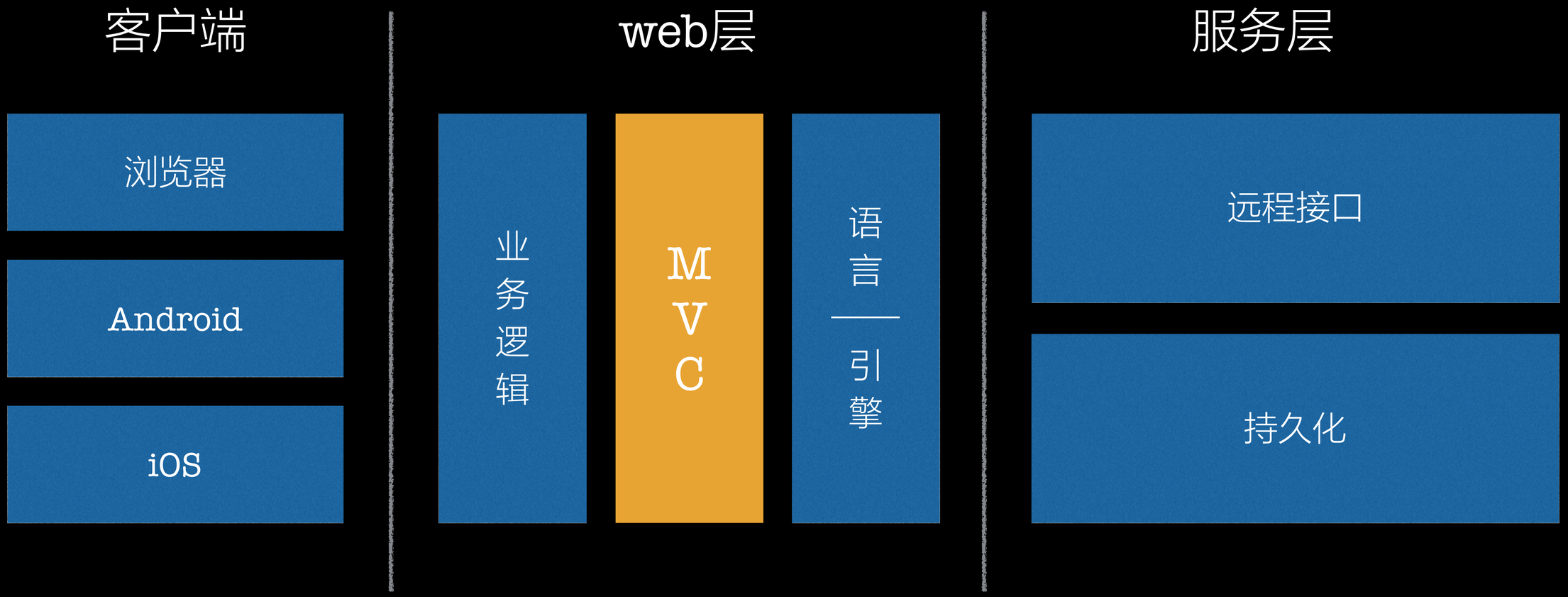
LIGHT

v 0.3.32

另一个全栈Node.js web 框架

v 1.1
无线中心/网站研发部
Allen Guo 郭君龙
@since 2016/11/01

MVC应用场景



一组开发规约 / 项目有序扩张 / 语言无关

无数个轮子

Express

① C&S全栈

② 实时

③ 瞬息万变

再来一个 

