1. React + Spring Boot todoList

React = components + features

Components:

1.jsx是js的扩展在js文件里有类似html的code，负责view

2.js负责逻辑 3.css负责style 4.state存数据 5.props传递数据

virtual DOM: React’s representation of DOM in memory, 在memory所以make change很简单。React先找到Virtual DOM和DOM有什么区别，再把不同的地方在DOM上更新了

Features:1.Routing 2.Forms & Validations 3.Rest API Calls 4.Authentications

**介绍：**

这是一个用户可以注册登录去做todolist的网站，里面可以新增 修改 删除todo。

每个todo都有id 任务描述 目标日期 是否完成这些内容

**细节：**

用BrowserRouter去到不同的路由(有登录,登出,欢迎页,todoList页)，用switch确保只去到一个路由，如果路由地址不对就会去到一个ErrorComponent。每页都会有header和footer，header能被点击去到其他路由，但是会用sessionStorage(在inspect里Application里的storage下，里面是key value形式，把用户用户名作为value存。不用localStorage是因为他没有expiration time)去(setItem(登录) removeItem(登出) getItem(判断是否是登录的状态))判断用户的状态，只有在登录状态下才会有home, todolist和登出的元素在header，只有登出状态下才有login这个Link。

**1.Get用户的todolist**：首先数据在后端获取然后映射到一个URL上，把用户名设为path variable，之后在前端的componentDidMount用axios获取response，也就是用户的todos来设置指定用户的state。这个放大用于每次render完 delete完

**2.Delete用户todo**：点击button的时候传todo的id，之后删除的方法里会用用户名和id用axio带上指定URL(用户名和id作为path variable)去delete。后端里mapping这个url的方法(API)就会执行去删除数据。同时前端页面上会重新get用户的todolist来展现删除后的todoList

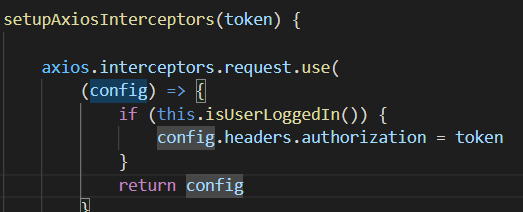
**3.update/add用户todo**：点击更新的button后会传todo的id然后带上这个id去到一个新的路由去改(this.props.history.push(`/todos/${id}`))这个todo，在新的页面里可以改todo的描述和目标日期，这里用了Formik这个包去对用户输入数据作一个验证，之后在点击保存这些修改的按钮的时候会把输入内容封装成一个object作为todo，之后看当前state的id是不是-1，如果是的话说明没有这个todo，是新增，就用axios和URL去调用后端新增的API，否则就调用更改的API。后端里也是判断todo的id，如果是-1就新增，如果不是就把原来的删了再新增。更改的时候URL要带上todo的id，新增的时候不用。更改完后会用history.push回到todoList界面

**Security：**

在后端加上jpa依赖并且import jpa的方法实现(网上找的framework)，

在前端登录的时候会用axios的带上账号密码(header)的post请求(也就是API\_URL/authenticate(spring里配置的)，

再带上header),(用BCryptPasswordEncoder的encode方法对用户的密码加密，底层大概是加点salt再转换成64进制)来得到(靠jpa方法) **JWT** (JSON Web Token),之后的request都会用这个JWT (Jott), 得到token后先在sessionStorage里放(‘authenticatedUser’,username), 之后会去用axios的interceptor(可以在request和response被then和error handle前处理request和response，这里就是用来带上header) 来把 ‘Bearer’ + token放到配置的header里。然后登录到欢迎界面



token也有过期时间,过期时间快到了可以发get请求(API\_URL/refresh)来更新token。对于授权其他请求(put delete get post)，只要在header里加上Bearer token就行。每个请求都会去到JWT authentication token filter

在后端pom里加上spring-starter-security依赖，(这样去到任何页面前都会有个form要输入账号密码)，这个账号密码可以在resources里配置成自己的，再加上spring security configuration(extends WebSecurityConfigurationAdapter)来disable CSRF和允许没有被authenticate的HTTP里带有Options的URL(因为在完整的请求被发送之前会有个请求call options请求去看有没有许可permission)最后在前端每次login的时候都会去做认证，也就是用axios带上账号密码组合加密(base64)后的token去请求到后端指定的URL，之后再做login需要的事情 -- 跳转到欢迎界面。Basic auth就是axios带上header这个参数，里面是Basic + encoded账号和密码(encode用window.btoa() -- base 64 encoding), 这里是用axios的interceptors来实现的

