蘑菇街-海猪

蘑菇街APP容器化初探

大纲

- 组件化之后遇的新问题
- App简单容器及组件服务化
- 容器化的后续思考

- 多团队，多人协同开发
- 代码之间的依赖错综复杂
- 一次改动范围难以确定

组件化回顾

- 工程拆分
- 总体结构分层，依赖约束
- 约束跨组件直接调用

新的挑战

- QA并行集成
- 组件移植
- 组件的初始化

什么是容器

- 隔离组件并提供组件间通信的能力
- 运行、卸载等管理组件的能力
- 足够且合理的生命周期回调
- 提供明确的接入规则

探索与实践

组件入口

组件服务化

依赖注入

xml配置

容器的雏形-MSH

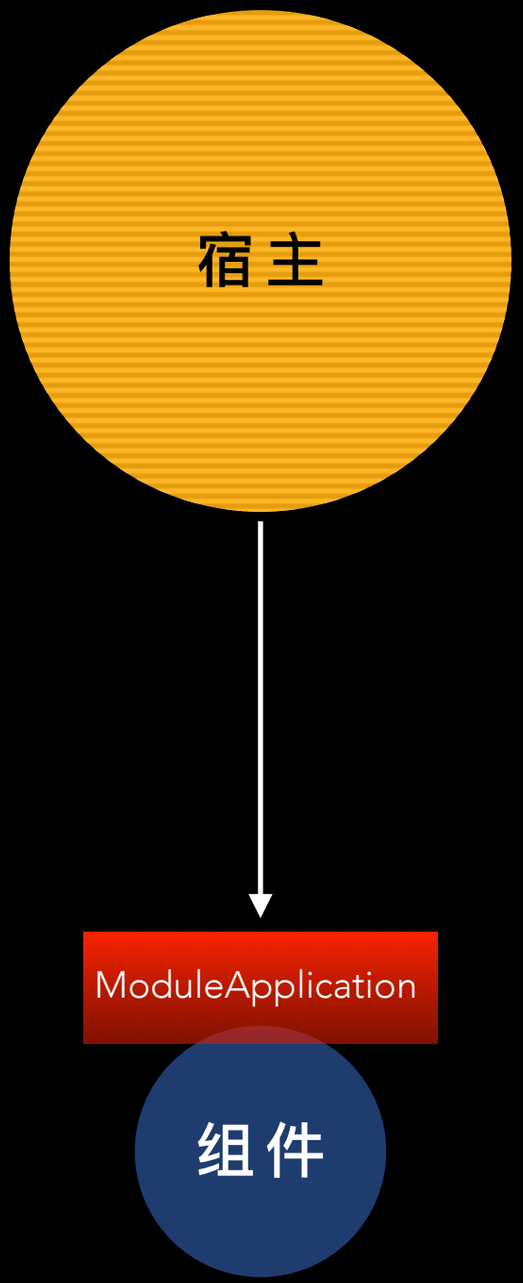
- 入口-ModuleApplication
- 服务化-ServiceHub&MGRouter
- 接入规则-ModuleConfig.xml

MSH配置

```
<module name="ModuleLogin" package="com.mogujie.login">
    <application>
        <name>.LoginModuleApplication</name>
    </application>
    <runtime>
        <alias name="config">com.mogujie.login.LoginConfig</alias>
    </runtime>
</module>
```

组件入口-ModuleApplication

- 初始化入口
- 组件生命周期回调
- 组件服务注册时机
- 外部依赖注入



```java
public class LoginModuleApplication extends ModuleApplication {
    @MSHRuntime(
        alias = "config"
    )
    private ILoginConfig mConfig;

    public LoginModuleApplication() {
    }

    public void onModuleWillCreate() {
        super.onModuleWillCreate();
        MGEvent.getBus().register(this);
        LoginConfigHelper.getInstance().setDelegate(this.mConfig);
    }

    protected void obtainService(Map<String, ModuleService> container) {
        super.obtainService(container);
        container.put("ILoginService", new LoginService());
    }

    protected void obtainUrlProtocol(Map<Uri, RouterCallBack> container) {
        super.obtainUrlProtocol(container);
        RouterCallBack callback = new RouterCallBack() {
            public RouterGo route(RouterGo routerGo) {
                boolean templateEnabled = ((Boolean)(new
HoustonStub("userConfig", "loginTemplateSwitch", Boolean.class,
Boolean.valueOf(false))).getEntity()).booleanValue();
                if(templateEnabled) {
                    Intent intent = new Intent(routerGo.getContext(),
MGLoginAct.class);
```

