

Goldschmiede



29.04.2022, Hans Jörg Heßmann

Vorstellung

Hans Jörg Heßmann

- Softwareentwickler
- Java seit 1997
- Schwerpunkte
 - Webanwendungen
 - Batch-Entwicklung
 - Build und Deployment
 - Codequalität und Testing
- Java, Spring, Vaadin, AsciiDoc, Dessert, ...


Unternehmen

Individuelle Anwendungsentwicklung - Java Enterprise, Web, Mobile

- seit 2005 ♦ in Köln ♦ für alle Branchen ♦ **Goldschmiede**  anderScore
- nach Aufwand & zum Festpreis


Unternehmen

Individuelle Anwendungsentwicklung - Java Enterprise, Web, Mobile

- seit 2005 ♦ in Köln ♦ für alle Branchen ♦ **Goldschmiede**  anderScore
- nach Aufwand & zum Festpreis
- ✓ Digitalisierung / Prozesse / Integration
- ✓ Migration
- ✓ Neuentwicklung
- ✓ Notfall / kritische Situationen
- pragmatisch, zielgerichtet, zuverlässig

Unternehmen

Individuelle Anwendungsentwicklung - Java Enterprise, Web, Mobile

- seit 2005 ♦ in Köln ♦ für alle Branchen ♦ **Goldschmiede**  anderScore
- nach Aufwand & zum Festpreis
- ✓ Digitalisierung / Prozesse / Integration
- ✓ Migration
- ✓ Neuentwicklung
- ✓ Notfall / kritische Situationen
- pragmatisch, zielgerichtet, zuverlässig

Einführung neues
Versicherungsprodukt

µservices
Automotive

Onlinebanking
Naturschutz: Kartie-
rung bedrohter Arten

Migration
Energieversorger

Stabilisierung
Retail

Schnittstellen
Gesundheitswesen

Adressverwaltung
Logistik

Nutzerservices
Energieversorger


Börsenhandel
Wertpapiere

Security Härtung

EAM & Refactoring
Leasing

Unternehmen

Individuelle Anwendungsentwicklung - Java Enterprise, Web, Mobile

- seit 2005 ♦ in Köln ♦ für alle Branchen ♦ **Goldschmiede**  anderScore
- nach Aufwand & zum Festpreis
- ✓ Digitalisierung / Prozesse / Integration
- ✓ Migration
- ✓ Neuentwicklung
- ✓ Notfall / kritische Situationen
- pragmatisch, zielgerichtet, zuverlässig

Kompletter SW Life Cycle

- Projektmanagement / agile Methodik
- Anforderungsanalyse
- Architektur & SW-Design
- Implementierung & Testautomation
- Studien & Seminare



... und für Sie? Sprechen Sie uns an!

Name: Hilla

Ungefähr 3.860.000.000 Ergebnisse (0,40 Sekunden)

Finnisch



Deutsch



Hilla



Moltebeere



[In Google Übersetzer öffnen](#) • [Feedback geben](#)

Moltebeere



Die **Moltebeere** (*Rubus chamaemorus*), auch Multebeere, Multbeere, Schellbeere, Sumpfbrombeere, Torfbeere oder Nordic Berry genannt, ist eine [Pflanzen-Art](#) aus der Gattung [Rubus](#). Sie ist einziger Vertreter der Untergattung *Chamaemorus* und gehört zur Unterfamilie der [Rosoideae](#) innerhalb der [Familie](#) der [Rosengewächse](#) (Rosaceae). Die Moltebeere kommt im Norden Amerikas, Europas und Asiens vor. Sie ist in Mitteleuropa sehr selten. Es gibt keine Vorkommen in [Österreich](#) und der [Schweiz](#). Die sehr geringen Vorkommen in Norddeutschland sind streng geschützt. Die Bekanntheit wuchs mit der Abbildung der Moltebeere auf der [finnischen 2 Euro](#)

Moltebeere



Vaadin

Ungefähr 3.860.000.000 Ergebnisse (0,40 Sekunden)

Finnisch



Deutsch



Vaadin



Weibliches Rentier



[In Google Übersetzer öffnen](#) • [Feedback geben](#)

hilla.dev



[Docs](#) [Support](#) [Blog](#) [GitHub](#)

The modern web framework for Java

Hilla integrates a Spring Boot **Java back end** with a reactive **TypeScript front end**. It helps you build apps faster with type-safe server communication, included UI components, and integrated tooling.