Basic auth的缺点：1.token没有过期时间，别人可以一直用 2，token里没有用户细节

**防止用户输入网址就跳过登录就到todolist界面**：创建一个新的component，里面判断也是用sessionStorage的component来判断用户是否登入。如果登入就用...this.props传入所有参数(path, component)，来去到指定路由。之后对于那些要防止用户直接输入网址的路由，我们可以用这个component

**JWT**(JSON Web Token) 确保服务的安全性，使其他用户不能输入URL就访问到别人的todolist。好处：1.可以设置过期时间 2.可以包含用户细节(header, payload-用户名字，创建时间等等,verify signature)，包括可以设置不同的加密算法

**知识点：**

**1.webservice** application request到web service，web service response。之间靠的是XML，JSON这样的data format，具体用什么要看service definition

service requestor把message放到MQ，service Provider会接受并把把产生response放回到MQ，requestor接受response

**2.REST：**1. Data Exchange Format没限制 JSON比较popular 2.transport只用HTTP 3.service definition用swagger这种

**3.dispatch servlet**接受bean的request，也就是看URL和method，dispatch servlet会找合适的controller去执行。之后得到bean再去invoke conversion，把他转变为JSON再把response发回

**4.Servlet和JSP**

1、不同之处在哪？

Servlet在Java代码中通过HttpServletResponse对象动态输出HTML内容

JSP在静态HTML内容中嵌入Java代码，Java代码被动态执行后生成HTML内容

2、各自的特点

Servlet能够很好地组织业务逻辑代码，但是在Java源文件中通过字符串拼接的方式生成动态HTML内容会导致代码维护困难、可读性差

JSP虽然规避了Servlet在生成HTML内容方面的劣势，但是在HTML中混入大量、复杂的业务逻辑同样也是不可取的

**MVC模式**（Model-View-Controller）是一种软件架构模式，结合servlet和JSP，扬长避短，把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）

Controller——负责转发请求，对请求进行处理

View——负责界面显示

Model——业务功能编写（例如算法实现）、数据库设计以及数据存取操作实现

在JSP/Servlet开发的软件系统中，这三个部分的描述如下所示：

过程：

Web浏览器发送HTTP请求到服务端，被Controller(Servlet)获取并进行处理（例如参数解析、请求转发）

Controller(Servlet)调用核心业务逻辑——Model部分，获得结果

Controller(Servlet)将逻辑处理结果交给View（JSP），动态输出HTML内容

动态生成的HTML内容返回到浏览器显示

MVC模式在Web开发中的好处是非常明显，它规避了JSP与Servlet各自的短板，Servlet只负责业务逻辑而不会通过out.append()动态生成HTML代码；JSP中也不会充斥着大量的业务代码。这大大提高了代码的可读性和可维护性。

**5.axios**可以异步处理请求asynchronous，用axois可以处理controller的URL，只要传URL进去就可以(用户名作为path variable)，之后可以对response.data等进行处理，比如设置为state的值, 如果controller里throw error也一样处理--error.data...

**6.执行顺序：**constructor -> render -> componentDidMount -> render(如果改变state就要再render) -> ComponentWillUnMount(view里component被remove了，比如切换页面了) componentShouldUpdate如果return false就不会更新view(render不会被call第二次)，但state改变了

**7.JPA**的替代品：问题：如何定义object和table

**JDBC** query: update todo set user=?, desc = ?, target\_date=? 如果是select就会返回一个结果，要对结果自己mapping到bean

**myBatis**可以用XML做这样相同的事情，而且mapping更简单 user=${user} {}里是bean

**8.JPA(Java persistence API)(**一个interface，Hibernate implements JPA，是一个具体实现**)**可以允许我们定义class和table的mapping，可以用SQL/JPQL去query数据库。有个entityManager(允许保存data到数据库)