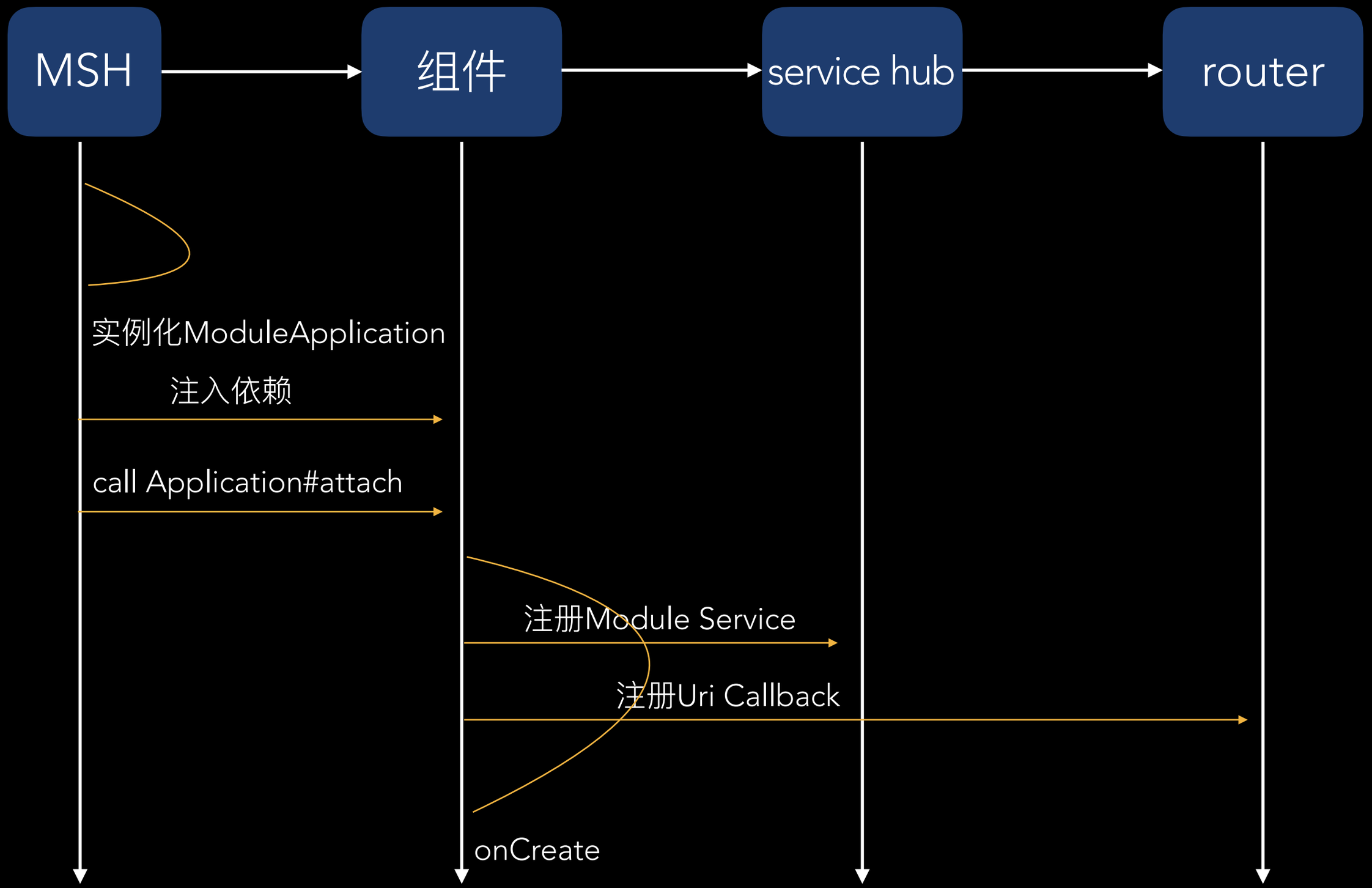
```
public class LoginModuleApplication extends ModuleApplication {
    @MSHRuntime(
        alias = "config"
    )
    private ILoginConfig mConfig;

    public LoginModuleApplication() {
    }

    public void onModuleWillCreate() {
        super.onModuleWillCreate();
        MGEvent.getBus().register(this);
        LoginConfigHelper.getInstance().setDelegate(this.mConfig);
    }

    protected void obtainService(Map<String, ModuleService> container) {
        super.obtainService(container);
        container.put("ILoginService", new LoginService());
    }

    protected void obtainUrlProtocol(Map<Uri, RouterCallBack> container) {
        super.obtainUrlProtocol(container);
        RouterCallBack callback = new RouterCallBack() {
            public RouterGo route(RouterGo routerGo) {
                boolean templateEnabled = ((Boolean)(new
HoustonStub("userConfig", "loginTemplateSwitch", Boolean.class,
Boolean.valueOf(false))).getEntity()).booleanValue();
                if(templateEnabled) {
                    Intent intent = new Intent(routerGo.getContext(),
MGLoginAct.class);
```

```
    public LoginModuleApplication() {
    }

    public void onModuleWillCreate() {
        super.onModuleWillCreate();
        MGEvent.getBus().register(this);
        LoginConfigHelper.getInstance().setDelegate(this.mConfig);
    }

    protected void obtainService(Map<String, ModuleService> container) {
        super.obtainService(container);
        container.put("ILoginService", new LoginService());
    }

    protected void obtainUrlProtocol(Map<Uri, RouterCallBack> container) {
        super.obtainUrlProtocol(container);
        RouterCallBack callback = new RouterCallBack() {
            public RouterGo route(RouterGo routerGo) {
                boolean templateEnabled = ((Boolean)(new
HoustonStub("userConfig", "loginTemplateSwitch", Boolean.class,
Boolean.valueOf(false))).getEntity()).booleanValue();
                if(templateEnabled) {
                    Intent intent = new Intent(routerGo.getContext(),
MGLoginAct.class);
                    intent.setData(routerGo.getUri());
                    routerGo.getContext().startActivity(intent);
                    routerGo = null;
                }

                return routerGo;
            }
```