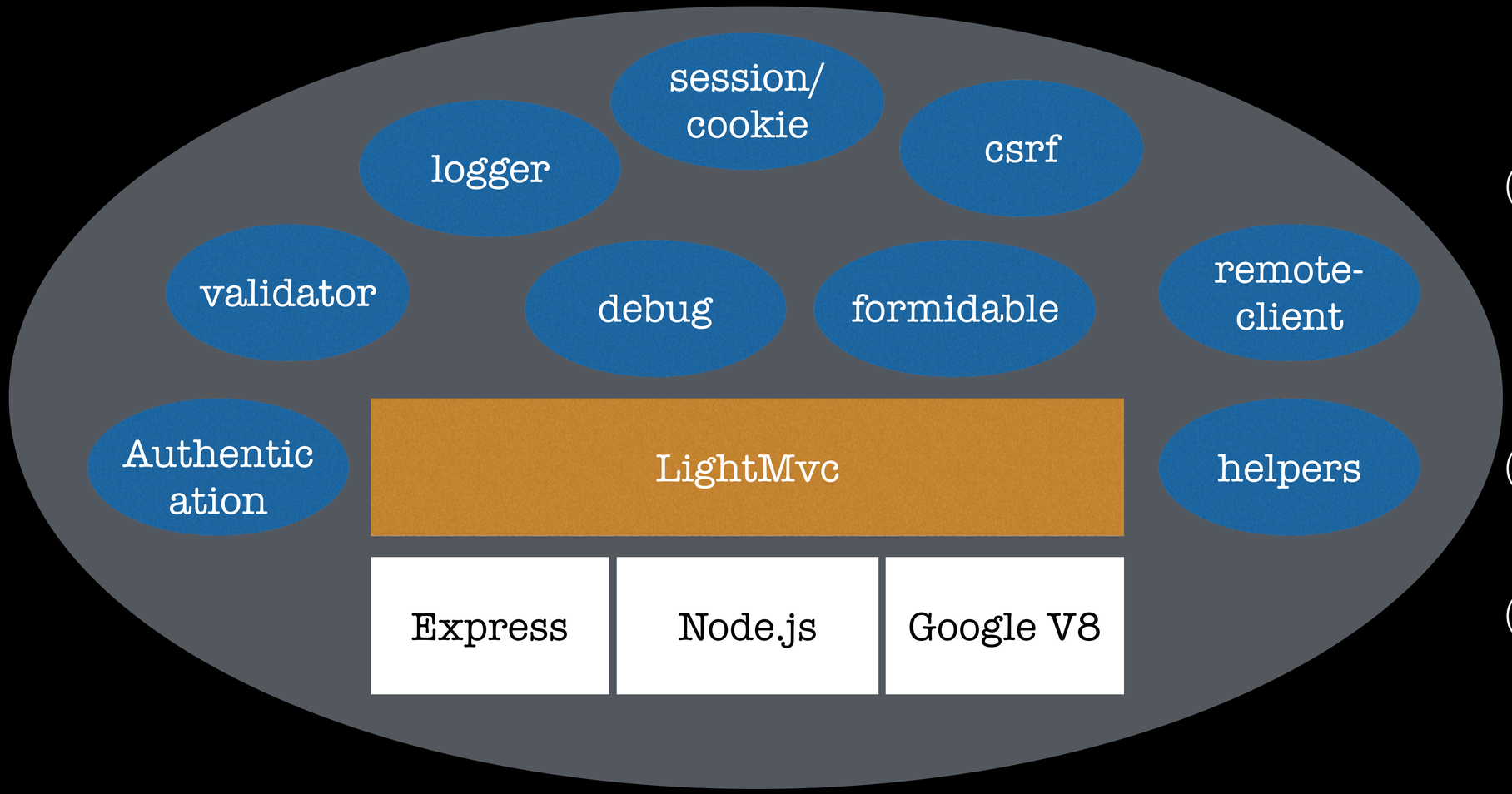
LIGHT

① Js发展大时代，Node.js Web框架也处于极不稳定的时代，每天有无数个“下一代”产生。不妨停下来等一等开源社区。

② 造一个稳定的轮子：基于最稳定的技术，遵循标准。

③ 造一个超轻量级的轮子：最小的成本切换到其他技术栈。

LIGHT生态



① 基于
- Node.js 4.2+
- Express 4.0+

② ES6风格.

