



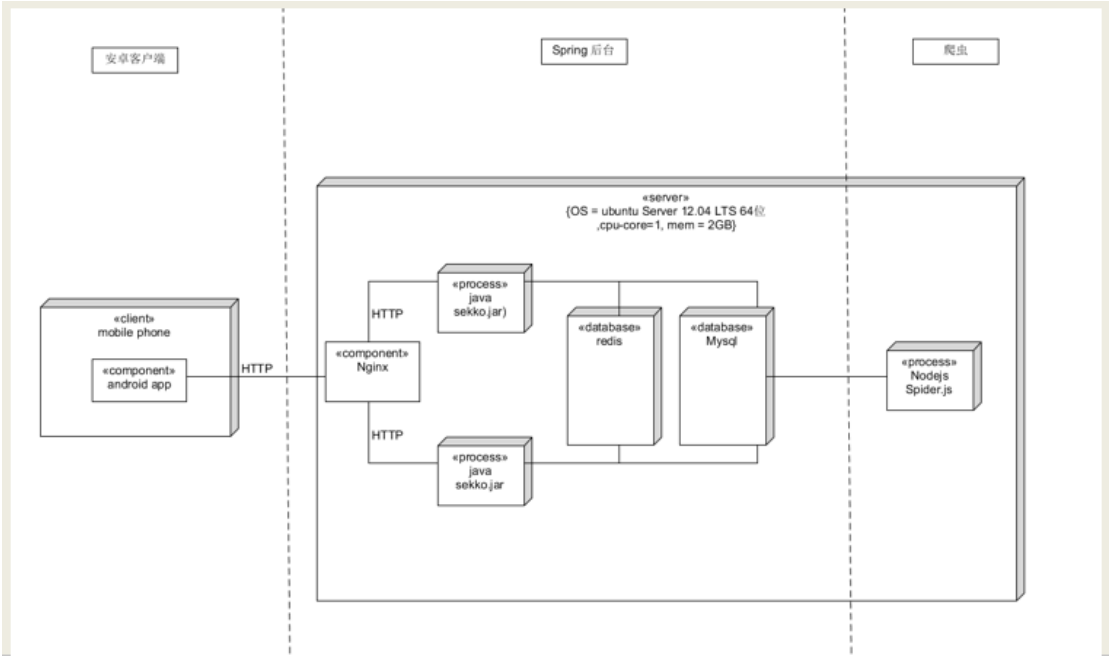
# 软件设计文档

----Sekko 购票 APP

负责人:王东升 史家瑞

1 架构设计

