

Web开发新时代

Michael Chen, 2005年8月6日

Email: mehiland@gmail.com

目录

- 引言
- Ajax与Amowa
- Web组件
- 轻量级框架

新技术的通用目的

- 使我们的客户满意
 - 短的开发工期
 - 频繁而及时的交付
- 使我们的用户满意
 - 更好的用户体验
- 使我们自己满意
 - 更少的重复代码
 - 更少的代码耦合
 - 学习新的知识，不浪费学习成本
 - 更容易测试
 - ...

如何满足需求(技术层面)

- 使我们的客户满意：
 - 使用更快的技术
 - 使用容易测试的技术
 - 尝试敏捷实践
- 使我们的用户满意：
 - **AJAX(AMOWA)**相关技术，使得用户体验更好
- 使我们自己满意：
 - 使用**Web**组件，加快开发速度，避免重复劳动
 - 使用轻量级容器及其基础设施，管理依赖，提高开发速度

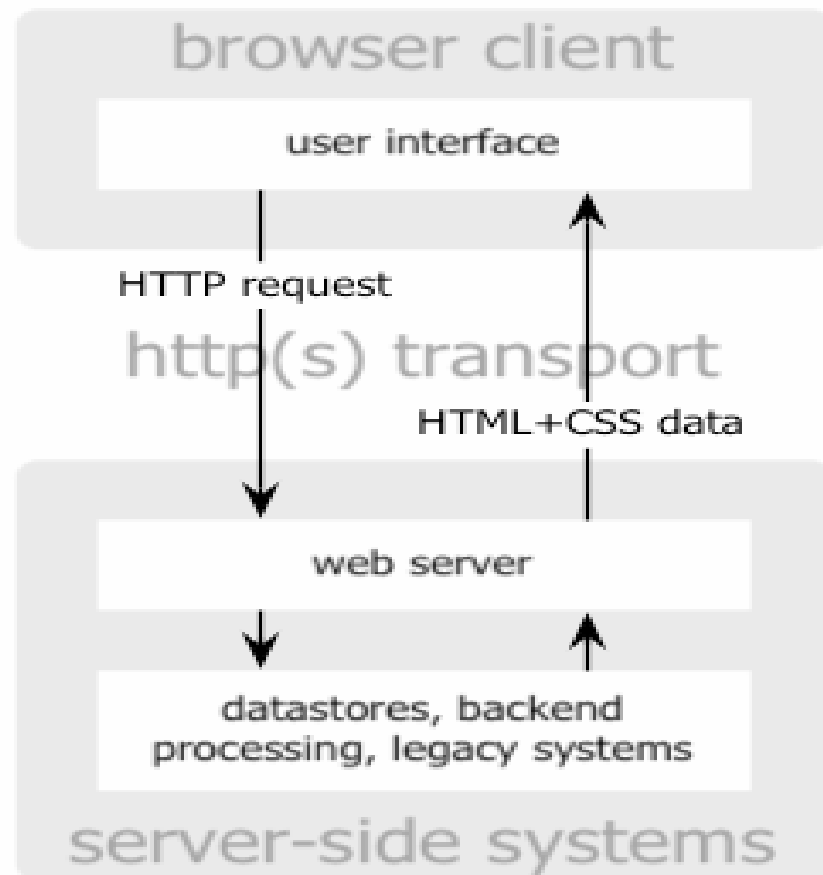
AJAX, Amowa

Web开发新思路。

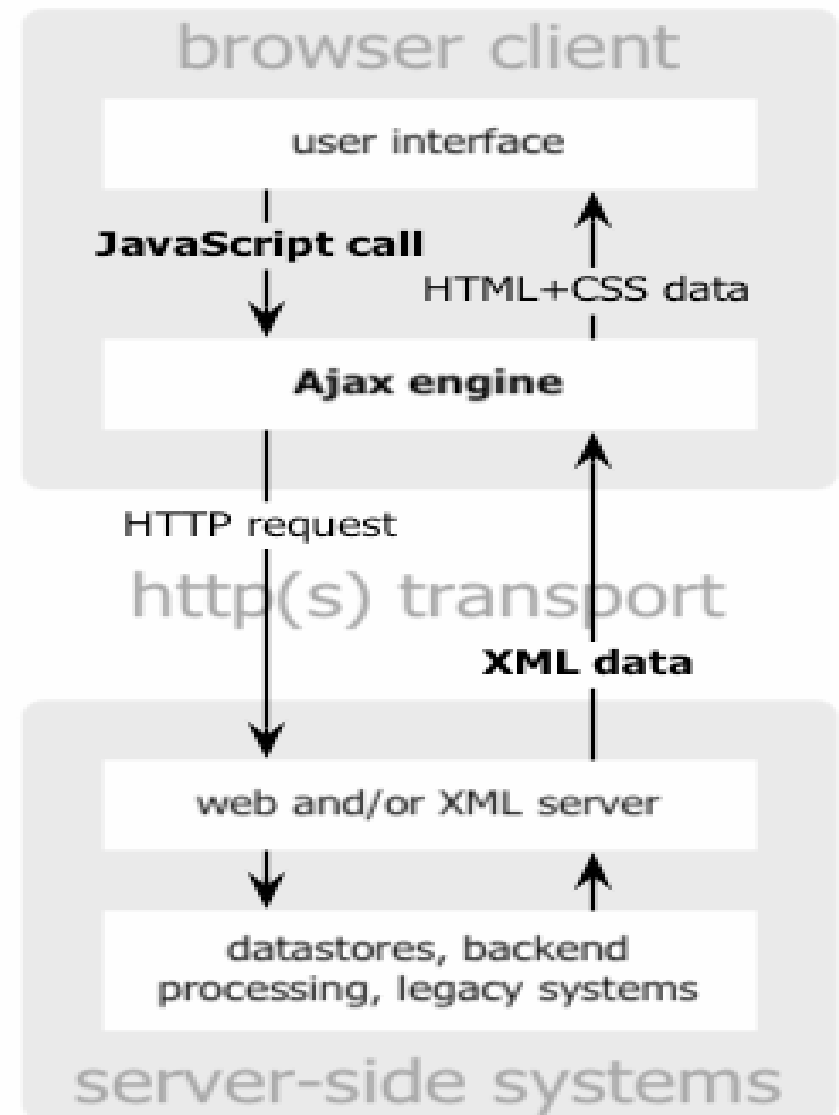
A way we improve the user experience

AJAX

- Asynchronous JavaScript + XML
 - 异步 + JavaScript + XML
- 05年3月，由Jesse James Garrett提出
- 通过Xmlhttp，实现异步调用
- XML: 作为传输的载体，为了统一消息的处理方式。但不是所有的AJAX引擎都实现
- J2EE Blueprints中加入了这项解决方案