注释有@Entity(要存到table就要加) @ManyToMany

@Id 说明是primary key

dao service(有insert get等)里@Transactional Sping创建和class implements一样interface的代理(要去对数据库做修改) @Repository说明要和数据库interact

**persistence context：**a synchronizer object that tracks the state of a limited set of Java objects and makes sure that changes on those objects are eventually persisted back into the database. 而entityManager不行，只负责创建时候

JPA需要default constructor

SLF4J(simple logging facade for java)

**9.H2(in-memory数据库)**，不同于Mysql，Oracle, H2不用安装，直接加了依赖了就能被创建，程序run -> 表、schema自动创建 -> 数据自动被insert -> 去localhost:8080/h2-console可以看见data

application stop -> 整个数据库销毁，从内存里remove

这些因为auto configuration：spring boot知道是否连接了in-memory数据库, 会自动初始化schema和删除schema(application stop)。hibernate JPA auto configuration

自动配置了entity管理data source和一个transaction manager

JPA repository interface有很多方法可以更好地管理entity，有save(create新的) findby findAll

具体操作：运行后在日志里找spring.datasource.url 之后在localhost:8080/h2-console网站里输入url就可以对数据库进行操作

1. React 汉堡点餐网站

**介绍：**

这是一个用于汉堡店的点餐网站，大体上说用户可以通过邮箱和密码注册登录然后按照自己喜好来定制汉堡，比如通过点击来加减各种配料的数量，然后相应的汉堡模型也会及时地显示在页面上。点完了之后用户需要输出他们的信息，联系方式，地址和配送方式等。最后这个订单会被存在每个用户各自的数据库里，用户也可以通过导航栏里点击订单来查看订单并且还有登出登入的按钮

**细节：**

1. 在index.js里用redux组合burgerBuilder, order, auth三个reducer，创建store去放这个总reducer，并且redux的thunk去实现middleware。用连接React和redux的Provider去连接然后在里面放react-router-dom的BrowserRouter，最后在里面放进App。

2.在App里放进checkout, orders, auth, logout, /这几个Route，用react-router-dom里的Switch去实现只运行第一个匹配的Route，这样用户点击的时候就能到正确的route

3.在每个路由里，分别用了不同的redux的reducer去实现。通过引入一个总file(index.js)，里面有各种方法，每个方法会去派遣对应的reducer里的函数去改变一些变量或者实现一些响应。比如加减ingredients，认证用户等等。同时在一些函数使用firebase来把存放数据和用户标识。

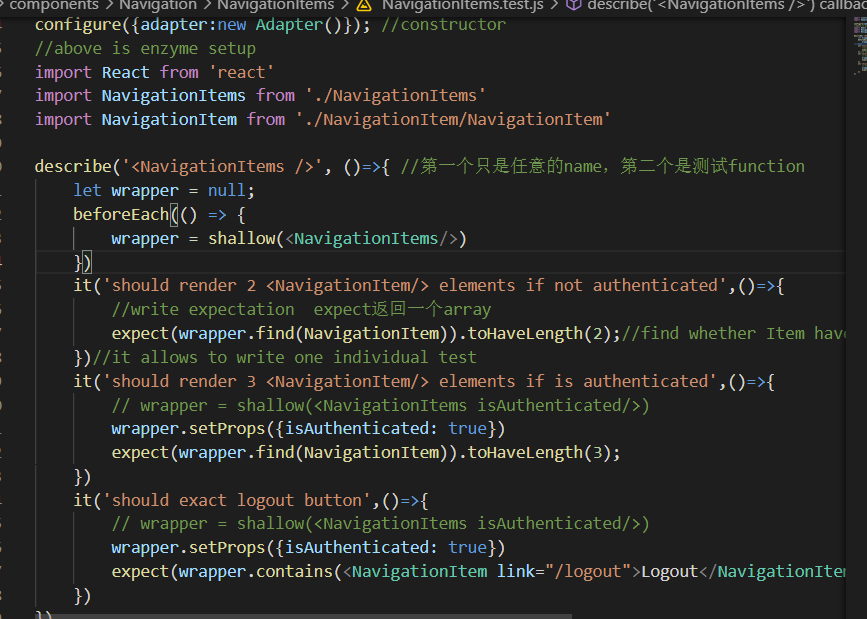
**防止刷新就要重新验证**：在App这个总file里的didMount函数里做检查，如果检查到本地储存了用户token就再从本地存储获取用户id来登录否则到登出的路由，这样刷新就自动登录了