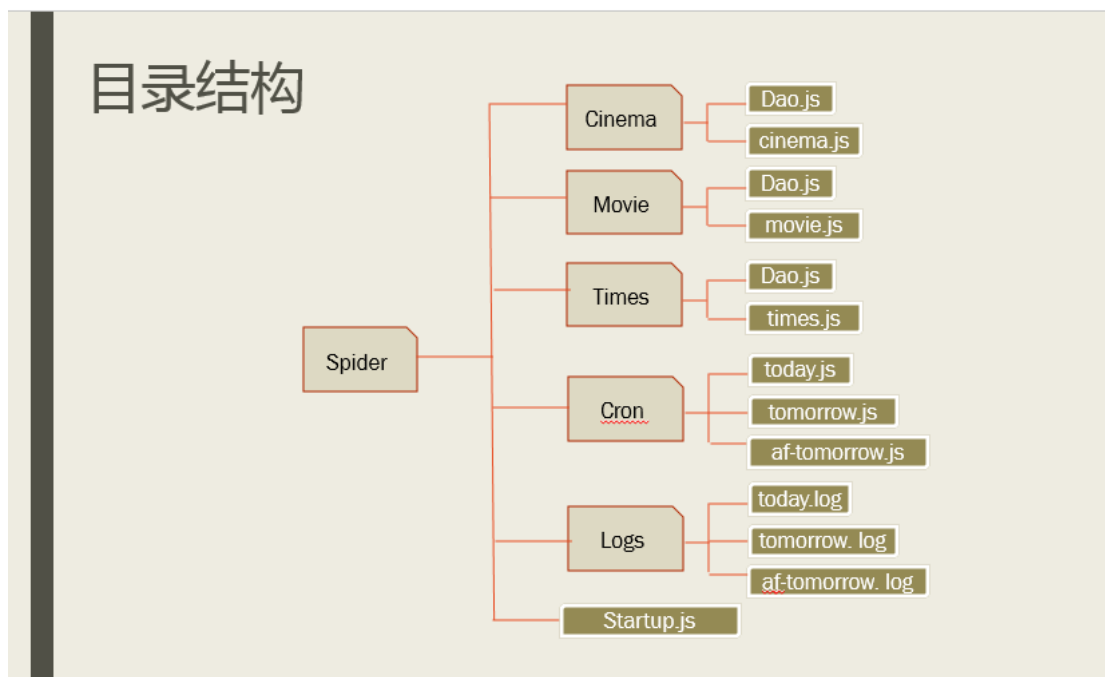
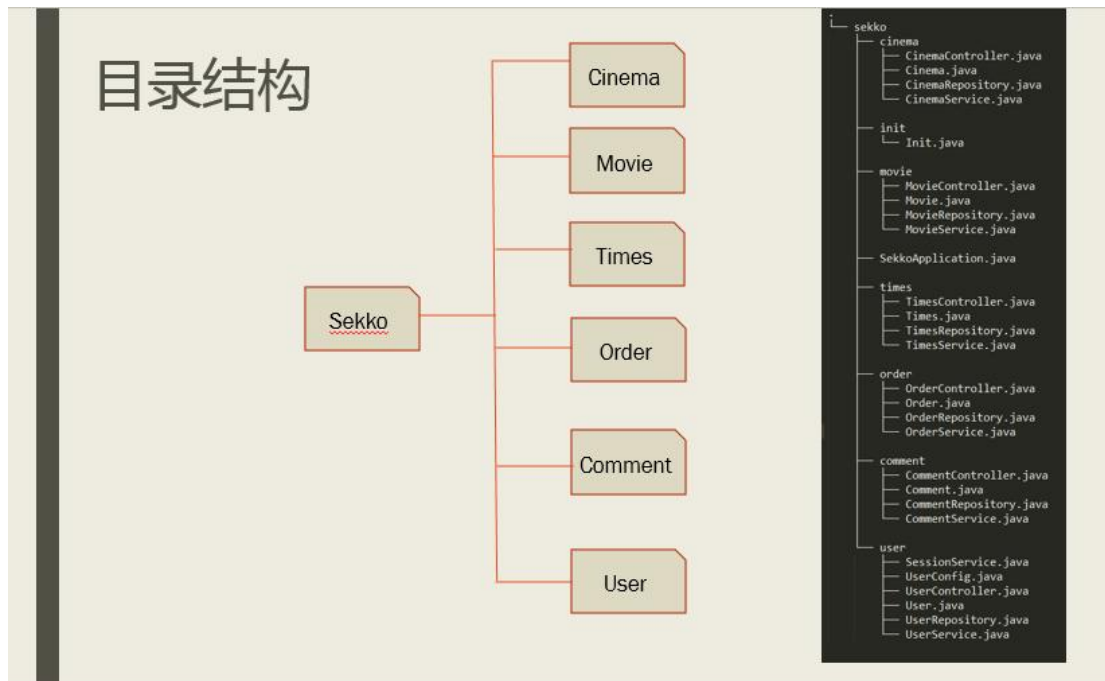
1.1 整体架构



