

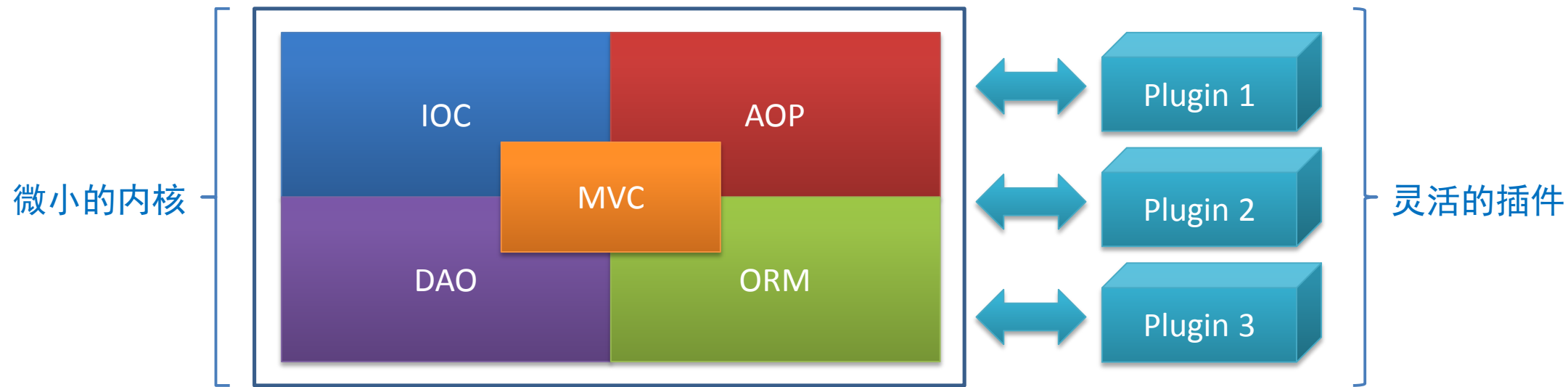
让你的开发变得如此 Smart

轻量级 Java Web 框架

项目简介

- ✓ 用于快速开发中小规模的企业应用 或 网站应用
- ✓ 它是一款轻量级 Java Web 框架
 - 不到 3000 行代码实现 IOC、AOP、ORM、DAO、MVC 等功能
 - 基于 Servlet 3.0 规范
 - 使用 Java 注解取代 XML 配置
- ✓ 使应用充分做到 “前后端分离”
 - 前端可使用 HTML 或 JSP 作为视图模板
 - 后端可发布 REST 服务
 - 通过 Ajax 获取后端数据并进行界面渲染

