinColumns = @JoinColumn(name = "model2_id")) //OPTIONAL
    public List<Model2> otherModel2List;

    // CONSTRUCTOR etc.
}

public class Model2 extends PanacheEntity {
    //does not need id because of PanacheEntity

    //bi directional relationships with Model1

    @ManyToOne
    public Model1 model;

    @ManyToMany(mappedBy = "otherModel2List")
    public List<Model1> model1List;
```

```
}
```

Inheritance

```
@Entity
@Inheritance(strategy = InheritanceType.JOINED) // or .SINGLE_TABLE or
.TABLE_PER_CLASS
public class ParentModel ...
```

DTO

```
public record ModelDTO(DataType param1, ...) { }
```

Repository

```
@ApplicationScoped
// or PanacheRepository<Model> but then findById etc. only works if the ID of the
model is a Long
public class ModelRepository implements PanacheRepositoryBase<Model, ID_DataType> {
    @Inject
    OtherRepo otherRepo;

    public void panacheFunction() {
        // Panache Functions:
        find("param", param);
        findById(id); // only working if ID is a Long or PanacheRepositoryBase is used
        list();
        listAll();
        listAll(Sort.by("name").ascending());
        delete("param", param);
        deleteById(id); // only working if ID is a Long or PanacheRepositoryBase is
used
        persist(model);
        ...

        // Sort Results
        listAll(Sort.by("attribute").ascending())
        listAll(Sort.by("lastName").and("firstName"));
        ...
    }

    public List<ModelDTO> dtoFunction() {
        return getEntityManager().createQuery("select new at.htl.dto.ModelDTO(params
...) from Entity e", ModelDTO.class)
                .getResultList();
```

```
        // OR

        return listAll().stream()
                .map(model -> model.getModelDTO()) // or create new DTO
                .toList();
    }

    public List<Model> entityManagerFunction(Long value) {
        return getEntityManager().createQuery("SELECT m FROM Model m WHERE m.param =
:param", Sport.class)
                .setParameter("param", "value")
                .getResultList();


    }

    public List<Model> empPerLocation() {
        TypedQuery<Model> query = getEntityManager().createQuery("select new
at.htl.model.dto.DTO_Name(params,COUNT(CASE WHEN p.role = true THEN 1 ELSE null END),
...) " +
                "from Table t " +
                "left join Table2 t2 on (t2.paramName = t1.paramName)" +
                "...", Model.class);
        return query.getResultList();
    }

    public Model listMax(){
        TypedQuery<Model> typedQuery = em.createQuery("select m from Model m" +
" left join Model2 m2 on m.id = m2=id" +
"where t.done = false", Model.class);
        typedQuery.setMaxResults(1);
        Model model = typedQuery.getSingleResult();
        return model;
    }

}
```

Resource

```
@Path("/path")
public class CourseResource {

    @Inject
    RepoClass Repo

    @GET // or POST, PUT, DELETE
    @Path("/{param}")
    public DataType function(@PathParam("param") DataType param) {
        ...
```

```
    }

    @POST
    @Path("/")
    @Transactional
    public Response postFunction(DataType bodyData) {
        ...
        return Response.status(statusCode).entity(data).build();
        return Response.ok().entity(data).build();

        // Return a Created Respone with a URI to the object in the location header
        return Response.created(new URI("link/to/object").build();
    }

    @PUT
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_JSON})
    @Transactional
    public Response update(@PathParam("id") Long id, Course course) {
        Course existing = courseRepo.findById(id);
        if (existing == null) {
            return Response.status(Response.Status.NOT_FOUND).build();
        }
        existing.title = course.title;
        ...
        return Response.ok(existing).build();
    }
}
```

Websockets

```
@ServerEndpoint("/socket")
@ApplicationScoped
public class CalenderSocket {
    @Inject
    ObjectMapper objectMapper;

    Set<Session> sessions = new HashSet<>();

    @OnOpen
    public void onOpen(Session session) {
        sessions.add(session);
        ...
    }

    // Same for OnClose as OnOpen but remove session

    @OnError
    public void onError(Session session, Throwable throwable) {
        ...
```

```
    }

    @OnMessage
    public void onMessage(String message, @PathParam("name") String name) {
        ...
    }

    public void broadcast(DataType data) {
        try{
            String data = objectMapper.writeValueAsString(data);
            sessions.forEach(s -> {
                s.getAsyncRemote().sendObject(data, result -> {
                    // check for Exeption -> result.getExeption
                    ...
                });
            });
        }catch (Exception e){
            ...
        }
    }
}
```

Frontend

Angular Commands

```
// Start Project
ng serve

// Generate Component
ng g c component-name

// Generate Service
ng g s service-name

// Generate Interface
ng g i interface-name
```

Angular Material

```
ng add @angular/material

// Example for Angular Material Component
ng g @angular/material:navigation menu
```

General

Binding

```
// One-Way Binding
{{ value }}

// Two-Way Binding
[(ngModel)]="value"

// Event Binding
(click)="function()"

// Property Binding
[disabled]="isDisabled"
```

onInit

```
class Component implements OnInit {
    ngOnInit() {
        // Code
    }
}
```

ngFor