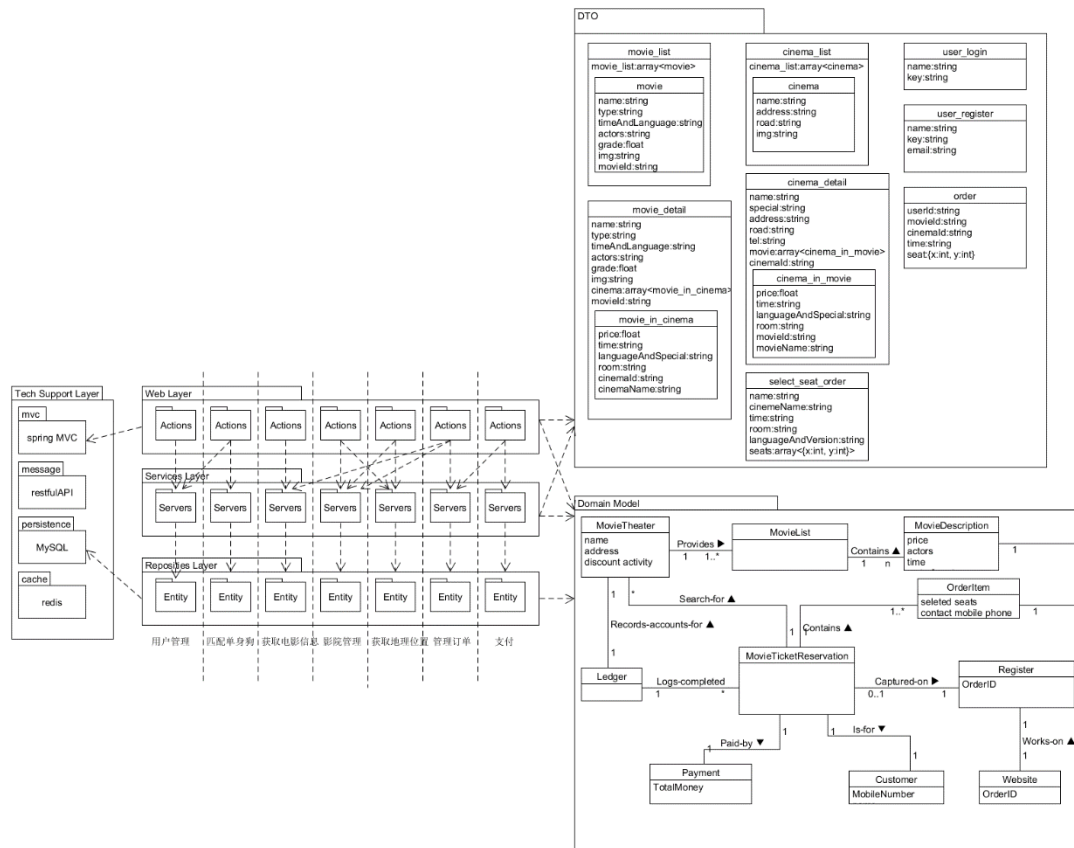
1.2 RUP 4+1 表格

非功能需求		
约束	运行期质量属性	开发期质量属性
要求能在常见的浏览器运行	易用性	易理解性
要求适配各种终端	高性能	模块间松散耦合

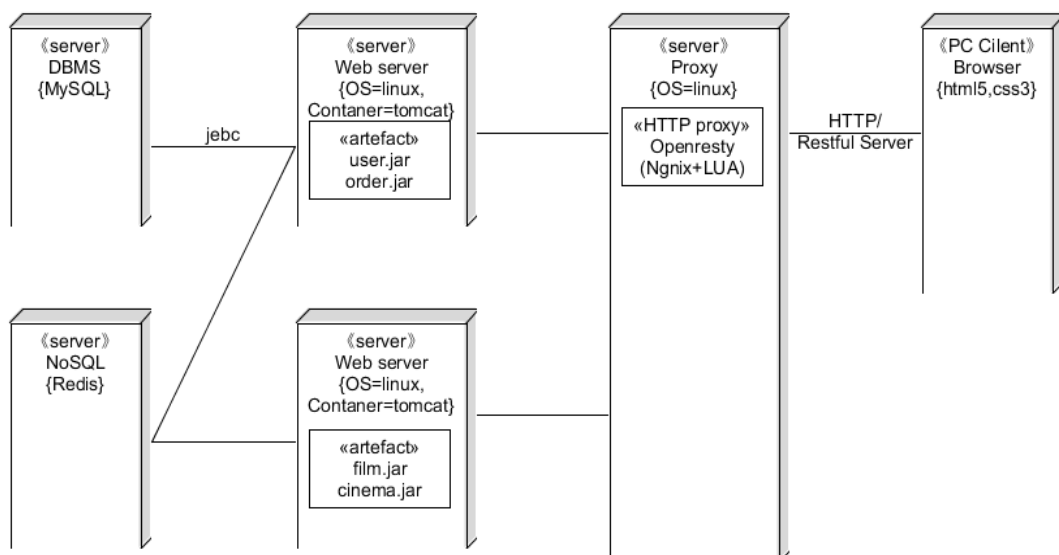
### 1.3 目录结构



## 1.4 逻辑模型



## 1.5 部署图



## 2 技术总览

### 2.1 对 SEKKO 技术架构的基础要求

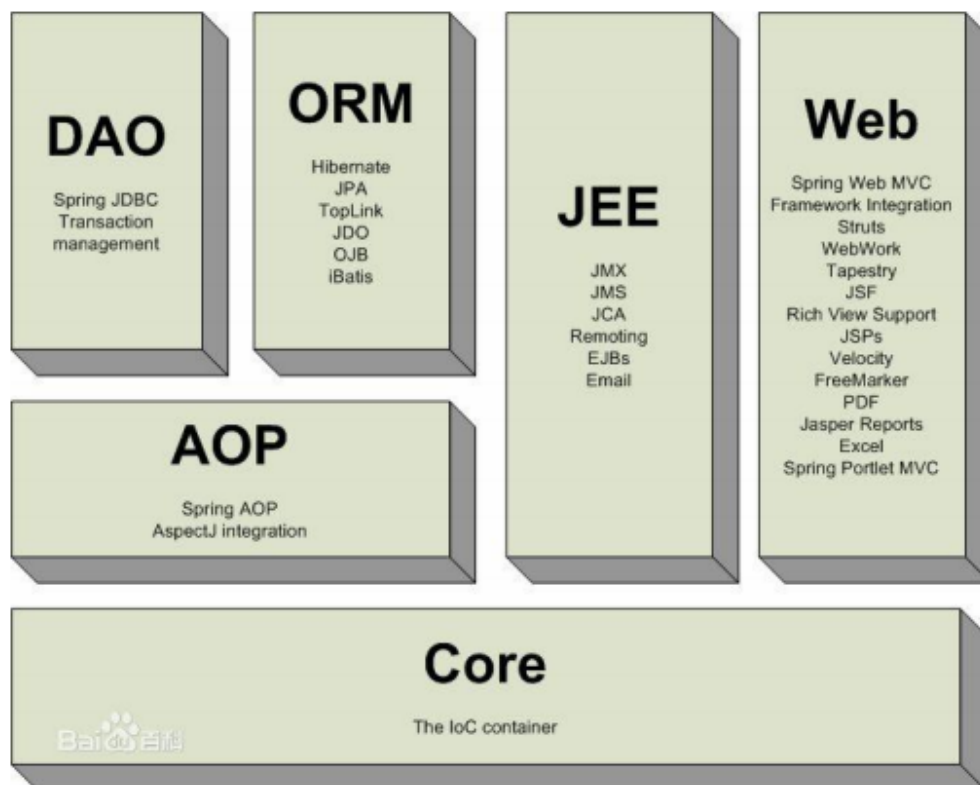
- 高稳定性
- 高性能
- 高维护性

### 2.2 整体框架



使用 Spring Tool Suite 管理开发项目，使用 Spring MVC 开发架构。

1. Spring 能有效地组织中间层对象。
2. Spring 实现针对接口编程
3. MVC Web 架构提供一种清晰，无侵略性的 MVC 实现方式
4. 简化访问数据库时的处理
5. 可以不直接操作 JTA 来对事务进行管理



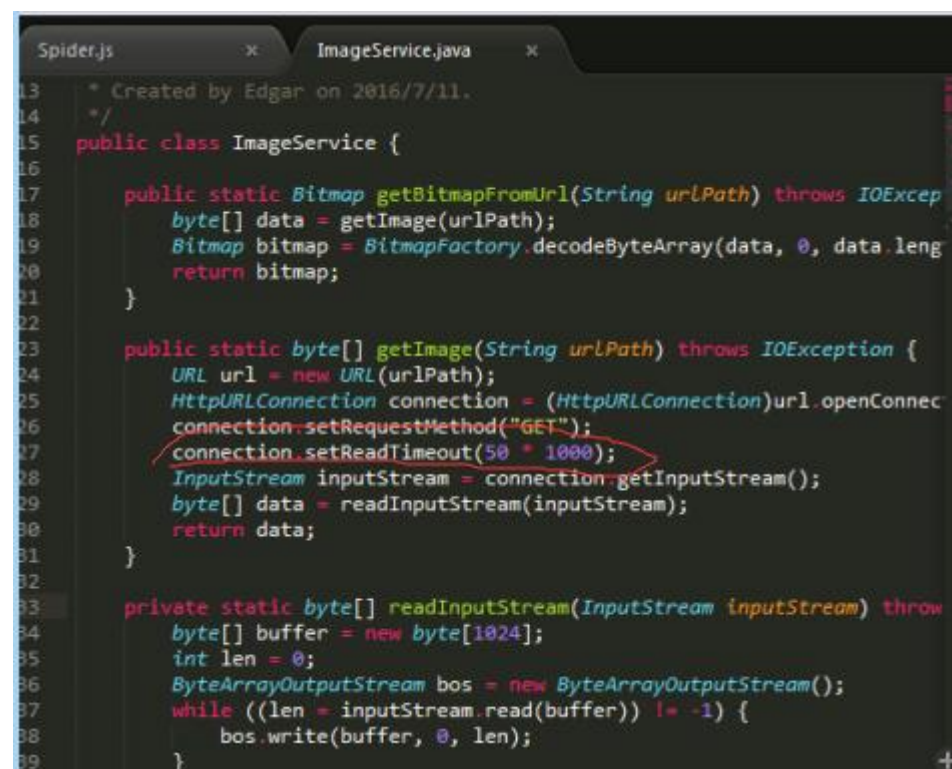
### 2.3 前端技术

1. 使用 Reactjs 创建 html 组件，通过 ES6 中的 fetch 函数获取动态数据
2. 通过 seajs 引进 js 模块，同一使用 ES6 标准的 js
3. 通过 jade, less 编写页面
4. 使用 gulp 作为自动化工具，编译 html 和 css，压缩 js, css
5. 图片使用懒加载

### 2.3.1 通过 ES6 中的 fetch 函数获取动态

```
// This is the function that is called to fetch the next token. It
// is somewhat obscure, because it works in character codes rather
// than characters, and because operator parsing has been inlined
// into it.
//
// All in the name of speed.
//
pp.readToken_dot = function () {
  var next = this.input.charCodeAt(this.pos + 1);