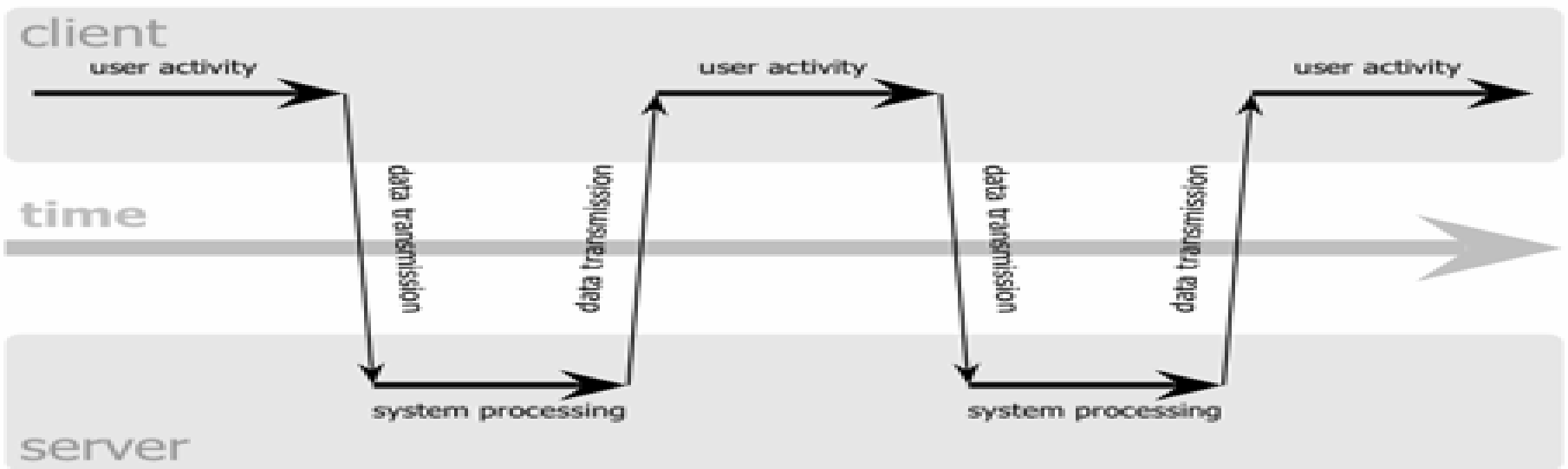


classic
web application model

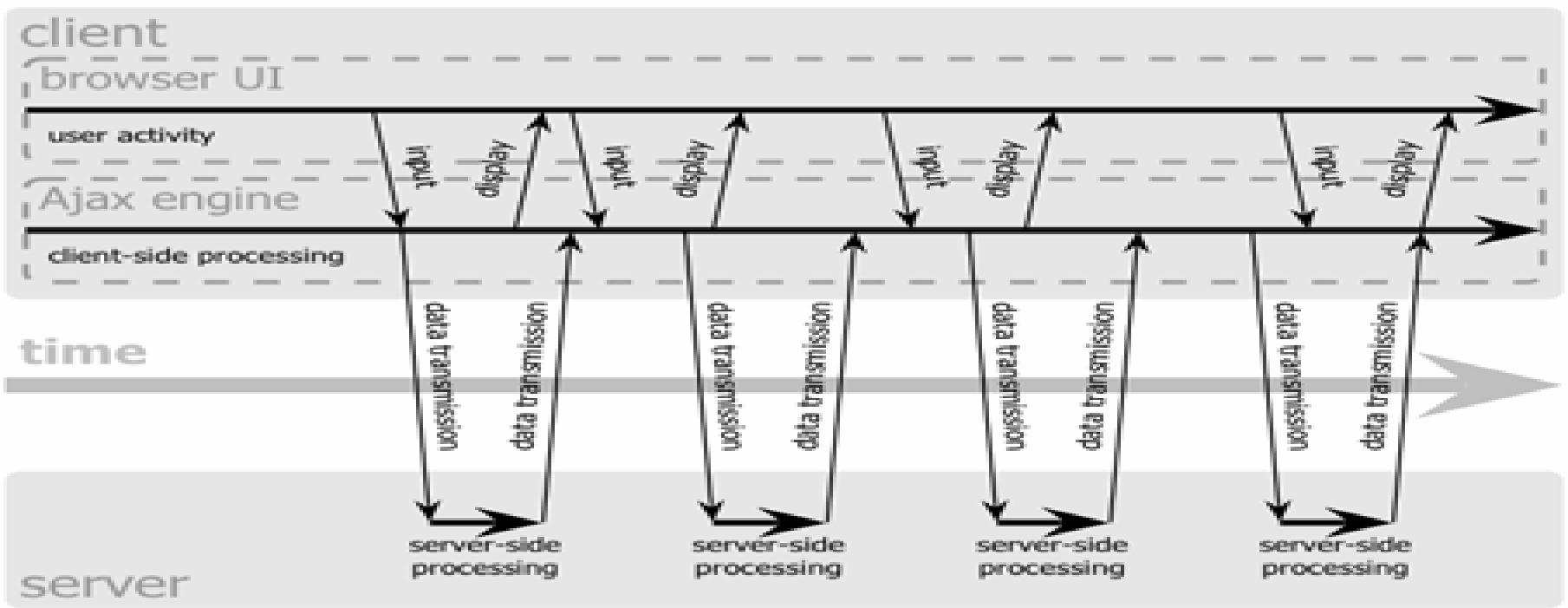


Ajax
web application model

classic web application model (synchronous)



Ajax web application model (asynchronous)



AJAX样例：获取服务器时间



```
// some servlet
public void doGet(request, response){
    String xml = "<responseXml>";
    String date = new Date().toString();
    xml += date+"</responseXml>";
    response.setContentType("text/xml;charset=utf-8");
    response.getWriter().println(xml);
}
```

```
// client javascript
var url = "someServlet";
var xmlhttp = getXmlHttpRequest();
xmlhttp.onreadystatechange = processRequest;
xmlhttp.open("GET", url, true);
xmlhttp.send(null);
function processRequest() {
    if (xmlhttp.readyState == 4) {
        if (xmlhttp.status == 200) {
            document.getElementById("timeDiv").innerHTML =
xmlhttp.responseText;
        }
    }
}
```

In IE:
new ActiveXObject("MSXML2.XmlHttp")

In Mozilla/Firefox:
new XMLHttpRequest();

AJAX样例 – 评价

- 效果
 - 避免页面刷新，用户体验好
 - 数据局部加载，数据流量小，加载速度快
 - 可以控制加载数据时的动作
- 不足
 - 编码琐碎，太细节
 - 界面的工作太多