③ 自动路由

LIGHT项目目录布局

```
[$PROJ_DIR]
|_ [app]
|   |_ [config]
|   |    |_ config.js // 配置文件
|   |_ [src] //源代码目录
|   |    |_ [controllers] // 控制文件目录
|   |    |      |_ TourController.js
|   |    |_ [models] // 模型文件目录
|   |    |      |_ [tuniu]
|   |    |            |_ TourService.js
|   |    |_ [views] // 模板文件目录
|   |           |_ [tour]
|   |                 |_ detail.ejs
|   |    |_ [bundles]
|   |         |_ [deluxe] // 一个名字叫deluxe的bundle
|   |                 |_ [config]
|   |                 |_ [controllers]
|   |                 |_ [models]
|   |                 |_ [views]
|   |_ [test] // 单元测试文件目录
|_ [node_modules] // 第三方node模块目录
|_ app.js // 项目启动文件
|_ package.json
```

LIGHT路由规则

http://localhost:5000/tour/detail/120

```
// $SRC_DIR/controllers/TourController.js
 'use strict';
 class TourController {
     detail(id) {
         this.res.send('something');
     }
 }
 module.exports = TourController;
```

A. 三级路由&两级目录查找

① 配置文件指定路由

② 代码文件指定路由

③ URL自动路由

B. 项目分割

① autoUrlPrefix

② bundles

Light 如何MVC

C

```
// $SRC_DIR/controllers/TourController.js
'use strict';
class TourController {
  detail(id) {
    let tourSrv = this.ctx.getService('tuniu.TourService');
    let tour = this.tourSrv.getDetail(id);
    this.res.htmlRender(tour);
  }
}
module.exports = TourController;
```

V

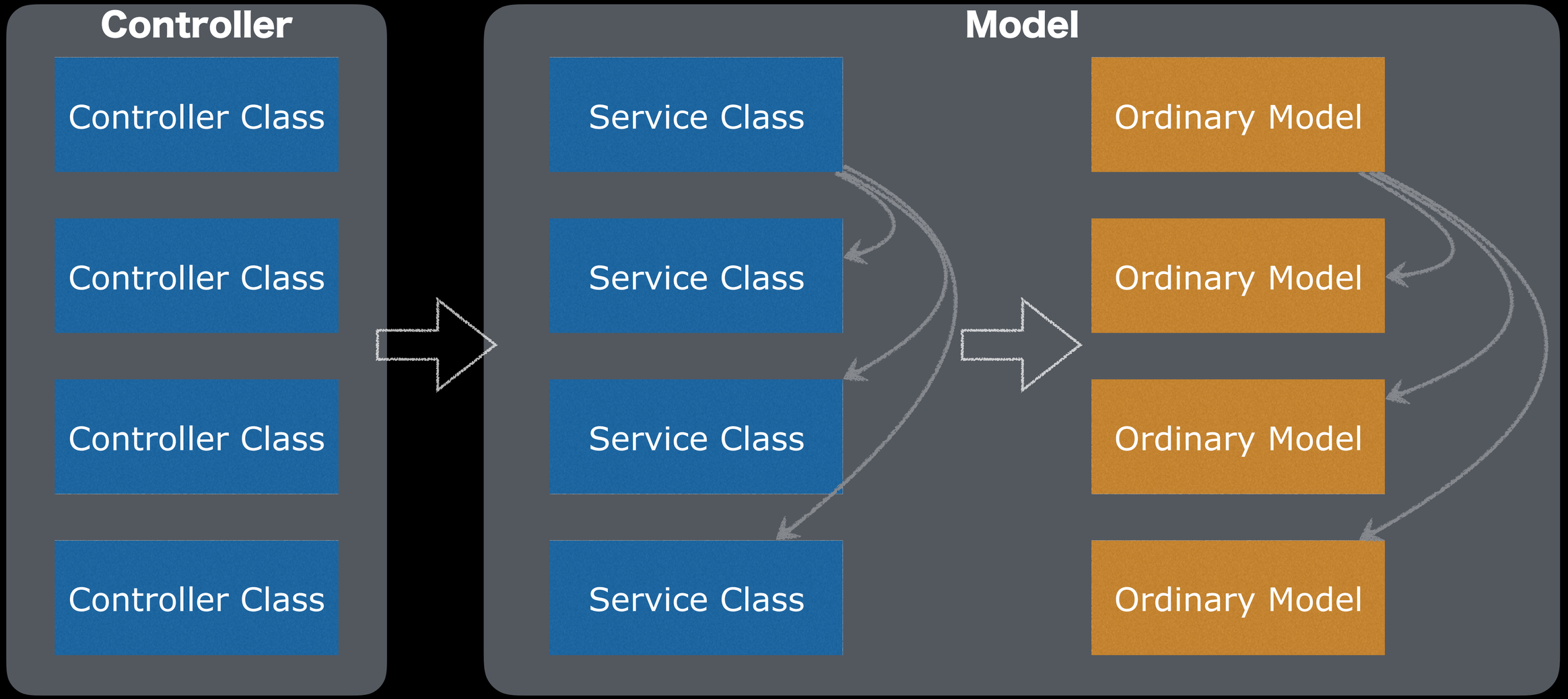
```
// $SRC_DIR/views/tour/detail.ejs
<body><html>
    Title: <span><%=title%></span>
    Price: <span><%=price%></span>
</body></html>
```