**登录**：首先用户点击switch to signIn，状态里signIn变量变成true，之后用户输完邮箱密码点submit按钮就会设置状态里的邮箱和密码变量，然后到了actions文件夹的文件里检查这个isSign变量，axios根据是不是登录还是注册post不同的url来提交不同数据，服务端会根据客户端传过来的用户id查询数据库，获取secret，然后重新生成一份token，之后比较客户端传过来的token和服务端生成的token比较，相等的话返回数据到客户端，然后设置定制汉堡的路由到定制汉堡页面，不相等的话界面上就会显示密码错误信息

**困难**：在加完料的时候用户点击continue会弹出一个订单总结，上面每个料的数量包括价格，这时候用户有两个选项，一个是点取消就是返回继续加料，一个是继续去到结账页面。我遇到的难点是怎么简单地处理用户点击取消的函数里把页面回到加料页面，点击继续的函数里把页面设置成要跳转的页面。

**解决**：先打印出当前的props，然后在检查页面里看BOM(浏览器对象模型)里的history对象，history里面又有很多的属性可以用，这里我就用了this.props.history.goBack()去回到之前加料页面，然后用了this.props.history.replace(里面加上结账页面的相对路径)去到结账页面

**测试：**单元测试



知识点：

1. axios：基于Promise用于浏览器和NodeJs的HTTP客户端，符合ES规范
2. Promise是异步编程的一种解决方案
3. 把组件看成一个函数，那么他接受props作为参数，内部由state作为函数的内部参数
4. JSX是JS的语法扩展，ES6是JS下个版本标准
5. 相关知识：为什么要引入(import)React，但在代码里没有用，

import React from ‘react’ 实际上是

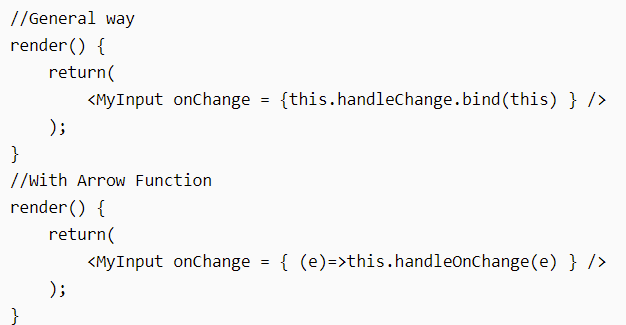
React.createElement(component, props, ...children) ，JSX(Javascript XML)为此提供了语法糖(指计算机语言中添加的某种语法,这种语法对语言的功能并没有影响,但是更方便程序员使用，相当于成语，用更简练的语言表达较复杂的的含义，提升交流效率),所以只要import React就行

1. DOM就是文档对象模型。在这个文档对象里，所有的元素呈现出一种层次结构，就是说除了顶级元素html外，其他所有元素都被包含在另外的元素中。VDOM虚拟DOM是真实DOM的内存表示。与真实DOM同步比如通过ReactDOM
2. 类组件和函数组件的区别：类组件不能改自己的props，但能改state，redux传进来state到props里，父组件改变，子组件视图会更新。函数组件接收一个props返回一个React元素。函数组件性能高，因为类组件使用时要实例化。类组件有this和生命周期
3. render函数不往DOM树上渲染或者装载内容，只是返回JSX对象，由React库来根据这个返回对象决定如何渲染
4. redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，主要有三个核心方法，action，store，reducer，工作流程是 view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据， redux 中只能定义一个可更新状态的 store，redux 把 store 和 Dispatcher 合并,结构更加简单清晰
5. 新增 state, 对状态的管理更加明确，通过 redux，流程更加规范了，减少手动编码量，提高了编码效率，同时缺点时当数据更新时有时候组件不需要，但是也要重新绘制，有些影响效率。一般情况下，我们在构建多交互，多数据流的复杂项目应用时才会使用它们
6. redux 缺点：一个组件所需要的数据，必须由父组件传过来当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断。
7. 
8. 生命周期函数和组件地生命周期：