  if (next >= 48 && next <= 57) return this.readNumber(true);
  var next2 = this.input.charCodeAt(this.pos + 2);
  if (this.options.ecmaVersion >= 6 && next === 46 && next2 === 46) {
    // 46 = dot '.'
    this.pos += 3;
    return this.finishToken(_tokentype.types.ellipsis);
  } else {
    ++this.pos;
    return this.finishToken(_tokentype.types.dot);
  }
};
```

### 2.3.2 图片懒加载



```
Spider.js x ImageService.java x
13  * Created by Edgar on 2016/7/11.
14  */
15  public class ImageService {
16
17      public static Bitmap getBitmapFromUrl(String urlPath) throws IOException {
18          byte[] data = getImage(urlPath);
19          Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
20          return bitmap;
21      }
22
23      public static byte[] getImage(String urlPath) throws IOException {
24          URL url = new URL(urlPath);
25          HttpURLConnection connection = (HttpURLConnection)url.openConnection();
26          connection.setRequestMethod("GET");
27          connection.setReadTimeout(50 * 1000);
28          InputStream inputStream = connection.getInputStream();
29          byte[] data = readInputStream(inputStream);
30          return data;
31      }
32
33      private static byte[] readInputStream(InputStream inputStream) throws
34          IOException {
35          byte[] buffer = new byte[1024];
36          int len = 0;
37          ByteArrayOutputStream bos = new ByteArrayOutputStream();
38          while ((len = inputStream.read(buffer)) != -1) {
39              bos.write(buffer, 0, len);
40          }
41          return bos.toByteArray();
42      }
43  }
```

### 2.3.3 ES6 标准的 JS

```
// Fallback for IE < 9 with es6-shim.  
if (enumerate && !propertyIsEnumerable.call({ 'valueOf': 1 }, 'valueOf')) {  
  baseKeysIn = function(object) {  
    return iteratorToArray(enumerate(object));  
  };  
}
```

### 2.3.4 使用 gulp 作为自动化工具，编译 html 和 css，压缩 js, css

```
gulp.task('patch', [ ], function () { return inc('patch'); })  
gulp.task('minor', [ ], function () { return inc('minor'); })  
gulp.task('major', [ ], function () { return inc('major'); })
```

