010: Sequelize / Datenbank.

```
// install module
Const Sequelize = require('sequelize');
// Create object / build connection to DB.
Const Sequelize = new Sequelize('mysql' : // root : localhost : 3306 /consultingdb');
```

dialect — host — name of DB

```
// Create model / table
class Shop extends Sequelize.model{}
// init table, define type of columns
Shop.init({ shopName: Sequelize.STRING ... }, { sequelize, modelName: 'shop' });
```

Col. Name — TYPE: .STRING, .BOOLEAN, .INTEGER, .DATE — model name

```
// define association between tables
1) one to one:
    Mitarbeiter.hasOne(Abteilung);
2) one to many
    Abteilung.hasMany(Mitarbeiter);
    Mitarbeiter.belongsTo(Abteilung);
3) many to many
    class MitarbeiterProjRelationship extends Sequelize.Model{}
    MitarbeiterProjRelationship.init({ // only own attr. no for. key }, { seq-, m... });
    Mitarbeiter.belongsToMany(Projekt, { through: MitarbeiterProjRelationship });
    Projekt.belongsToMany(Mitarbeiter, { through: MitarbeiterProjRelationship });
// .sync: send info. to DB
sequelize.sync({ force:true }).then(() => {
    // define function to create init DB data
    let dataTransfer = async() => {
        // row without foreign key
        const firmA = await Firma.create({
            id: 3,
            titel: "..." });
        // row with foreign key
        const michaela = await Mitarbeiter.create({
            vorname: '...',
            .... }).then(mitarbeiter => { mitarbeiter.setFirm(firmA); });
        // row from relationship table
        const relationship01 = await MitProjRelationship.create({
            // foreign-keys
            mitarbeiterId: 2,
            ProjektId: 22,
            rolle: 'PL' });
        // close DB connection
        sequelize.close();
    }; dataTransfer();
});
```

09.01 koa

```
const Koa = require('koa');
const Router = require('koa-router');
const bodyparser = require('koa-bodyparser');
Const fs = require('fs');
// create Koa object
Const app = new Koa();
// create Router object
var router = new Router({ prefix: '/music' });
// install pug module
const Pug = require('koa-pug');
// path
const path = require('path');
// create pug object
const pug = new Pug({
    viewPath: path.resolve(__dirname, 'views');
    app: app });
router
    // reply get request with response
    .get('/', ctx => {
        ctx.response.body = fs.readFileSync('mLib.json', 'utF-8');
        ctx.response.status = 200; })
    // use Pug template and send data to pug.
    .get('/', ctx => {
        // read file
        const musicData = JSON.parse(fs.readFileSync('mLib.json', 'UTF-8'));
        // render to pug.
        await ctx.render = ('index', { itemList: musicData.songs });
    // modify file by dealing with put request
    .put('/', ctx => {
        // error-handling
        if (typeof ctx.request.body.titel == 'undefined'){
            ctx.throw(500, 'keine Information erhalten!'); }
        // read file
        let musicData = JSON.parse(fs.readFileSync('mLib.json', 'UTF-8'));
        // modify
        musicData.songs.push(ctx.request.body);
        // write back
        fs.writeBackSync('mLib.json', JSON.stringify(musicData)); });
// error-handling
app.on('error', err => { console.log('Error: ', err); });
// register middlewares
app.use(router.allowMethods())
    .use(bodyparser())
    .use(router.routes());
```

filename — format — name of pug file — Status code — Information.

09.02 Pug

```
// file: head.pug
head
    title Meine Webseite
    meta(charset='UTF-8')
    style
        .filmBorder{
            border: 2px solid yellow;
            background-color: pink;
        }
        .filmFont{
            text-align: left;
            font-family: ...;
            color: white;
        }
```

class name

```
// file: index.pug
doctype html
html
    include head.pug
    b
        h1 Mein Titel
        p Text
        include itemList.pug
// file: itemList.pug
mixin Film(titel, datum, regie)
    div.filmBorder
        div.filmFont = titel
        div.filmFont = datum
        div.filmFont = regie
div
    each item in itemList
        if item.datum
            // use mixin
            + Film(item.titel, item.datum, item.regie)
```