组件第一次挂载，获得父组件的props和初始的state，接着经历 ComponentWillMount, render, ComponentDidMount，挂载完成后组件的props 和state的任意改变都会导致组件进入更新阶段, props改变进入 ComponentWillReceiveProps, 再进入ComponentShouldUpdate判断是否要更 新，如果是state改变则直接进入ComponentShouldUpdate判断，默认为true， 要更新的话进行WillMount, render, DidMount三个阶段，不更新组件继续进 行，挂载时，进入ComponentWillUnMount，之后进行卸载

1. \*初始渲染阶段：\*这是组件即将开始其生命之旅并进入 DOM 的阶段。
2. \*更新阶段：\*一旦组件被添加到 DOM，它只有在 prop 或状态发生变化时才可能更新和重新渲染。这些只发生在这个阶段。
3. \*卸载阶段：\*这是组件生命周期的最后阶段，组件被销毁并从 DOM 中删除。
4. componentWillMount： 在渲染之前执行，用于根组件中的应用程序级别配置。
5. componentDidMount： 仅在客户端的第一次渲染之后执行。 这是AJAX请求和DOM或状态更新应该发生的地方。此方法也用于与其他JavaScript框架以及任何延迟执行的函数(如 setTimeout 或 setInterval )进行集成，在这里使用它来更新状态，以便我们可以触发其他生命周期方法。
6. componentWillReceiveProps： 只要在另一个渲染被调用之前更新 props 就被调用。 当我们更新状态时，从 setNewNumber 触发它。
7. shouldComponentUpdate： 确定是否将更新组件。默认情况下，它返回true。如果您确定组件在状态或道具更新后不需要渲染，则可以返回false值。这是提高性能的好地方，因为如果组件收到新的道具，它可以防止重新渲染。
8. componentWillUpdate： 在由shouldComponentUpdate确认返回正值的优点和状态更改时，在重新渲染组件之前执行。
9. componentDidUpdate： 通常用于更新DOM以响应属性或状态更改。
10. componentWillUnmount： 它将用于取消任何传出的网络请求，或删除与该组件关联的所有事件侦听器。

React Context:当你不想在组件树中通过逐层传递 props 或者 state 的方式来传递数据时，可以使用 Context 来实现 跨层级 的组件数据传递。

1. props介绍Props 是 React 中属性的简写。它们是只读组件，必须保持纯，即不可变。它们总是在整个应用中从父组件传递到子组件。子组件永远不能将 prop 送回父组件。这有助于维护单向数据流，通常用于呈现动态生成的数据。
2. 箭头函数
3. 
4. GroundRating
5. Express是NodeJS Web应用框架，提供了很多特性和HTTP工具，比如路由处理，提供中间件(可以修改请求和响应对象)，来帮助创建Web应用
6. 在app.js文件里先连接mongoose然后加载(require)express，之后用use去到不同的路由，然后在不同的路由里分别执行不同的操作，比如get到了new的route就去执行创建帖子的页面的模块。在每个页面几乎都用了ejs-mate去统一页面的布局，比如导航栏，注脚和一些flash

知识点：

1. 哈希密码是在用户密码后面加salt(加强隐蔽性)，也就是随机的值，再哈希，这也别人即使通过哈希完的值去暴力配对密码也不能知道正确的密码，破解难度大大增加
2. Bcrypt是这个哈希function，同一个密码每次生成的hash都是不一样的

在加密的时候，先随机获取salt，然后跟password进行hash。在下一次校验的时候，从hash中取出salt，salt跟password进行hash，得到的结果跟保存在在数据库的hash进行比较。

BCrypt比MD5更安全，但加密更慢

1. EJS是一个JavaScript模板库，用来从JSON数据中生成HTML字符串。
2. npm是Java的包管理程序，提供JS的各种第三方框架的下载
3. Node.js是一个后端的js运行环境，用来解释js代码
4. ajax的全称是AsynchronousJavascript+XML 异步传输+js+xml,点一下不会刷新页面
5. 出现404（链接指向的网页不存在）的排查

一般使请求的URL错了

先检查url有没有拼对

然后再一些链接前后加一些console log看看去找哪里除了问题

app.use((req, res) => {

    res.status(404).send('404 not found!!!')