M

```
// $SRC_DIR/models/tuniu/TourService.js
'use strict';
class TourService {
    getDetail(id) {
        let thisProd = {};
        tours.forEach(function(item) {
            if (item.id == id) {
                thisProd = item;
            }
        });
    }
}
module.exports = TourService;
```

http://localhost:5000/tour/detail/120

LIGHT设计模式


Controller
Controller Class
Controller Class
Controller Class
Controller Class
Model
Service Class
Service Class
Service Class
Service Class
Ordinary Model
Ordinary Model
Ordinary Model
Ordinary Model

LIGHT设计模式－概念

① **Controller**

1) 是一个ES6 Class，由框架自动实例化。被注入了容器上下文信息和http请求(req,res, next)信息

② **Model：Service和Ordinary(Non-Service)**

1) **Service**：是一个ES6 Class，由代码实例化。被注入了容器上下文信息，和Controller极相似。

2) **Ordinary**：普通的、任意JS Module，没有被框架注入上下文，require直接加载。

```
'use strict';
class TourService {
 // 可选的回调函数，实例化后被调用.
 __construct() {
  this.trainSrv = this.ctx.getService('tuniu.TrainService');
  this.dbSrv = this.ctx.getService('common.DbService');
 }
 detail(id) {
    let tour = getTour(id);
    this.trainSrv.getDetailPromise(id)
    .then(function(train) {
        tour.traffic = train;
        this.res.htmlRender(tour);
    })
    .catch(function(e) {
        return this.next(e);
    });
  }
}
module.exports = TourService;
```

LIGHT设计模式-调用关系

① Controller可以调用Model，反之则不可以

② Service只能被Controller或另一个Service实例调用：

```
this.ctx.getService('tuniu.TourService');
```

③ Ordinary是普通JS Module，使用require或者importModel调用：

```
importModel('tuniu.common.TuniuError');
this.bundle.importModel('tuniu.tour.Product');
```

LIGHT 可配置的Service

BEFORE

```
// service调用文件
this.ctx.getService(
    'tuniu.TourService',
    'deluxe',
    {
        logLevel: 3,
        yourFavorite: 'Dog'
    },
    false
);
```

AFTER

```
// 配置文件
'use strict';
module.exports = {
    srevices: {
        'tuniu-TourService': {
            serviceName: 'tuniu.TourService',
            bundleName: 'deluxe',
            params: {
                logLevel: 3,
                yourFavorite: 'Dog'
            },
            reload: false
        }
    }
};
// Service调用代码文件
this.ctx.getService('tuniu-TourService');
```

- 在service类中this.params存取logLevel、yourFavorite

LIGHT 组织大规模项目

```
[$PRJO_DIR/app]
  |_ [src]
     |_ [bundles]
        |_ [deluxe]
           |_ [config]
           |_ [controllers]
           |     |_ TourController.js
           |_ [models]
           |_ [views]
           |_ [resources]
                 |_ [css]
                 |_ [images]
                 |_ [js]
```

http://localhost:5000/deluxe/tour/detail/120

http://deluxe.tuniu.me/tour/detail/120

bundles

① 独立目录、独立配置

② 零配置、懒加载

③ 域名隔离

④ 发行第三方modules

LIGHT包bundles加载规则

http://localhost:5000/panel

① 去源码目录下寻找

controllers/PanelController.js

② 去源码bundles子目录下寻找

bundles/panel/controllers/IndexController.js

③ 默认加载(require) 如下命名的模块：

- @tuniu/light-bundle-panel
- panel

你可以设置本项目允许的bundle模块前缀

```
// 第三方bundles典型目录结构
[$BUNDLE_DIR]
   |_ [src] //源代码目录
   |    |_ [controllers] // 控制文件目录
   |    |_ [models] // 模型文件目录
   |    |_ [views] // 模板文件目录
   |    |_ [config] //配置目录
   |    |_ [resources] // 资源目录
   |         |_ [js]
   |         |_ [css]
   |         |_ [images]
   |_ [node_modules] // node模块
   |_ index.js // 模块启动文件
   |_ package.json
```