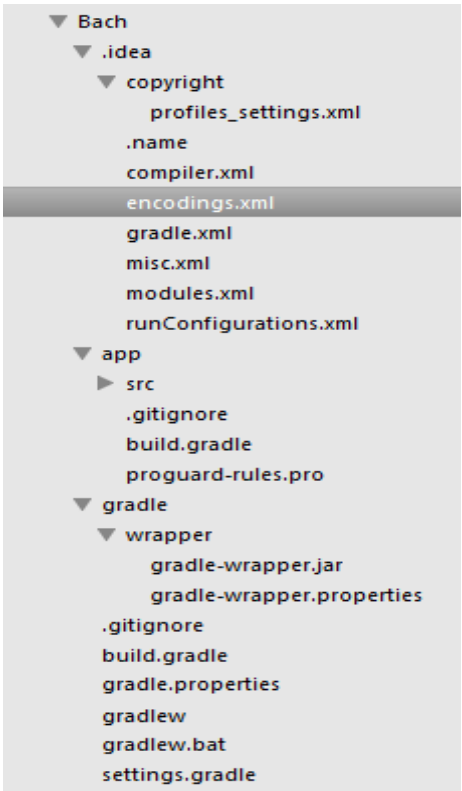
## 2.4 后台技术

1. JDK1.8
2. 分层技术
3. Redis 缓存
4. 使用 JavaBean 组件
5. 数据库使用 MySQL
6. 使用 JDBC 访问数据库
7. SPRING 框架

在对应文件中：



### 2.4.1 分层技术



### 2.4.2 Redis 缓存

	.travis.yml	C:\用户\lenovo-pc\桌面\王青软设\Software-Desi...	类型: YML 文件	修改日期: 2016/7/14 20:48 大小: 69 字节
	.travis.yml	C:\用户\lenovo-pc\桌面\王青软设\Software-Desi...	类型: YML 文件	修改日期: 2016/7/14 20:48 大小: 44 字节
	.travis.yml	C:\用户\lenovo-pc\桌面\王青软设\Software-Desi...	类型: YML 文件	修改日期: 2016/7/14 20:48 大小: 59 字节
	.travis.yml	C:\用户\lenovo-pc\桌面\王青软设\Software-Desi...	类型: YML 文件	修改日期: 2016/7/14 20:48 大小: 68 字节
	.travis.yml	C:\用户\lenovo-pc\桌面\王青软设\Software-Desi...	类型: YML 文件	修改日期: 2016/7/14 20:48 大小: 123 字节
	.travis.yml	C:\用户\lenovo-pc\桌面\王青软设\Software-Desi...	类型: YML 文件	修改日期: 2016/7/14 20:48 大小: 151 字节
	.travis.yml	C:\用户\lenovo-pc\桌面\王青软设\Software-Desi...	类型: YML 文件	修改日期: 2016/7/14 20:48 大小: 48 字节

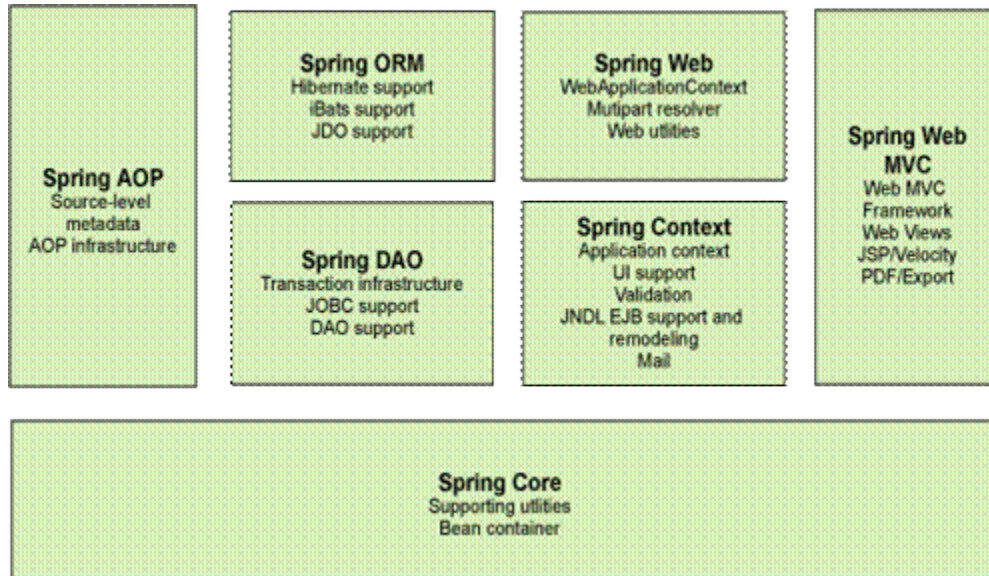
↓

```
.travis.yml
1 language: node_js
2 sudo: false
3 node_js:
4   - '0.10'
5   - '0.12'
6   - '4'
7
```

