**推荐的技术栈**

**沈备军**

**2023**

# 互联网产品的技术栈推荐

## Web前端

1. Html, CSS, JavaScript, TypeScript
2. React或Vue（js框架）/Flutter（dart框架）
3. 使用WebPack/npm/yarn等工具打包管理

## 移动前端

1. 微信小程序, 或
2. 跨平台App：React或Vue（js框架）/Flutter（dart框架）
3. Ios App: objective c++/swift
4. Android App: Kotlin/java

## 服务器后端

Java框架：Spring技术栈

1. 使用Spring来实现(Spring Boot 和 JPA) 或 使用类似框架
2. 使用Maven等工具打包管理 （建spring boot工程默认就是maven的工程）

Python框架： Django、Flask、FastAPI

JavaScript框架：node.js

微服务技术栈：

1. 框架：Spring Cloud 或Dubbo
2. 容器：Docker

定义和运行多容器应用程序的工具：Docker Compose

1. 集群管理: kubernetes

含Kubeadm、Kubectl、Kubelet工具

1. 反向代理：Nginx
2. 消息中间件: Kafka等

## 前后端通讯

REST （Restful web services）、Ajax

## 数据库

1. 关系型数据库：MySQL/Mariadb
2. NoSQL数据库：MongoDB
3. 内存数据库：redis/memcached
4. 图数据库：Neo4j
5. 分布式文件系统：nfs等
6. 全文搜索：Elastic Search

## 鉴权

验证用户是否拥有访问系统的权利。

1. Springboot Security
2. Session、Cookie
3. JWT (JSON Web Token)

## IDE开发工具

1. IntelliJ / WebStorm/PyCharm/ Eclipse
2. Visual Studio Code （即VSCode）
3. Android Studio
4. WeChat Studio
5. HbuliderX

## 举例

一个在线购书系统，前端采用JavaScript语言和React框架；后端使用Java语言和tomcat服务器, Spring Boot + Data JPA框架，数据库用MySQL和MongoDB；前后端采用REST通讯；鉴权采用Springboot Security。

若采用Python, 则后端最简单的是采用Flask框架。

若采用JavaScript，则后端采用node.js框架。

# 游戏的技术栈推荐

1. 编程语言和框架
   * C#, unity引擎
   * C++/蓝图，unreal引擎
   * C#, monogame引擎
   * C++, cry engine引擎
2. 前后端通信：socket
3. IDE工具：VS Code

举例：一个手游软件，前端用unity引擎，使用xlua做热更，不少功能都是使用xlua来编写的，只有网络、协议、核心通用工具是用C#来做的；服务器后端使用Java语言和netty服务器，数据库用MySQL；前后端通讯协议使用google的protobuf（在git上可下载，支持xlua和 java等）。

# 机器学习的技术栈推荐

## 语言和框架

1. 编程语言：Python
2. IDE开发工具： PyCharm/ Spyder/ Intellij
3. 深度学习框架： PyTorch / TensorFlow

## NLP库

* 清华的THULAC（C ++ / Java / Python）: <http://thulac.thunlp.org/>
* 中科院的NLPIR（Java）: <https://github.com/NLPIR-team/NLPIR>
* 哈工大的LTP （C ++）: <https://github.com/HIT-SCIR/ltp>
* 复旦的FudanNLP（Java）: <https://github.com/FudanNLP/fnlp>
* PaddleNLP: <https://github.com/PaddlePaddle/PaddleNLP>
* JieBa Python中文分词组件: https://github.com/fxsjy/jieba

--------------------------------------------------------------------------------------

* Stanford NLP库 (Python) : <https://github.com/stanfordnlp/stanfordnlp>
* Stanford CoreNLP库（Java）: <https://github.com/stanfordnlp/CoreNLP>
* Spacy 自然语言文本处理库 (Python)：https://spacy.io
* NLTK 自然语言工具包 (Python)：http://www.nltk.org/
* Pattern 自然语言处理工具包: https://github.com/clips/pattern
* Polyglo 多语言NLP工具包: https://github.com/aboSamoor/polyglot
* Allennlp 基于Pytorch的NLP研究库：<https://allennlp.org/>
* hugginface transformers： <https://github.com/huggingface/transformers>
* Facebook seq2seq: <https://github.com/pytorch/fairseq>
* 预训练语言模型Bert：<https://github.com/google-research/bert>
* 预训练中文语言模型：<https://cpm.baai.ac.cn/>
* 预训练编程语言模型CodeBert/GraphCodeBert：https://github.com/microsoft/CodeBERT