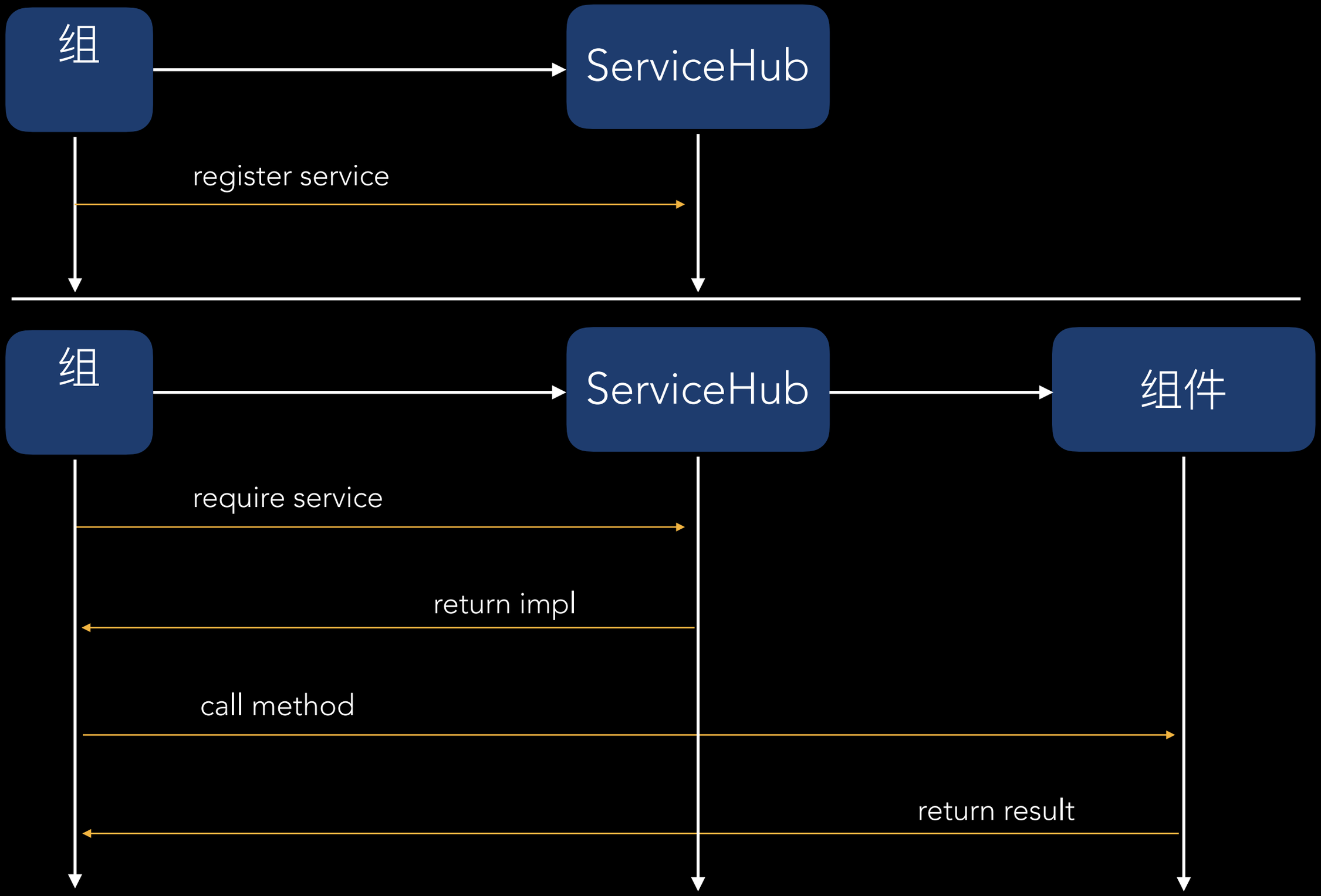
```
    protected void obtainService(Map<String, ModuleService> container) {
        super.obtainService(container);
        container.put("ILoginService", new LoginService());
    }

    protected void obtainUrlProtocol(Map<Uri, RouterCallBack> container) {
        super.obtainUrlProtocol(container);
        RouterCallBack callback = new RouterCallBack() {
            public RouterGo route(RouterGo routerGo) {
                boolean templateEnabled = ((Boolean)(new
HoustonStub("userConfig", "loginTemplateSwitch", Boolean.class,
Boolean.valueOf(false))).getEntity()).booleanValue();
                if(templateEnabled) {
                    Intent intent = new Intent(routerGo.getContext(),
MGLoginAct.class);
                    intent.setData(routerGo.getUri());
                    routerGo.getContext().startActivity(intent);
                    routerGo = null;
                }

                return routerGo;
            }
        };
        container.put(Uri.parse(this.getString(string.mgj_login_app_schema)
+ "://login"), callback);
    }

    @Subscribe
    public void onEvent(Intent intent) {
        if(intent != null) {
            if("event_register_all_complete".equals(intent.getAction())) {
```

```
HoustonStub("userConfig", "loginTemplateSwitch", Boolean.class,
Boolean.valueOf(false))).getEntity()).booleanValue();
                if(templateEnabled) {
                    Intent intent = new Intent(routerGo.getContext(),
MGLoginAct.class);
                    intent.setData(routerGo.getUri());
                    routerGo.getContext().startActivity(intent);
                    routerGo = null;
                }

                return routerGo;
            }
        };
        container.put(Uri.parse(this.getString(string.mgj_login_app_schema)
+ "://login"), callback);
    }

    @Subscribe
    public void onEvent(Intent intent) {
        if(intent != null) {
            if("event_register_all_complete".equals(intent.getAction())) {
                String loginSource = UnpackUtils.getValue(intent,
"login_source", "");
                CheckHelper.checkIsNeedCoupon(loginSource);
            }

        }
    }
}
```