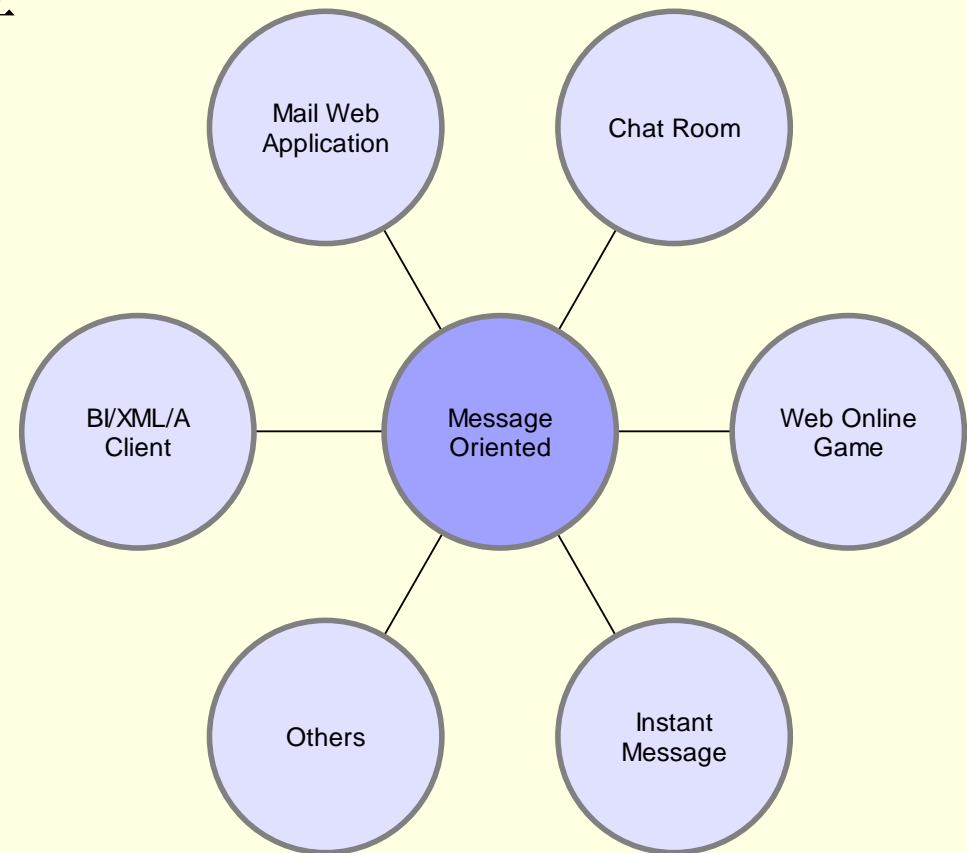
当前可用的AJAX引擎

- DWR(Direct Web Remoting)
- Buffalo
- JSON-RPC
- Prototype
- Rico
- 更详细全面的列表:

<http://www.ajaxpatterns.org/AJAXFrameworks>

AMOWA: 面向异步消息的Web应用

- AJAX概念限制了Web应用的范围
- Amowa: 从广度和深度上, 对Web开发提出另外层面的启发
 - 首先是一个Web应用
 - 系统中具备消息原型, 无论是显式还是隐式
 - 异步



AJAX/Amowa引擎应当具备的功能

- 客户端与服务器端的对接
- 异步调用的封装，支持自定义动作
 - 当请求进行时，请求完成时，发生错误时
- 对象的自动序列化与反序列化
- 界面控件与返回值的绑定处理

Buffalo的实现

- 简单。
- 采用Burlap协议
- 基于回调的编程模型
 - 简单。无需关心实现细节。
- JS/Java任意数据类型的双向序列化/反序列化，全面支持
- 界面绑定模块(未成熟)
 - 属性侵入，而非节点侵入，不影响设计
 - 支持绝大多数原生HTML界面控件
- 一个properties即可完成普通的应用配置
- 与Spring集成，直接使用Spring的各种便利设施以及bean的创建方式。
- 非常容易的为现有应用添加Amowa/Ajax特性。

Buffalo编码样例

```
class UserService extends BuffaloService {
    public boolean login(String username, String password) {
        if (username.equals(password))
        { getRequest().getSession().setAttribute("login_username", username);
            return true; }
        else { return false; }
    }
    public List userList() { return userDao.getAllUsers(); }
}
```

```
#buffalo.properties
userService=my.service.UserService
```

```
var buffalo = new Buffalo("/BUFFALO", true);
buffalo.remoteCall("userService.login",[username, password], function(reply) {
    if (reply.getResult()) { alert("登录成功"); }
    else { alert("登录失败!"); }
})
```

```
buffalo.bindReply("userService.userList",[], "userTable");
```

应用场景

- 全面应用
 - 是否关心URL
 - 论坛(forum/viewthread.ext?page=abc)
 - 邮件(mail/?)
- 组件应用
 - 关注同一页面内的数据交互
 - Tree
 - Grid

AJAX小结

- 优点
 - 减轻服务器的负担
 - 无刷新更新页面，减少用户实际和心理等待时间
 - 更好的用户体验
 - 可以把以前的一些服务器负担的工作转移到客户端
 - 基于标准化的并被广泛支持和技术
 - Ajax使WEB中的界面与应用分离
 - 对于用户和ISP来说双盈
- 缺点：
 - 一些手持设备（如手机、PDA等）现在还不能很好的支持
 - 用JavaScript作为Ajax引擎，兼容性问题以及调试问题
 - Ajax的无刷新重载，变化不明显
 - 对流媒体的支持有限

Web组件

Which we accelerate our development.