系统架构



内核与插件

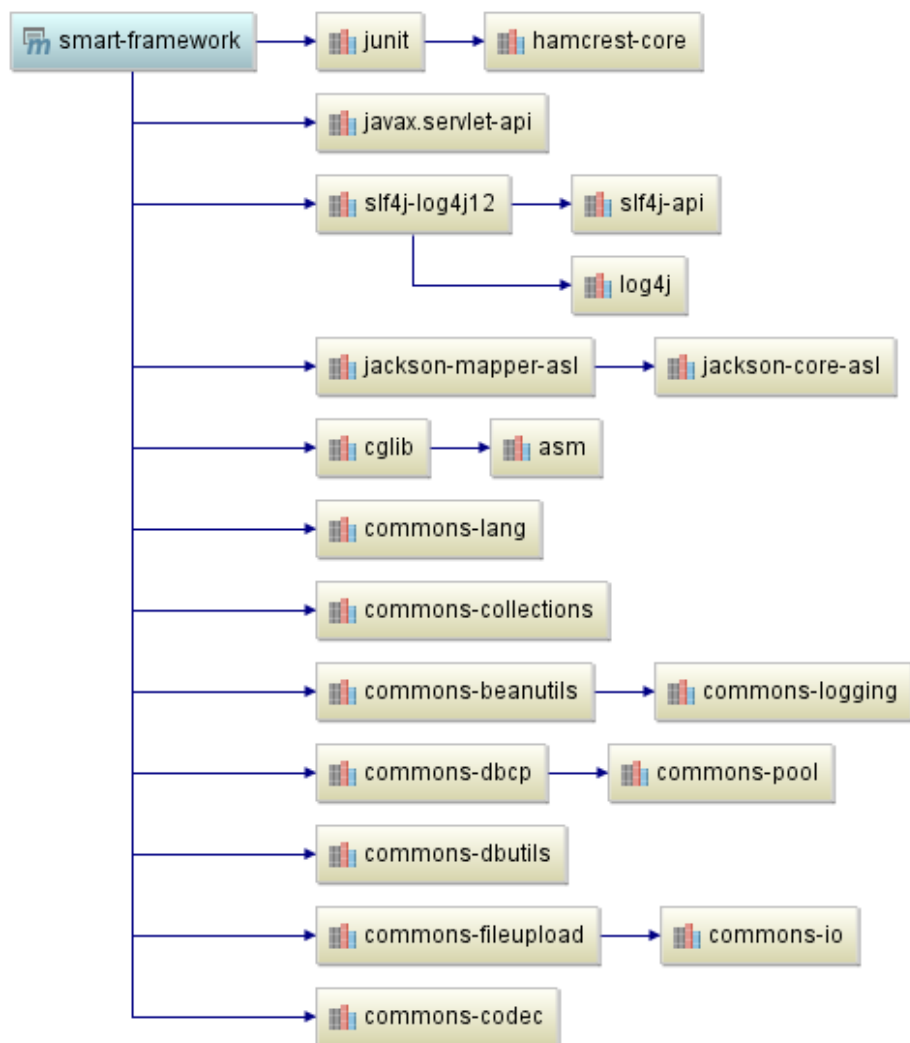
内核

- ✓ MVC
 - 基于 Servlet 3.0 规范
- ✓ IOC
 - 轻量级 IOC 容器
- ✓ AOP
 - 轻量级 AOP 框架
- ✓ ORM
 - 基于 JDBC 规范
- ✓ DAO
 - 统一的数据访问 API

插件

- ✓ Cache
 - 基于注解或使用 Cache API
- ✓ Webservice
 - 发布与调用 SOAP 服务或 REST 服务
- ✓ Mail
 - 邮件发送与收取
- ✓ I18N
 - 国际化多语言包
- ✓ Job
 - 基于 Quartz 的 cron 表达式
- ✓ Hessian
 - 通过 HTTP 传输二进制数据

项目依赖



- 使用 [JUnit](#) 进行单元测试
- 基于 [Servlet 3.0](#) 规范，可部署在 [Tomcat 7.0](#) 上
- 使用 [SLF4J](#) 作为日志接口，使用 [Log4J](#) 作为日志实现
- 使用 [Jackson](#) 作为 JSON 序列化与反序列化
- 使用 [CGLib](#) 作为动态代理工具
- 使用 [Apache Commons](#) 工具包
- 使用 [DBCP](#) 作为数据库连接池
- 使用 [DbUtils](#) 封装 JDBC 操作
- 使用 [FileUpload](#) 实现文件上传

准备开发工具

✓ Java 虚拟机

- [JDK 1.6](#)

✓ 集成开发环境：

- [Eclipse](#)
- [IntelliJ IDEA](#)

✓ 项目构建

- [Maven](#)

✓ Web 服务器

- [Tomcat 7.0](#)
- [Apache](#)（可选）

✓ 数据库

- [MySQL](#)（服务器）
- [Navicat](#)（客户端）

✓ 源码版本控制

- [Git](#)（服务器）
- [SourceTree](#)（客户端）

搭建开发环境

✓ 搭建 Maven 开发环境

- 修改 Maven 全局配置文件 setting.xml
- 参考: <http://my.oschina.net/huangyong/blog/195559>

✓ 搭建源码开发环境

- git clone <http://git.oschina.net/huangyong/smart-framework.git>
- mvn install

前后端分离

前端	后端
制作界面原型（HTML 或 JSP）	设计数据模型
定义 URL 规范 设计页面跳转流程	定义数据库表结构 编写 Entity 类（数据模型） 定义 Service 接口（业务接口）
编写 Action 类（调用 Service 接口）	实现 Service 接口
完善界面细节（发送请求 与 数据展现）	单元测试