[Get started](#)

[Documentation](#)

TypeScript UI

Java back end

```
export class PersonView extends LitElement {
  @state() people: Person[] = [];

  async firstUpdated() {
    this.people = await PersonEndpoint.findAll();
  }

  render() {
    return html`
      <vaadin-grid .items=${this.people}>
        <vaadin-grid-column path="firstName"></vaadin-grid-column>
        <vaadin-grid-column path="lastName"></vaadin-grid-column>
      </vaadin-grid>`;
  }
}
```

<https://hilla.dev/>

Vergleich

https://vaadin.com/comparison?compare=hilla_vs_angular

Getting Started

Projekt anlegen:

```
npx @vaadin/cli init --hilla hilla-todo
```

→ <https://hilla.dev/docs/getting-started>

Build

Bauen und starten mit:

```
mvn clean install  
mvn spring-boot:run
```

Backend-Dependencies hinzufügen

<https://start.spring.io/>

- Lombok
- Spring Data JPA
- HSQL
- Actuator

New View anlegen

frontend/views/todo/todo-view.ts

```
@customElement('todo-view')
export class TodoView extends View {
  name = '';

  connectedCallback() {
    super.connectedCallback();
    this.classList.add('flex', 'p-m', 'gap-m', 'items-end');
  }

  render() {
    return html`
      <h1>Todo-Liste</h1>
    `;
  }
}
```

View hinzufügen

frontend/routes.ts

```
import './views/todo/todo-view';
...
{
  path: 'todo',
  component: 'todo-view',
  icon: 'la la-tasks',
  title: 'Todo',
},
...
```

Backend: Entity

```
@Entity
@Table(name = "TODO")
@Getter
@Setter
@ToString
public class TodoItem {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @NotEmpty
    @Size(min = 2, max = 64)
    private String name;

    @FutureOrPresent
    private LocalDate due;

    private boolean done;
}
```

Backend: Repository

```
public interface TodoItemRepository extends JpaRepository<TodoItem, Long> {  
}
```

Backend: Endpoint

```
@Endpoint
@AnonymousAllowed
@Slf4j
public class TodoEndpoint {

    @Autowired
    private TodoItemRepository repository;

    @Nonnull
    public List<@Nonnull TodoItem> findAll() {
        return repository.findAll();
    }

    public TodoItem saveItem(TodoItem item) {
        log.info("saveItem: {}", item);
        return repository.save(item);
    }

    public void deleteItem(TodoItem item) {
        log.info("deleteItem: {}", item);
        repository.delete(item);
    }
}
```

Einfaches Formular

```
import '@vaadin/button';
import '@vaadin/text-field';

...

render() {
  return html`
    <h1>Todo-Liste</h1>
    <div>
      <vaadin-text-field label="Aufgabe"></vaadin-text-field>
      <vaadin-button>Hinzufügen</vaadin-button>
    </div>
  `;
}
```


Layout-Anpassungen

```
connectedCallback() {  
  super.connectedCallback();  
  this.classList.add('flex', 'flex-col', 'p-m', 'gap-m', 'items-start');  
}
```

→ <https://vaadin.com/docs/latest/ds/foundation/utility-classes>

Binding

```
import {Binder, field} from "@hilla/form";
import TodoItemModel from "Frontend/generated/com/example/application/repository/todo/TodoItemModel";

...

binder = new Binder(this, TodoItemModel);

render() {
  return html`
    <h1>Todo-Liste</h1>
    <div>
      <vaadin-text-field label="Aufgabe" ${field(this.binder.model.name)}></vaadin-text-field>
      <vaadin-button>Hinzufügen</vaadin-button>
    </div>
  `;
}
```