Web组件

- 定义：可复用，界面相关。
- 目的：解决代码无意义重复问题，Web开发中，界面成为工作量瓶颈。
- 常见场景：
 - Tree
 - Grid
 - Graph
 - 更大颗粒、具备业务含义的组件。

出现历史

- C/S时代丰富的组件(Text, Grid, ComboBox...)
- HTML Form有限的组件
- HTC, JavaScript
- JSP Tags
- JSF
- 各种Web框架自己的组件机制: Tapestry, Echo1/2, ...

分类

- 服务器端实现，客户端使用：**JSP Tag**，**JSF**(节点侵入)，特点：开发难度大，使用难度小，设计期不可见，大部分都需要与服务器进行页面级别交互。
- 服务器端实现特例：**Tapestry**，完全面向组件的**Web**框架，属性侵入，设计期可见
- 客户端实现：大量的**JS**组件(**ActiveGrid**，**WebFX Tree...**)，设计的较差的需要写**JS**，较好的通过**css**样式侵入，设计期可见。

组件思路

- **Web**组件很重要，重视**Web**组件的积累将会为后续的开发带来若干便利
- 尝试采用某一种具备明确组件特征并且成熟的**Web**框架，例如**Tapestry**
- 设计期的支持需要进行深入考虑。**HTML**的兼容并不是时刻都需要满足。

AJAX/Amowa时代的Web组件

- 组件在客户端实现(JS/AS/XUL), 状态在客户端管理, 数据交互通过Ajax或者其他远程调用方式来进行, 代表: xLoadTree, ActiveGrid
- 组件在服务器端实现(JSP Tag), 但数据交互通过Ajax或其他远程调用方式, 代表:
ajax+jsf(in j2ee bpcategory).
- 交互的方式: 直接访问应用服务层, 或者其代理层。缩减MVC部分的工作。