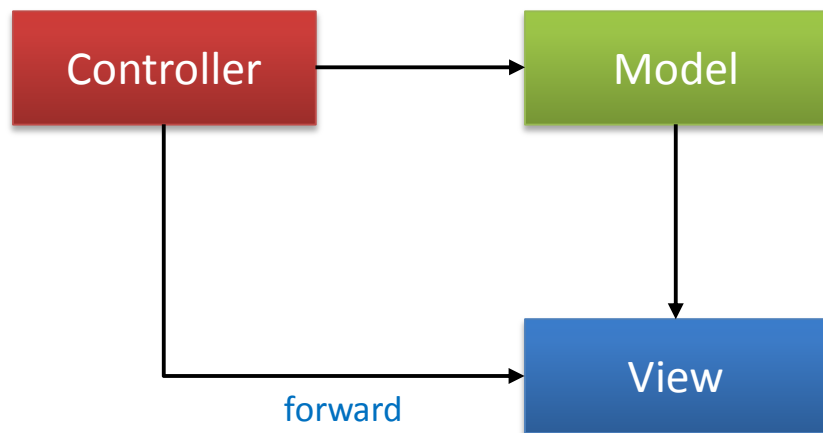


集成

框架初始化

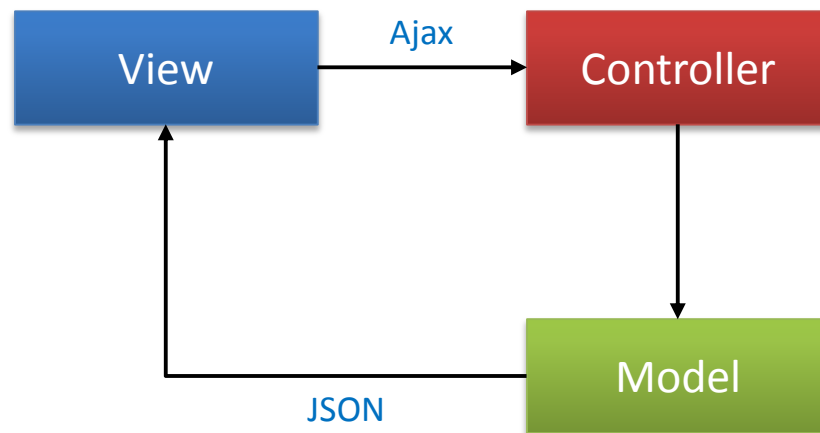


两种 MVC 模式



正向 MVC
推模式

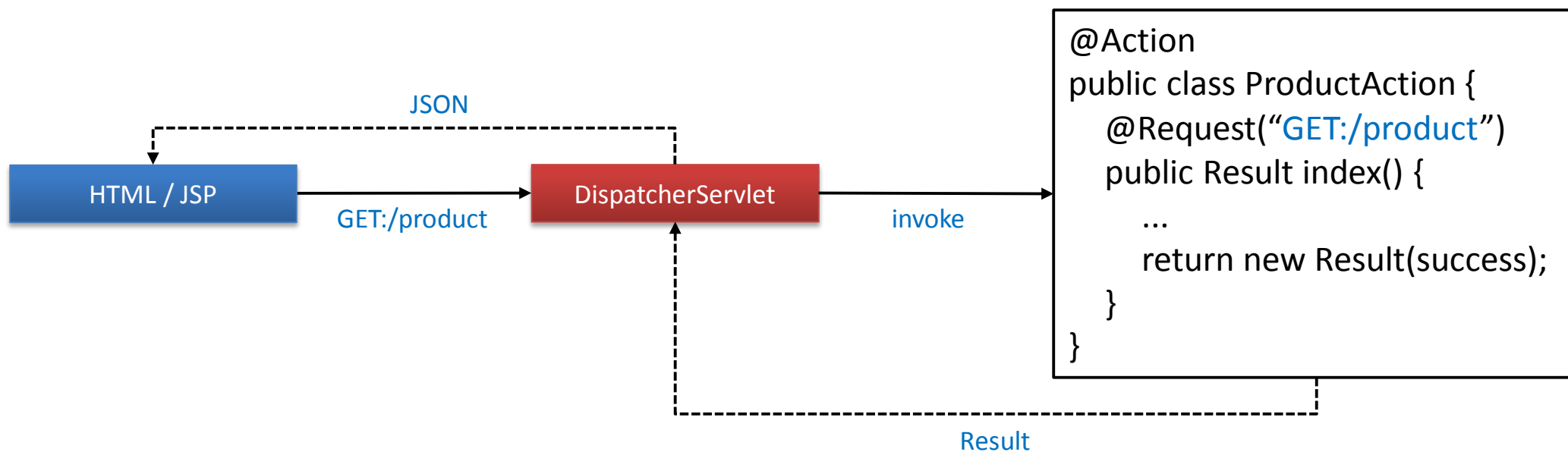
也称为传统 MVC，在 Controller 中获取 Model，并将 Model 转发（forward）到 View 中，这是推模式，可理解为：服务端将数据推给 View。