Click-Handler

```
import {TodoEndpoint} from "Frontend/generated/endpoints";

...

render() {
  return html`
    <h1>Todo-Liste</h1>
    <div>
      <vaadin-text-field label="Aufgabe" ${field(this.binder.model.name)}></vaadin-text-field>
      <vaadin-button @click="${this.addItem}">Hinzufügen</vaadin-button>
    </div>
  `;
}

async addItem() {
  const savedItem = await this.binder.submitTo(TodoEndpoint.saveItem);
  if (savedItem) {
    console.log("created item with id " + savedItem.id);
    this.binder.clear();
  }
}
```

Liste anzeigen

```
@state()
private items: TodoItem[] = [];

async firstUpdated() {
  this.items = await TodoEndpoint.findAll();
}

render() {
  return html`
    <h1>Todo-Liste</h1>
    <div>
      ${this.items.map(item => html`
        <div class="todo-item">${item.name}</div>
      `)}
    </div>
    <div>
      <vaadin-text-field label="Aufgabe" ${field(this.binder.model.name)}></vaadin-text-field>
      <vaadin-button @click="${this.addItem}">Hinzufügen</vaadin-button>
    </div>
  `;
}

...

this.items = [...this.items, savedItem];
```

Styling

frontend/themes/hilla-todo/styles.css

```
todo-view .todo-item {  
  min-width: 20em;  
  background-color: lightgreen;  
}
```

Grid

```
import '@vaadin/date-picker';
import '@vaadin/checkbox';
import '@vaadin/grid';

...

render() {
  return html`
    <h1>Todo-Liste</h1>
    <vaadin-grid .items="${this.items}">
      <vaadin-grid-column path="id"></vaadin-grid-column>
      <vaadin-grid-column path="name" header="Aufgabe"></vaadin-grid-column>
      <vaadin-grid-column path="due"></vaadin-grid-column>
      <vaadin-grid-column path="done"></vaadin-grid-column>
    </vaadin-grid>
    <div>
      <vaadin-text-field label="Aufgabe" ${field(this.binder.model.name)}></vaadin-text-field>
      <vaadin-date-picker label="bis" ${field(this.binder.model.due)}></vaadin-date-picker>
      <vaadin-checkbox label="erledigt" ${field(this.binder.model.done)}></vaadin-checkbox>
      <vaadin-button @click="${this.addItem}">Hinzufügen</vaadin-button>
    </div>
  `;
}
```

MobX

The view is a function of the state (reactive programming model):

```
renderedHtml = template(state)
```

Store: Initialisierung

```
import TodoItem from "Frontend/generated/com/example/application/repository/todo/TodoItem";
import {makeAutoObservable, observable, runInAction} from "mobx";
import {TodoEndpoint} from "Frontend/generated/endpoints";

class TodoItemStore {
  todoItems: TodoItem[] = [];

  constructor() {
    makeAutoObservable(
      this,
      {
        initFromServer: false,
        todoItems: observable.shallow
      },
      {
        autoBind: true
      }
    );

    this.initFromServer();
  }

  async initFromServer() {
    const items = await TodoEndpoint.findAll();

    runInAction(() => {
      this.todoItems = items;
    })
  }
}

export const todoItemStore = new TodoItemStore();
```


Store: Operationen

```
async saveItem(item: TodoItem) {  
  const saved = await TodoEndpoint.saveItem(item);  
  if (saved) {  
    this.saveLocal(saved);  
  }  
}  
  
private saveLocal(saved: TodoItem) {  
  const itemExists = this.todoItems.some((item) => item.id === saved.id);  
  if (itemExists) {  
    this.todoItems = this.todoItems.map(item => item.id === saved.id ? saved : item);  
  } else {  
    this.todoItems = [...this.todoItems, saved];  
  }  
}
```

Store: Verwendung

```
<vaadin-grid .items="${todoItemStore.todoItems}">

...

async addItem() {
  await this.binder.submitTo(todoItemStore.saveItem);
  this.binder.clear();
}
```

Vertiefung

- Ausführliches Tutorial:
<https://hilla.dev/docs/tutorials/in-depth-course>
- Sourcen aus dieser Goldschmiede:
<https://github.com/anderscore-hjhessmann/hilla-goldschmiede>