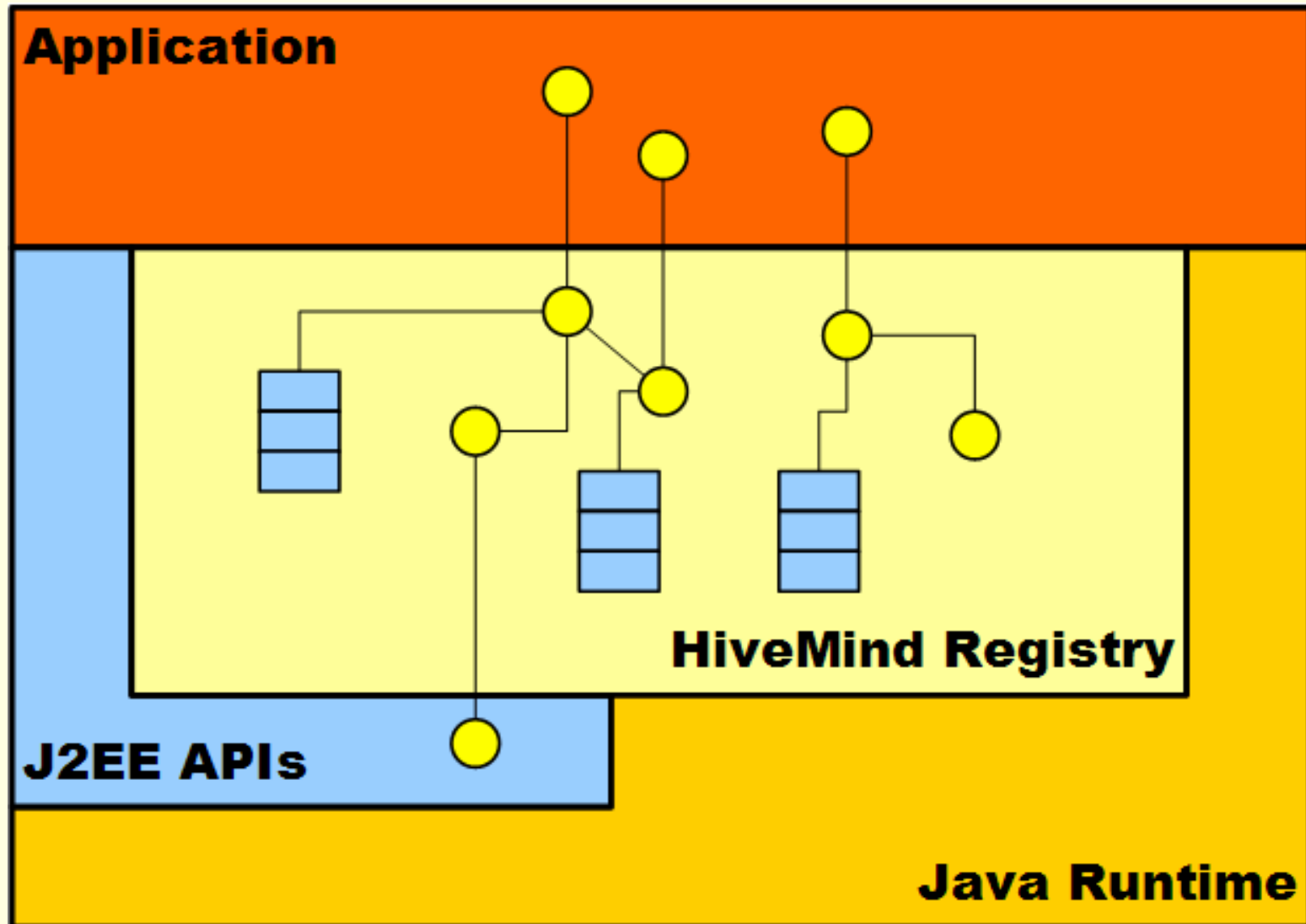
轻量级框架

Lightweight Frameworks

开源产品

- Pico/Nano
- Spring
- HiveMind

基本设计图



何谓“轻量”？

- 对外界依赖最少，甚至没有
- 管理内部依赖
- 容器内容器

核心设计：IoC

■ 控制反转，或者称依赖注入

```
myDao = new MyDAO();  
ds = new BasicDataSource(...);  
myDao.setDataSource(ds);
```

```