})

1. JQuery与react不同，是事件驱动，其他事数据驱动，他的UI里还有交互逻辑，所以逻辑混乱，JQuery操作真正的DOM，React根据virtual DOM来修改真正的DOM，JQuery简化DOM操作以及动画，降低了js门槛
2. password是在app.js里使用，提供了很多函数简化注册流程，会hash密码
3. 用响应的params得到每个帖子的id，用findByIdAndDelete删掉帖子
4. mongoose创建schema(定义变量以及他的类型)，创建model然后连接到数据库
5. 把图片传到cloudinary，然后把图片的文件名和url传到对应的帖子，再在网站里展示出来

**登录**：首先设置响应的user为请求的user，根据user存不存在来在导航栏上显示登录或注册和登出按钮，用户点击不同的按钮会到不同路由，不同路由有不同界面，在登陆界面用户输完验证信息点登录的话把用户信息通过

passport-Mongoose/password的一些方法传到数据库检查对不对然后会对应再到不同界面

**困难**：

**解决**：

**测试：**Postman

XHR：一种浏览器API，简化了异步获取信息的过程

从浏览器导入request到postman：检查页面里选网络里的XHR - 勾选preserve log - 完成操作 - 右键Name里发送了的请求 - 选copy-copy as cURL - 到postman点左上角的import - 选paste raw text - 复制进去 - 要填的信息都有了

cookie：上一个请求返回的cookie会加到下一次的请求中

cookie是以header的值发送的，postman提供了查看这些值的功能，值里打包了很多东西

troubleshooting：1.检查URL 2.检查所访问的服务是否可用 3.在浏览器输入API域名看能否 访问 4.协议是否正确(http/https)，是https要在设置里disable掉SSL整数 验证 5.用self signed证书，需要在设置里加证书。还不行就去看console

collections：可以保存请求，collections能帮助组织和管理这些请求，还能建文件夹

还能分享--通过导出文件或分享postman链接

pm.test(测试名+回调函数(断言))-写测试的函数，发生error是非阻塞式工作方式(一个报错不 会影响别的测试)，可以包含多个断言

路径(path)里 :加变量名字，postman用value替换他(不能删除); query参数用键值形式增删

全局变量:设置: 在test里pm.globals.set(“boardID”,response.id) pm.globals.unset(“variable\_key”) //清楚变量

使用：Params里{{全局变量名字}} 在test里pm.globals.get(“boardID”)

环境变量(url)：作用域比环境变量少，方便切换不同设置，在请求构造器和全局变量一样

设置和使用只是把globals换成了environments pm.environments.set()

一般只用环境变量或者只用全局变量或者集合变量(只能用pm.variables.get())

都可以用pm.variables.get()

把环境变量和全局变量的initial value设置成YOUR KEY等分享时就不会分享敏感信息

bulk edit批量更改params的值

工作顺序：pre-request test - 进行HTTP请求 - 运行测试

作用域：local <data<environment<collection<global, 优先级相反，local最大

集合变量：值不变，只有一个环境

数据变量：用不同的数据集执行一样的request

Debug：先consle.log去postman 控制台去看详细信息，不存在时是null

Chai：Postman的断言库，让断言更易读

写断言：先解析响应体，比如pm.response.json()，cheerio(pm.response.text())

用hasOwnProperty()来找出有某个特定属性的key，一般这个key是每次send就变

自动化测试工具：Collection runner -> Postman Monitors->Newman -> Newman+Jekins -> Newman+其他CI服务器

Postman Monitor就是在网站上按照间隔时间自动跑测试，还可以设置邮件提醒

但是不能导入全局变量，之恩导入环境变量

Newman(nodejs库)：CLI(开发命令行)工具，让postman的json文件可以在命令行执行的 插件

Postman API：通过API和collection交互的一种方式(生成API key)

从newman访问postman collection的3种方式：

1. 分享链接(collection修改后要更新)
2. 2.导出json文件(修改后要更新)
3. 3.用postman API和要测试的collection的ID得到一个URL，可以在cmd里 就运行monitor的test (postman更新后cmd测试结果也同步更新)