010 express.js

```
const Express = require('express');
const fs = require('fs');
const cors = require('cors');
// create express object
var app = new Express();
// use middleware
app.use(express.Json(), cors());
// server side methods
// get: read from server
app.get('/myMovies', (req, res) => {
    // read file from server fs
    var filmData = fs.readFileSync
        ('./spielFilm.Json', 'UTF-8');
    // response
    res.Json(filmData);
    res.Status(200); });
```

send data by response

```
// put: modify server file
app.put('/myMovies', (req, res) => {
    // read request
    var newFilm = req.body;
    // read file, store in JS format
    var filmData = JSON.parse(
        fs.readFileSync('./spielFilm.Json', 'UTF-8'));
    // modify
    filmData.filme.push(newFilm);
    // write back
    fs.writeFileSync('./spielFilm.Json', filmData);
    // optional: write info in console
    console.log("Film added!");
    res.Status(201);
});
```

```
// delete-
app.delete('./myMovies', (req, res) => {
    // read request
    let reqData = req.body.titel;
    // read file in server
    let filmData = JSON.parse(fs.readFileSync('./file.Json', 'UTF-8'));
    // find reqData in file
    for (index in filmData.filme){
        if(reqData == filmData.filme[index].titel){
            // delete reqData from list
            filmData.filme.splice(reqData);
            // write back
            fs.writeFileSync('./file.Json', JSON.stringify(filmData));
            break; }}
```

//delete req. is the same as put

```
// start server at port 8080 (HTTP)
app.listen(8080, () => {
    Console.log('Server running at 8080'); });
```

```
// client-side
// connect with Dom-element
var ShowAll btn = document.getElementById('allFilms');
var contentDiv = document.getElementById('...');
// add event on btn
ShowAll btn.addEventListener('click', () => {
    // create get request
    fetch('http://localhost:8080/myMovies', {
        // set get method
        method: 'get',
        headers: {
            'Accept': 'application/Json',
            'Content-Type': 'application/JSON' }
        }).then(function(response){
            response.Json().then(function(data){
            // extract content from data
            let filmListe = JSON.parse(data);
            // create HTML content
            let table = `<table> ... </table>`;
            contentDiv.innerHTML = table; }); }); });
```

from response

```
// put request from client, modify server file
// create input DOM-Element
var titelInput = document.getElementById('titelInput');
var addBtn = ...;
addBtn.addEventListener('click', () => {
    fetch('http://localhost:8080/myMovies', {
        method: 'put',
        headers: { ... }, body: JSON.stringify({
            titel: titelInput.value }) }});
```

send request. body to server

07 Vue.JS / components

```
<!DOCTYPE html>
<html>
    <head>

    </head>
    <body>
        <div id='app'>
            <my-input v-for="inputItem" in inputList"
                v-bind:user-input="inputItem"><...>
            <create-input v-on:add-input="addInput"></...>
        </div>
        <script src="..."></script>
    </body>
</html>
```

取代 root / parent 的 prop 功能 component — data from root — M v-for 显示. — user-event from UIEntry — function in vue root — (查找 app.js (vue 文件))

```
var userInputSection = {
    // take data from parent
    props: ["user-input"]
    template:
        `<div ...>
            <p>{{ userInput.id }}</p>
            <p>{{ userInput.name }}</p>
            ...
        </div>`
```

kebab — counel

```
// register components
components: { 'my-input': userInputSection,
    'create-input': userInputEntry } });
```

注意: create-input btn: on-click 产生事件 'add-input'
input: v-model 输入 → 传出 object user Input 给 root, call addInput → 添加到 List. <List 数组> → <my-input> 中 v-for 读取 root 的 list 显示所有 item, 用 v-bind 把 props 的 user-input 传入 item, 使用 template 生成 html 内容.

```
var userInputEntry = {
    data: function() {
        return {
            // return object
            userInput: {
                userName: " ",
                userContent: " " }
    template:
        `<div ...>
            <input v-model="userInput.userName" type="text">
            <textarea v-model="userInput.userContent" type="text">
            // create user-event
            <button v-on:click="$emit('add-input', userInput)">...</button>
        </div>`};
```

使用 user-event 向父组件发送信息

user-event — parameter, local object.

```
var vue = new Vue({
    el: #app',
    data: { inputList: [{id: ..., name: ...},...] }
    methods: { addInput: function(userInput){
        // create let
        let input = { userName: userInput.userName};
        this.inputList.push(input); } },
```

ob 只使用 1 root instance, 注意: 在 methods 里的 function 中如果要调用 data 里的 attr., 要使用 (this) attr.