myDao = applicationContext.getBean("myDao");
```

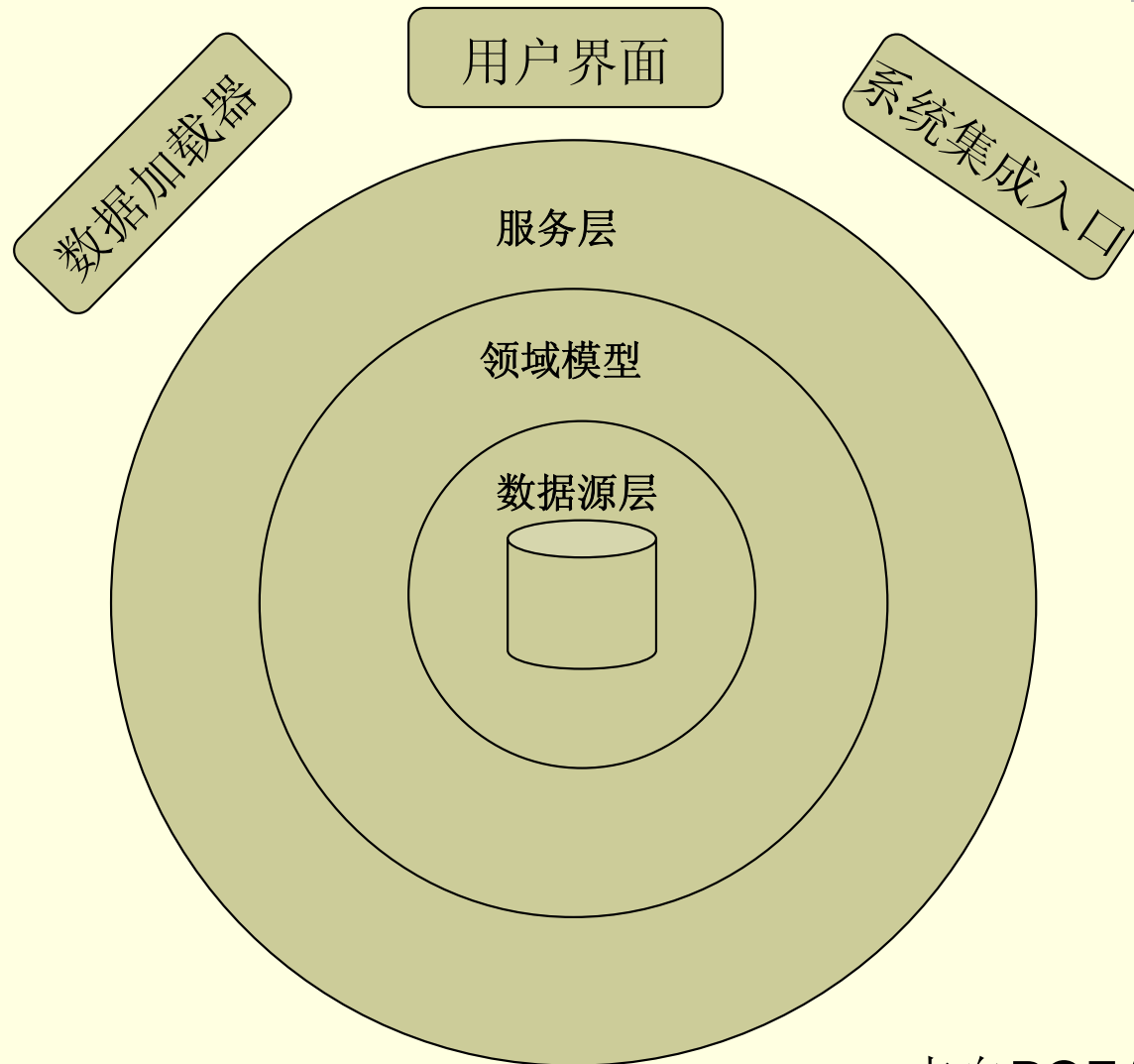
```
<bean id="myDao" class="dao.MyDAO">  
  <property name="dataSource"><ref bean="datasource" /></property>  
</bean>
```

```
<bean id="datasource" class="org.apache.commons.dbcp.BasicDataSource">  
  ....  
</bean>
```