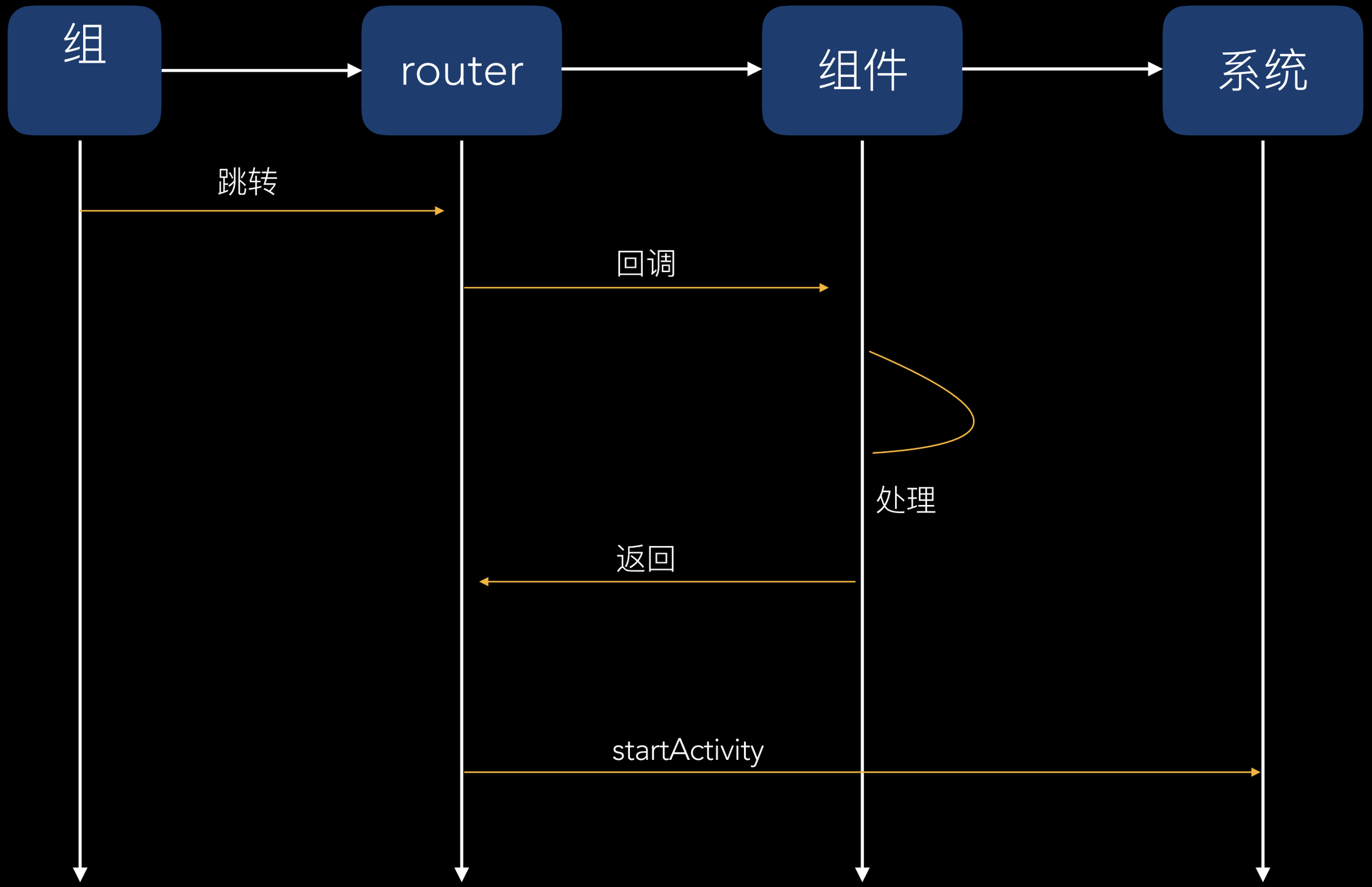
MSH初始化组件过程


MSH
组件
service hub
router
实例化ModuleApplication
注入依赖
call Application#attach
注册Module Service
注册Uri Callback
onCreate

服务化之ServiceHub



服务化之MGRouter



举个栗子

容器化的后续思考

- 组件延迟初始化(按需加载)
- 插件化框架的接入
- 组件与插件的随时转换
- 插件可卸载

感谢观看