在newman里使用环境变量：先把环境变量导出成json文件，在跑cmd里Newman url后面加上--environment 和json文件的路径

Jenkins就是在config里输入command--newman run，然后指定环境和reporters和报告位置来生成报告，点save然后点build来运行，如果公布了报告+下载了plugin(html, Junit)(在post-build action里),就能看到测试的更详细的技术信息。Postman里更新也会同步更新测试结果

gitlab-runner(项目自动话驱动)可以执行一个YAML文件，当push代码后会执行写好的脚本，由脚本执行进一步操作，从而实现自动话

Timcity分成构建代理(构建都是通过代理服务执行)+网页版控制端(通过网页进行管理)，其他和Jenkins差不多

Docker(CLI)：可以虚拟化的容器技术，让电脑保持原始状态，可以用虚拟的方式安装jenkins，不会被这种软件污染电脑。跑newman时候把postman的collection和环境变量导到桌面，再在docker里运行

用Git管理postman collection：做修改-导出文件-commit文件-push修改

其他人：pull修改-导入文件-做修改-导出文件-commit-push

Git优势：免费，有修改历史，所有数据保存在本地，可以code review，修改少时候好

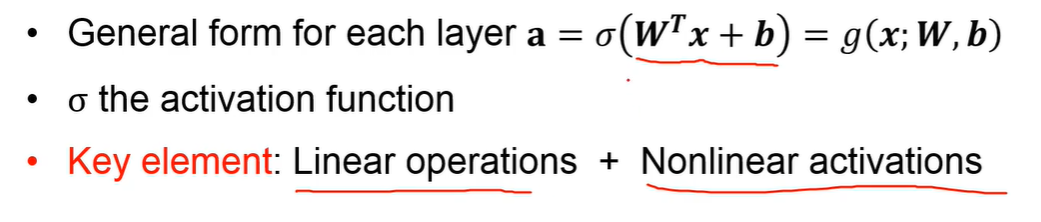
劣势：手动流程，不容易review json文件

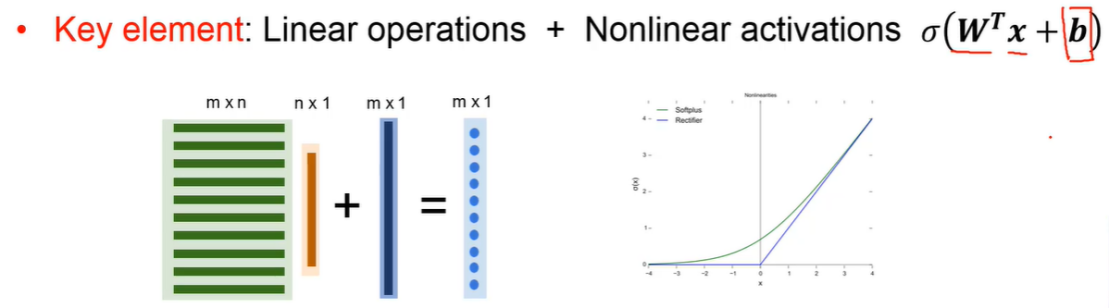
1. 人工智能
2. 先安装Tensorflow再利用keras的API建模，再加layer，训练，再用测试的数据来测试模型，还有一个预测类的方法用来显示图片在每个类的概率
3. 先安装PyTorch，用forward、backward、loss computation一些方法训练模型，同样加了不同种类的层比如Flatten、Dense层ReLU activation函数(来自torch API)