小结

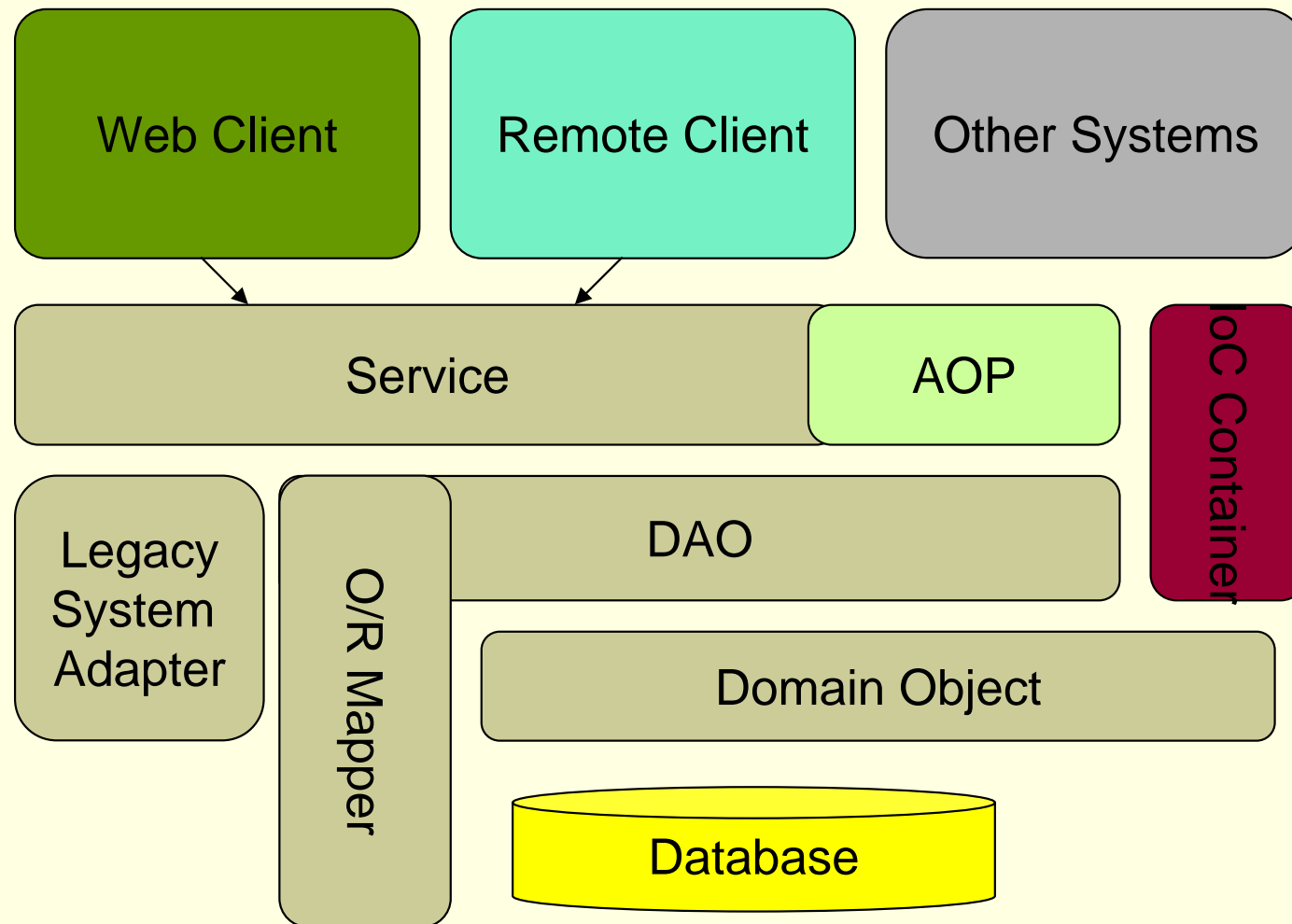
- 面向接口编程
- 使用某一种IoC框架来隔离变化，杜绝过度工厂设计
- **Spring**: 为应用开发做了更多: AOP, ORM, JDBC, DAO, MVC, Web, EJB. 对每一部分都进行了抽象和简化。