反向 MVC
拉模式

在 View 上发送 Ajax 请求到 Controller 中，在 Controller 中获取 Model，并将 Model 以 JSON 格式返回给 View，这是拉模式，可理解为：在 View 中将服务端数据拉回来。

请求与响应



在 HTML 或 JSP 中发送 GET:/product 请求，该请求被 Smart 的 com.smart.framework.DispatcherServlet 进行处理，根据请求方法与路径调用相应的 Action 方法，最终将结果以 JSON 格式返回。

依赖注入

```
@Action
public class ProductAction {
    @Inject
    private ProductService productService;

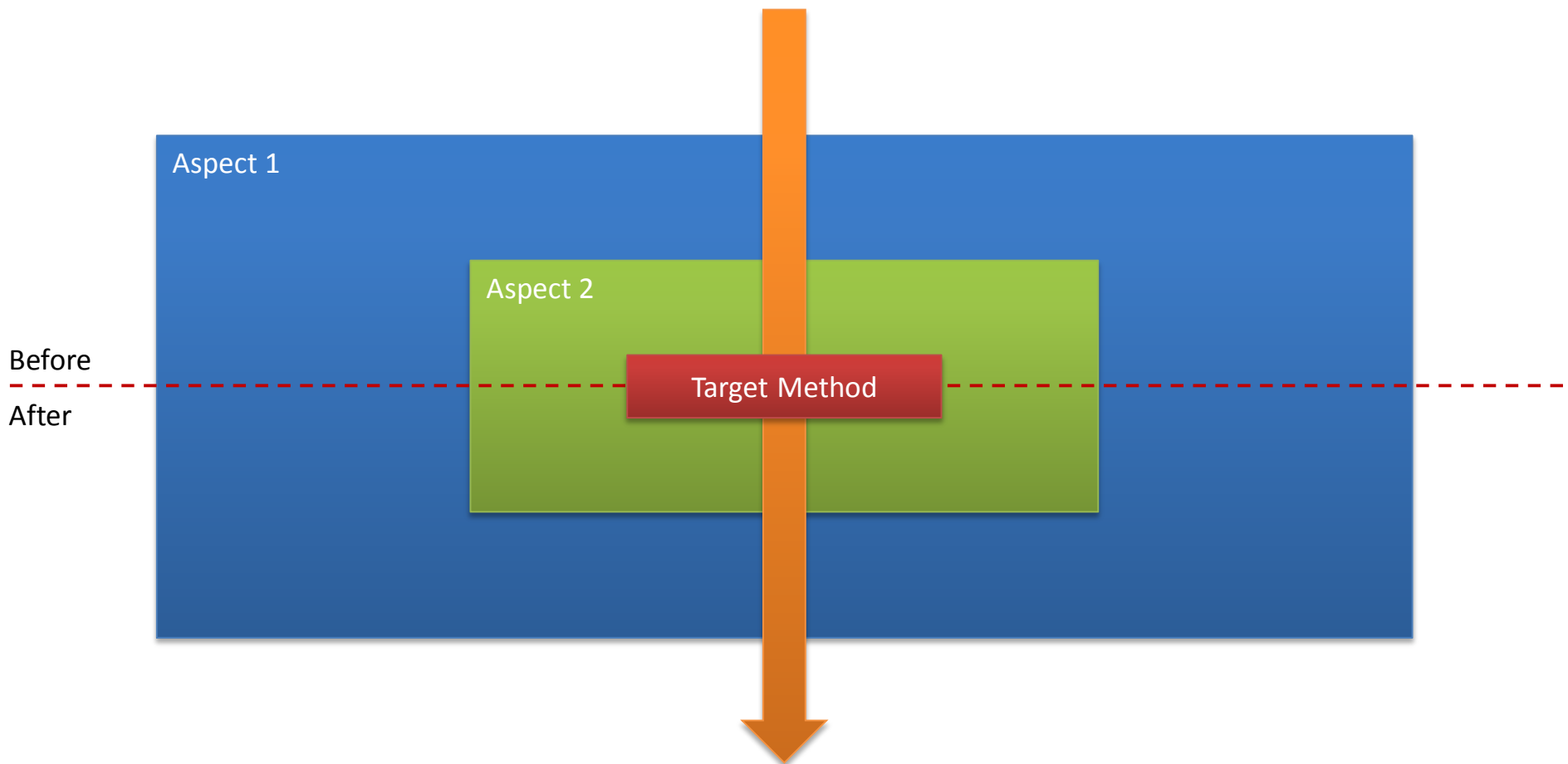
    @Request("GET:/product")
    public Result index() {
        List<Product> productList = productService.getProductList();
        return new Result(true).data(productList);
    }
}
```