### 2.4.3 SQL 技术

Software-Design-master\movie-data\node\_modules\sequelize\lib\sql-string.js

### 2.4.4 Spring 框架



### 2.4.5 JDK

```
</component>
<component name="ProjectRootManager" version="2" languageLevel="JDK_1_7" default="true" assert-keyword="true" jdk-15="true"
project-jdk-name="1.8" project-jdk-type="JavaSDK">
  <output url="file://$PROJECT_DIR$/build/classes" />
</component>
```

### 2.4.6 JavaBean 组件

事件:

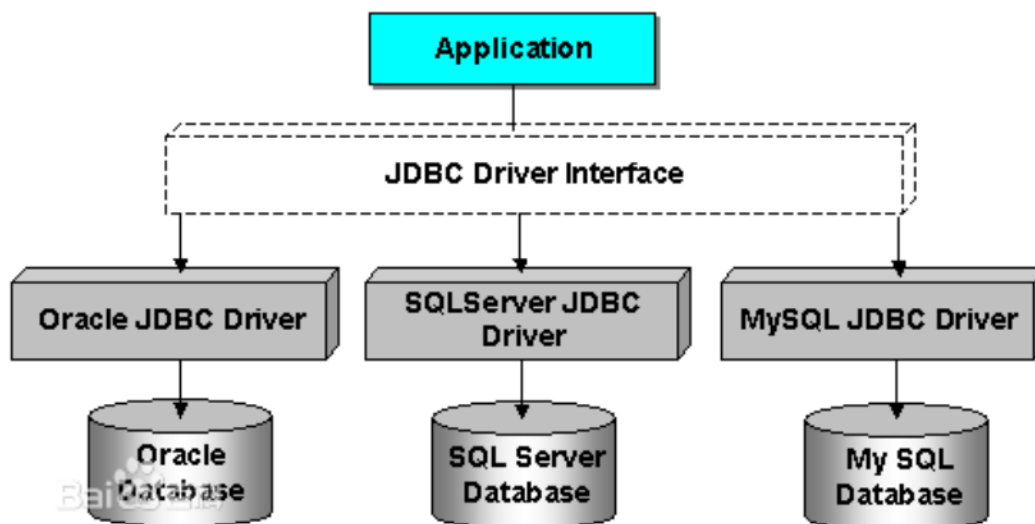
Software-Design-master\Software-Design-master\movie-data\node\_modules\jsdom\lib\jsdom\living\generated\Event

方法:

Software-Design-master\Software-Design-master\movie-data\node\_modules\bluebird\js\release\method

属性:

Software-Design-master\Software-Design-master\movie-data\node\_modules\cssstyle\lib\properties



前后端交互:

为了统一化管理资源,将采用 RESTful API 架构。简单来说就是通过向不同的 url 发送请求来获得不同类型的数据,并且数据交换统一采用 json 格式。

对于加载数据,既然采用 RESTful API 架构,那么前端 javascript 发出 Ajax 请求则变得十分频繁。如果仅仅使用原生 js (哪怕是 JQuery),编写操作 dom 元素的代码工作量也十分大。因此,我们前端拟采用 ReactJS 框架,不仅仅无需直接对 dom 元素进行操作,通过它的 Virtual DOM 机制,可以使前端渲染的速度大大加快(相对于直接操作 dom 元素)。并且能将不同的功能组件化,使一些模块可以得到复用(如 banner, footer 模块),减少代码量。

如果一切资源都是采用 RestfulAPI 来获得,就会涉及到验证用户权限的问题。这里有两个解决方案。第一,用传统的 cookies,但是若使用 ajax 进行传输,这种方式的前端处理要稍显复杂,而且要后台要编写特定的代码以支持 web 端的用户验证;第二,使用 token,这是跟 Android 和 iOS 客户端一致的解决方案,通过后台颁布的 token 来验证用户权限。但是前端如何存储 token 呢? 幸运的是,html5 已经支持 localStorage,是一种比较稳妥的存储策略,可以把 token 存储到 localStorage。