架构的变化

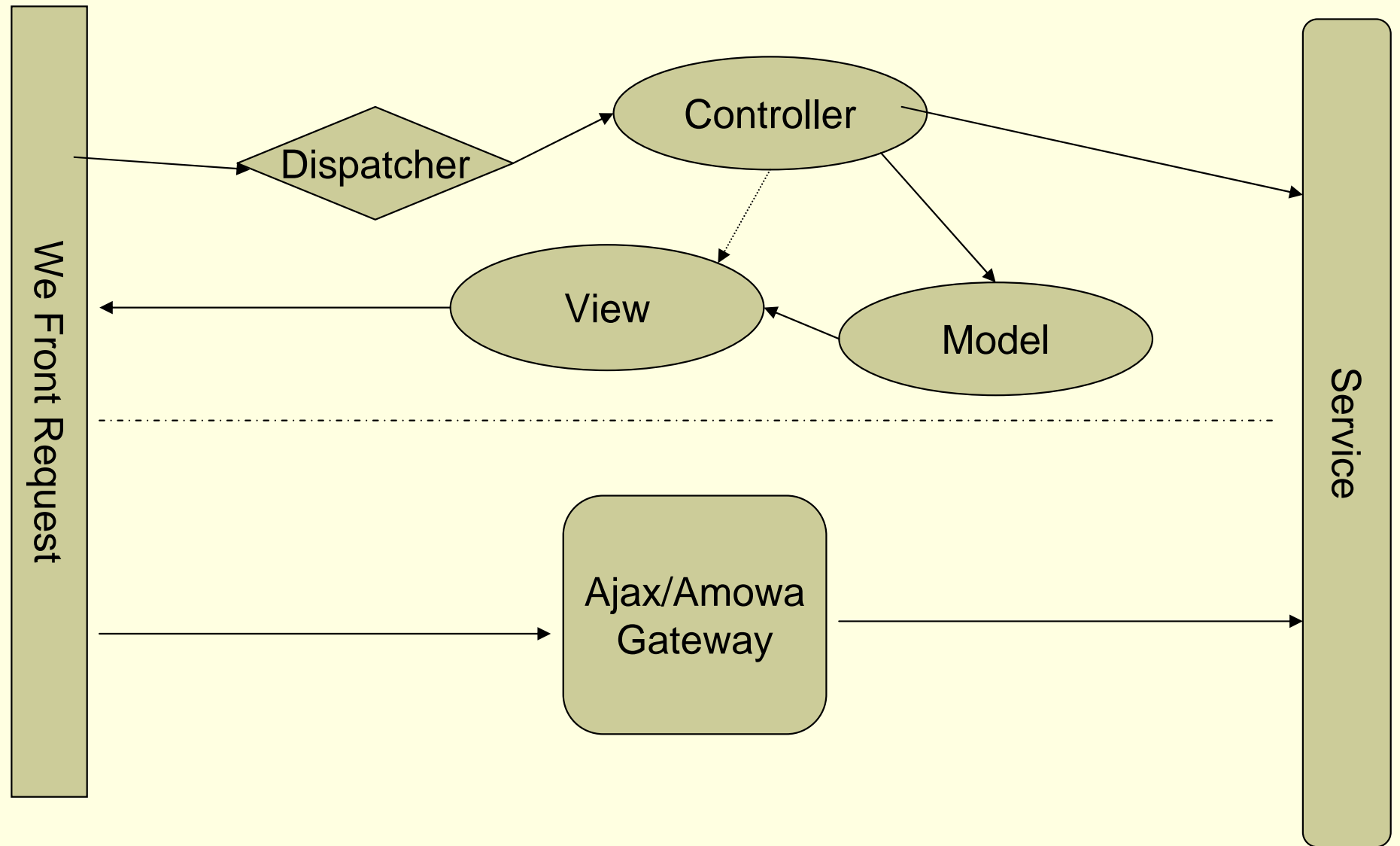


来自POEAA

更详细的架构



Web层设计变化



总结

- 学习Ajax/Amowa, 创建更具竞争力的Web应用
- 注重Web组件的积累, 加快开发
- 采用某一种具备丰富应用背景的IoC容器, 管理依赖, 并加快开发。
- Ajax/Amowa给应用架构在表示层上带来了一些变化。

Thanks

- Reference

- <http://www.amowa.net>
- <http://www.amowa.net/buffalo>