## 机器学习库

* Hugging Face 开源AI模型库 <http://www.huggingface.co>
* GenSim 文本相似度计算包：https://radimrehurek.com/gensim/
* TextBlob 文本处理工具：https://textblob.readthedocs.io/en/dev/
* Scikit-learn 开源Python机器学习工具: https://scikit-learn.org/stable
* WEKA： <http://www.cs.waikato.ac.nz/ml/weka/>

## 云上的API服务

* 百度： <http://ai.baidu.com/>
* 科大讯飞：<https://www.xfyun.cn/>
* 腾讯：<https://ai.qq.com/>
* 阿里：https://ai.aliyun.com/
* Google：<https://cloud.google.com/products/ai>
* Amazon：<https://aws.amazon.com/machine-learning/ai-services>
* Microsoft：<https://azure.microsoft.com/en-us/services/cognitive-services>
* IBM：<https://www.ibm.com/watson>

## 举例

使用Spyder作为IDE，用Python语言，在程序中引入PyTorch作为深度学习框架，使用JieBa库进行分词，随后构造模型并进行训练。

# 大数据处理的技术

1. 并行计算框架：Hadoop / Spark / MapReduce
2. 文件数据库： MongoDB/ HBase
3. 图数据库：Neo4j
4. 内存数据库：voltDB
5. 数据仓库：Hive
6. 数据挖掘：Mahout/ Apache Hama
7. 数据分析：datameer
8. 图处理：Apache Giraph
9. 图可视化：Gephi
10. ETL：Sqoop/ Talend
11. 日志收集：Flume
12. 实时查询：Cloudera Impala/Apache Drill

# 分布式系统的通信编程技术

1. 远程过程调用（Remote Procedure Call, RPC）

* 实现了一对一的请求/响应式的组件交互。通过二进制消息进行通信，因此性能比RESTful等更好、延迟更小。例如gRPC (google RPC), 支持多种语言（如Java、C++、C#、Go、Node、PHP、Python、Ruby）。还阿里开源的Apache Dubbo 、腾讯开源的Tars 、百度开源的bRPC等。

1. RESTful接口

* 实现了Web服务的调用接口，使用HTTP协议作为进程间（客户端进程与服务进程之间）通信机制，并使用XML或JSON格式来表示资源信息。优点：和RPC相比，编程简单。例如采用Spring Boot框架来实现。

1. Socket

* 在TCP/IP上的网络编程。与请求/响应式的组件交互不同，Socket可实现双向通讯，能用来实现聊天室。例如 采用java的netty，golang的net包，C++的Sogou Workflow等来实现。

1. 基于消息的通信

* 使用消息队列（Message Queue，简称MQ）来实现。消息的发送方（消息的生产者）将消息放入消息队列中。消息队列作为一种容器保存消息。消息的接收方（消息的消费者）从消息队列那里获取消息并进行处理。消息队列支持两种消息通道模式：点对点模式和发布-订阅模式。能应用于以下五种场景：异步处理、应用解耦、流量削锋、日志采集、消息通讯。例如ActiveMQ、RabbitMQ，Kafka等。

# 通用的开发工具推荐

除了IDE外，推荐以下工具：

1. 版本管理工具：Git，Git workflow
2. 项目管理工具：Github Issue, Jira, Git 敏捷项目管理工具，禅道等
3. 单元测试工具：xUnit，如JUnit（Java），CppUnit（C++），CUnit（C）, DUnit （Delphi ），NUnit（.net），PhpUnit（PHP），PyUnit/unittest (Python)，Jest (Javascript)等
4. 集成测试工具：Postman、 SoapUI等API测试工具；SpringBoot Test;

React Native集成测试工具Cavy等

1. 系统测试工具
   * + 性能测试工具：JMeter/ Locust/ LoadRunner
     + 功能测试工具：selenium
     + UI测试工具：Cypress
2. API一体化协作平台：Apifox

Apifox = Postman + Swagger + Mock + JMeter

只需要定义好API文档，API 调试、API 数据 Mock、API 自动化测试等功能就可以直接使用，保证前后端沟通顺畅，可以自动化生成接口请求代码等。

最佳实践：https://apifox.com/help/overview/best-practice

**举例**：一个Web应用系统，采用Github作为版本管理和项目管理平台，前端的单元测试采用Jest，后端的单元测试采用Junit, 性能测试采用JMeter.

# 互联网运维的技术栈推荐

## 持续集成和持续部署CI/CD

Github Action; Bamboo; Jenkins

## 自动化运维

Ansible、Ansible playbook

## 运维监控

1. Prometheus
2. Grafana
3. Springboot-actuator
4. ELK (Elasticsearch、Logstash和Kibana), ElasticAPM

# 致谢

本列表的整理得到了任锐老师、杨旭波老师、陈昊鹏老师等的帮助，在此表示感谢。