LIGHT配置规范

```
[app]
 |_[config] // 项目配置文件目录
    |_ config.js // 任意配置文件
    |_ config.light_mvc.js // LightMvc配置文件
    |_ [env]
    |     |_ env.js // 可选文件：如果存在该文件，并有env键，框架会自动加载健值同名的子目录下的所有配置文件
    |     |_ [local] // 可选目录：环境配置相关的配置文件.
    |          |_ parameters.yml
    |          |_ config.a.js
    |          |_ config.b.js
    |          |_ [autoloadings] // 可选目录：太多配置直接放在config目录感觉混乱，可以移至该目录
    |_ [autoloadings] // 可选目录：太多配置直接放在config目录感觉混乱，可以移至该目录
    |       |_ config.x.js
    |       |_ config.y.js
    |_ parameters.yml
```

LIGHT配置加载

```
[$PROJ_DIR]
    |_ [app]
        |_ [config]
            |_ parameters.yml
            |_ config.js
            |_ [autoloadings]
            |       |_ config.xx.js
            |_ [env]
                |_ [local]
                    |_ parameters.yml
                    |_ config.js
                    |_ [autoloadings]
                            |_ config.xx.js
```

```
let config ={}
```

```
{
    env: 'local'
}
```

❶

```
{
    env: 'local',
    params: {
    }
}
```

❷

```
{
    env: 'local',
    params: {}
    其他节点
}
```

❸

LIGHT配置加载顺序

① 判断process.NODE_ENV 或 $CONF_DIR/env/env.js的值，设置到config.env节点。

② 加载 $CONF_DIR/*.yml和$CONF_DIR/env/{$config.env}/*.yml(如果存在)，设置到 config.params节点。

③ 加载$CONF_DIR/**/* .js

A. $CONF_DIR/*.js

B. $CONF_DIR/autoloadings/*.js

C. 如果$config.env不为空，继续加载:

A. $CONF_DIR/env/{$config.env}/*.js

B. $CONF_DIR/env/{$config.env}/autoloadings/*.js

LIGHT全栈组件

- **Router & MVC**
  - ① @tuniu/light-mvc #LightMvc核心
  - ② @tuniu/light-common
  - ③ @tuniu/light-render
  - ④ @tuniu/light-config
- **Template**
  - ① ejs #ejs模板语言
  - ② express-partial #Express Layout
  - ③ @tuniu/light-helper
- **Validation/Form**
  - ① express-validator #http参数校验
  - ② validator #一般变量校验
  - ③ csurf #CSRF攻击防范
  - ④ joi #基于schema的对象校验
  - ⑤ formidable #文件上传处理
- **Cookie & Session**
  - ① cookie-parser #Cookie处理
  - ② express-session #Session处理
  - ③ connect-redis
- **Web Api**
  - ① @tuniu/light-common #基础公共类库
  - ② @tuniu/light-render #渲染处理（中间件）
- **Logging**
  - ① log4js
  - ② morgan
- **Bundles**
  - ① @tuniu/light-bundle-panel
- **Development & Debug**
  - ① @tuniu/light-exception #异常处理
- **Authentication/Authorization**
- **i18n & Locale**

LIGHT项目Kickstart

环境： Nodejs4.2+，npm2.15+, cnpm@3.4.1

```
$> cnpm set registry http://10.40.189.154:7001
$> cnpm i -g @tuniu/light-cli
$> light-cli init <Project Name>
$> cd <Project Name>
$> npm run local.start
    (window系统执行 npm run win.start)
```

http://localhost:5000/

LIGHT资源

LightMvc官方参考
http://wiki.tuniu.org/pages/viewpage.action?pageId=64775245

无线Web API规范推荐：
http://wiki.tuniu.org/pages/viewpage.action?pageId=65378359

THE END