将 ProductService 注入到 ProductAction 中

```
public interface ProductService {
    List<Product> getProductList();
}
```

```
@Service
public class ProductServiceImpl implements ProductService {
    public List<Product> getProductList() {
        ...
    }
}
```

方法拦截



在调用目标方法的前后都可进行 AOP 拦截，在调用之前，线程首先进入 Aspect 1，然后进入 Aspect 2，最后进入 Target Method。当调用完毕后，线程首先退出 Target Method，然后退出 Aspect 2，最后退出 Aspect 1。

实体关系映射

```
public class Product extends BaseEntity {  
  
    private long productTypeId;  
    private String name;  
    private String code;  
    private int price;  
    private String description;  
    private String picture;  
  
    // getter/setter ...  
}
```



product	
	id: bigint
	product_type_id: bigint
	name: varchar(100)
	code: char(5)
	price: int
	description: text
	picture: varchar(100)

实体类：











- ✓ 必须继承 com.smart.framework.base.BaseEntity 父类
- ✓ 实体名与属性名必须为驼峰风格
- ✓ 对于每个属性必须带有 getter/setter 方法

表结构：















- ✓ 表名必须均为小写
- ✓ 表名与列名必须为下划线风格

DataSet 与 DataHelper

实现单表的 CRUD 操作

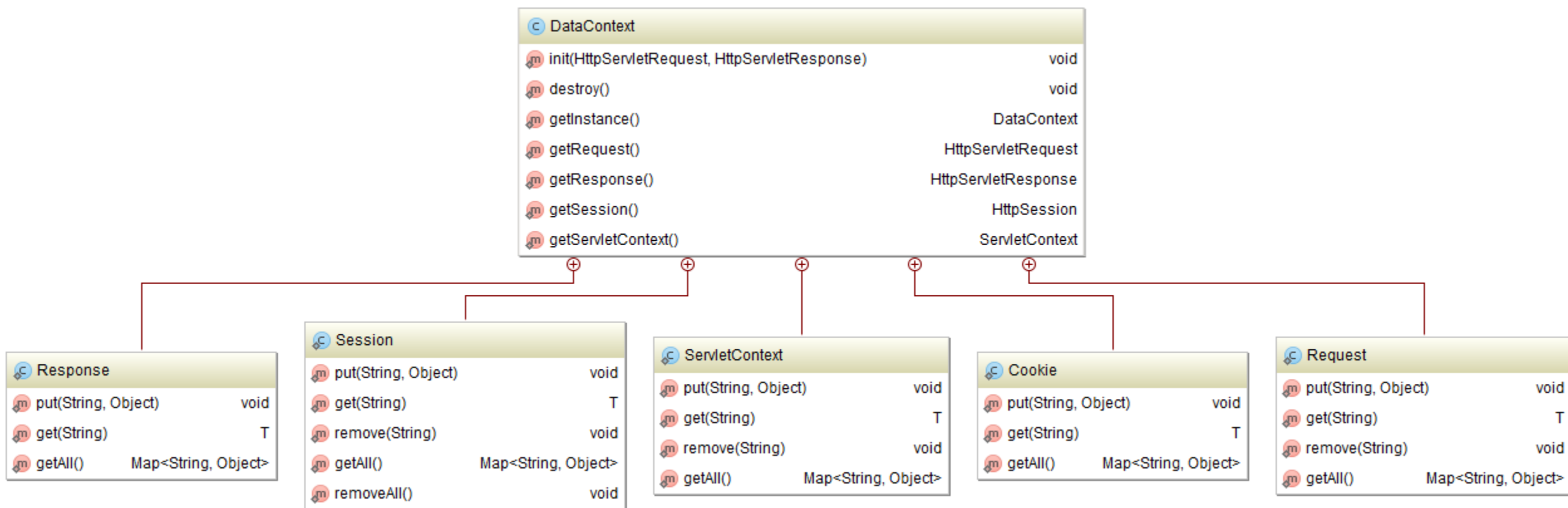
C DataSet		
	<code>select(Class<T>, String, Object...)</code>	<code>T</code>
	<code>selectList(Class<T>, String, String, Object...)</code>	<code>List<T></code>
	<code>insert(Class<?>, Map<String, Object>)</code>	<code>boolean</code>
	<code>update(Class<?>, Map<String, Object>, String, Object...)</code>	<code>boolean</code>
	<code>delete(Class<?>, String, Object...)</code>	<code>boolean</code>
	<code>selectCount(Class<?>, String, Object...)</code>	<code>long</code>
	<code>selectListForPager(int, int, Class<T>, String, String, Object...)</code>	<code>List<T></code>
	<code>selectMap(Class<T>, String, Object...)</code>	<code>Map<Long, T></code>
	<code>selectColumn(Class<T>, String, String, Object...)</code>	<code>T</code>