Tensor类似numpy数组用来解读模型的输入和输出但可以在GPU上运行

深度学习：利用多层次的分析和计算手段得到结果的一种方法

把一个函数分解成非线性函数的集合，每个函数代表使用一层神经网络





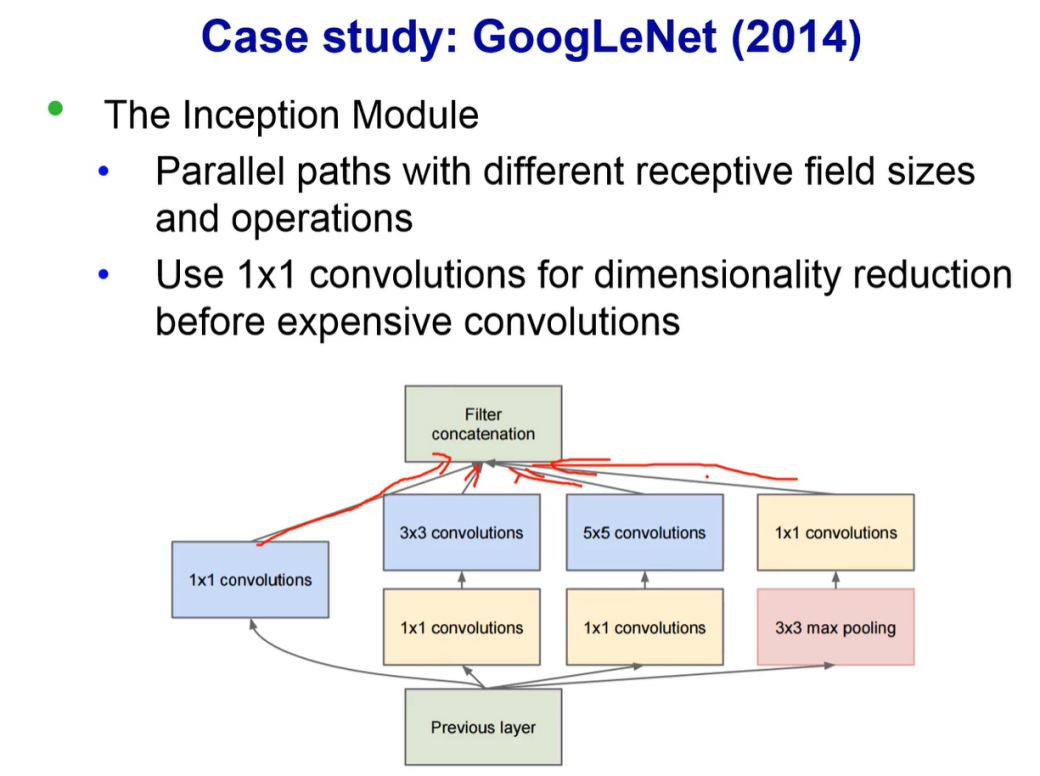
activation函数有好几种选择

线性函数可以选convolution(滑动窗口，每个窗口一个结果，最后得到一个总结果，比如一个图像，好处：稀疏连接，每个输出只取决于少量输入)

pooling增大感受野，让convolution看到更多信息，但在降维过程中丢了一些信息--降低了分辨率

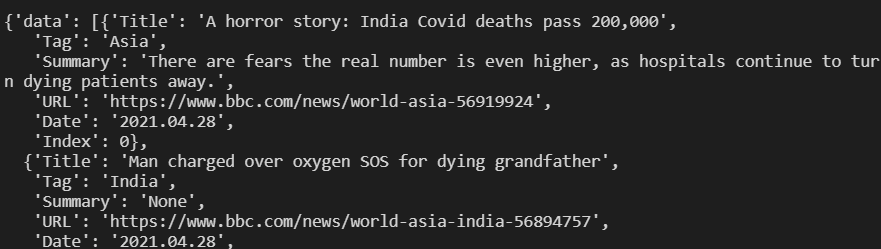
从对简单的图像进行分类

MNIST数据集：7万张照片，6训练，1万测试神经络



1. 爬虫

把数据录入到csv中，有6个section，再把数据从所有csv文件里拿出来按section整合成6块，再对数据进行格式处理



再把数据按section生成并放到6个json文件，再对每个json文件创建一个表把json里面的数据插入到表中，这些数据就都到db文件里了

再通过terminal把数据生成.sql --- 从sqlite导入到MySQL souce+BBC\_NEWS.sql

到路径下--sqlite3--.open BBC\_NEWS.db--.output BBE\_NEWS.sql--.dump

MySQL里输入指令要加分号；

1.反爬机制：1.设置限制访问：一段时间内一个IP不能访问

2.返回验证码：要求用户输入验证码，防的是机器

3.爬虫陷阱：让爬取的信息和本网站无关

解决反爬机制：设置不同IP代理这样相当于不同用户在爬网站

首先去神龙http网站获取大量IP 比如115.208.46.91:35508

再从中随机取去爬，如果有验证机制，每爬完一页设置程序暂停时间

困难：把title和summary插入到表的时候发现一直有Error: near “A”:syntax error，看了好几遍建表和插入数据的语法没有错误

解决：上网搜发现可能是title和summary的内容里有引号导致的，就把里面引号用replace去掉