```
<div *ngFor="let value of values">
...
</div>
```

ngIf

```
<div *ngIf="...">
Please select a school class.
</div>
```

ngStyle

```
[ngStyle]="{'background-color': isBlue() ? 'blue' : 'green'}"
```

ngClass

```
[ngClass]="{cssClass: someFunction()}"
```

Model

```
export interface Model {
    id: number;
    name: string;
    ...
}
```

Import / Output

parent.component.html

```
<app-children-component [inputName]="data" (outputName)="onEmit($event)"> // $event
sends parameters to parent function
```

children.component.ts

```
// name is optional and <Model> after EventEmitter is optional
@Input("name") parameter: Model = {} as Model;
@Output("name") parameterOutput = new EventEmitter<Model>();
```

Routing

app.routes.ts

```
{path: 'route/:param', component: RouteComponent} // Without / in front of route
{path: '**', component: NotFoundComponent} // Wildcard route
```

navigation

```
// RouterLink with routerLinkActive -> IMPORT in .component.ts
<div routerLink="/" class="link" routerLinkActive="link-active"
[routerLinkActiveOptions]="{exact: true}">
      HOME
</div>

// RouterOutlet -> IMPORT in .component.html
<router-outlet></router-outlet>
```

params

component.ts

```
router = inject(Router);
```

```
route = inject(ActivatedRoute);

// Subscribe to param changes -> in the ngOnInit function
this.route.params.subscribe(params => {
      this.value = params['param'];
})

// Get current param
this.route.snapshot.paramMap.get('param')

//navigate to another route
this.router.navigate(['/route', ...])
```

HttpClient

app.config.ts

```
provideHttpClient() // add to providers
```

http.service.ts

```
constructor(private http: HttpClient) { }

getData() {
    return this.http.get<Model>(API_URL)
}

postData(data) {
    return this.http.post(API_URL, data);
}
```

component.ts

```
httpService = inject(HttpService);

// In load funtion
this.httpService.getData().subscribe((value) => {
    this.data = value;
});
```

Forms

component.html

```
<form [formGroup]="studentForm" (ngSubmit)="onSubmit()">
      <div>
```

```
        <label for="name">Name:</label>
        <input id="name" formControlName="name" type="text">
        <div *ngIf="studentForm.get('name')?.invalid &&
studentForm.get('name')?.touched">
          Value is invalid.
        </div>
      </div>

      <div>
        <label for="dateValue">DatePicker:</label>
        <input id="dateValue" formControlName="dateValue" type="date">
      </div>

      <div>
        <label for="selectValue">Select:</label>
        <select id="selectValue" formControlName="selectValue">
          <option value="" disabled>Option 1</option>
            // Multiple Options -> *ngFor
        </select>
      </div>

      <button type="submit" [disabled]="studentForm.invalid">Submit</button>
</form>
```

component.ts

```
studentForm: FormGroup;

  constructor(private fb: FormBuilder) {
    this.studentForm = this.fb.group({
      name: ['', [Validators.required, Validators.minLength(2)]],
      dateValue: ['', Validators.required],
      selectValue: ['', Validators.required]
    });
  }

// Get Values
this.studentForm.value.name

// Reset Form
this.studentForm.reset();
```

Websockets

component.ts

```
ngOnInit() {
    const socket = new WebSocket("ws://localhost:8080/socket");
    socket.onmessage = (event: MessageEvent) => {
```

```
      const data = JSON.parse(event.data);
      console.log("Socket message: " + data);
      this.courses = data;
    };
    // socket.onopen or .onmessage ...
  }
```

Frontend Functionality

Arrays

component.ts

```
array.filter(value => {
        // check and return true/false
    })

array.sort((a, b) =>
        // compare a to b
        // Sort String: a.localeCompare(b)
        // Sort Number asc: a - b
        // Sort Number desc: b - a
    )

array.splice(2, 1); // Removes 1 item at index 2
```

Disabled Button

component.html

```
<button [disabled]="!isValid">Speichern</button>
```

component.ts

```
isValid: boolean = false;

checkIfValid() {
    // check and set true/false
    this.isValid = true;
}
```

Select

component.html

```
<select [(ngModel)]="selectValue" (change)="changeFunction($event)">
  <option value="" disabled>Default Option</option>
  <option *ngFor="let option of options" value="{{option}}">{{option}}</option>
</select>
```

component.ts

```
selectValue: string = "";
options: DataType[] = [];

// load options in ngOnInit

changeFunction(e: any) {
    this.selectValue = e.target.value;
    // call update/reload Function if needed
}
```

Radio Buttons

component.html

```
<div *ngFor="let option of options">
  <input
    type="radio"
    name="radioGroupName"
    [value]="option.id"
    [(ngModel)]="selectedOptionId"
    (change)="changeOption(option.id // or $event)"
    >
  {{option.name}}
</div>
```

component.ts

```
selectedOptionId: string = "";
options: DataType[] = [];

// load options in ngOnInit

changeFunction(e: any) {
    this.selectedOptionId = e; // or e.target.value if $event is used
    // call update/reload Function if needed
}
```