	<code>selectColumnList(Class<?>, String, String, String, Object...)</code>	<code>List<T></code>

实现复杂数据库操作

C DBHelper		
	<code>getDataSource()</code>	<code>DataSource</code>
	<code>getConnection()</code>	<code>Connection</code>
	<code>beginTransaction()</code>	<code>void</code>
	<code>commitTransaction()</code>	<code>void</code>
	<code>rollbackTransaction()</code>	<code>void</code>
	<code>getDbType()</code>	<code>String</code>
	<code>queryBean(Class<T>, String, Object...)</code>	<code>T</code>
	<code>queryBeanList(Class<T>, String, Object...)</code>	<code>List<T></code>
	<code>update(String, Object...)</code>	<code>int</code>
	<code>queryCount(String, Object...)</code>	<code>long</code>
	<code>queryMapList(String, Object...)</code>	<code>List<Map<String, Object>></code>
	<code>queryColumn(String, String, Object...)</code>	<code>T</code>
	<code>queryColumnList(String, String, Object...)</code>	<code>List<T></code>
	<code>insertReturnPK(String, Object...)</code>	<code>Serializable</code>

DataContext

实现 Servlet API 的封装



代码生成器

✓ 特性

- 快速创建 Smart 项目框架
- 快速生成代码骨架
- 快速测试与运行
- 快速打包

Smart Commands		
smart	create-app	: Create App
smart	create-entity <entity-name>	: Create Entity
smart	create-service <service-name>	: Create Service
smart	create-action <action-name>	: Create Action
smart	create-page <page-name>	: Create Page
smart	create-crud <crud-name>	: Create CRUD
smart	load-dict <dict-path>	: Load Dict
smart	run-test	: Run Test
smart	run-app	: Run App
smart	build-app	: Build App

✓ 源码

- Smart SDK: <http://git.oschina.net/huangyong/smart-sdk>
- Smart Generator: <http://git.oschina.net/huangyong/smart-generator>

技术路线图



相关资源

✓ 源码地址

- <http://git.oschina.net/huangyong/smart-framework>

✓ 应用示例

- <http://git.oschina.net/huangyong/smart-sample>

✓ 系列博文

- <http://my.oschina.net/huangyong/blog/158380>

谢谢
