代码生成器的正确用法

AXELOR 框架快速实现 CRUD（即使是用 JAVA）

老革命 <oldrev@gmail.com>

大纲

做 Java 开发为什么累?

元编程与代码生成器

Axelor 简介

Axelor 演示

Java 开发为什么累?

用 JAVA 开发业务管理系统也许是软件行业最无聊的工作

为什么用 Java?

Java 是使用人数最多的语言：

- 最多的库/包/框架
- 最多的文档/资料
- 最容易招人
- 基础设施的大量投资：JVM 优化，调试/测试/覆盖率/性能测量工具等等

长期培养培养出来的迷信和惯性

一个完美的 DRY 反例

典型 Java Web 应用做个 CRUD

添加一个表需要：

1. 领域模型实体
2. 几乎和实体定义一样的 DTO
3. ORM 的映射定义
4. 数据访问层 Repository 的接口+实现
5. 业务逻辑层服务接口+实现
6. MVC 控制器？

表里增加个字段，上面全部改一遍

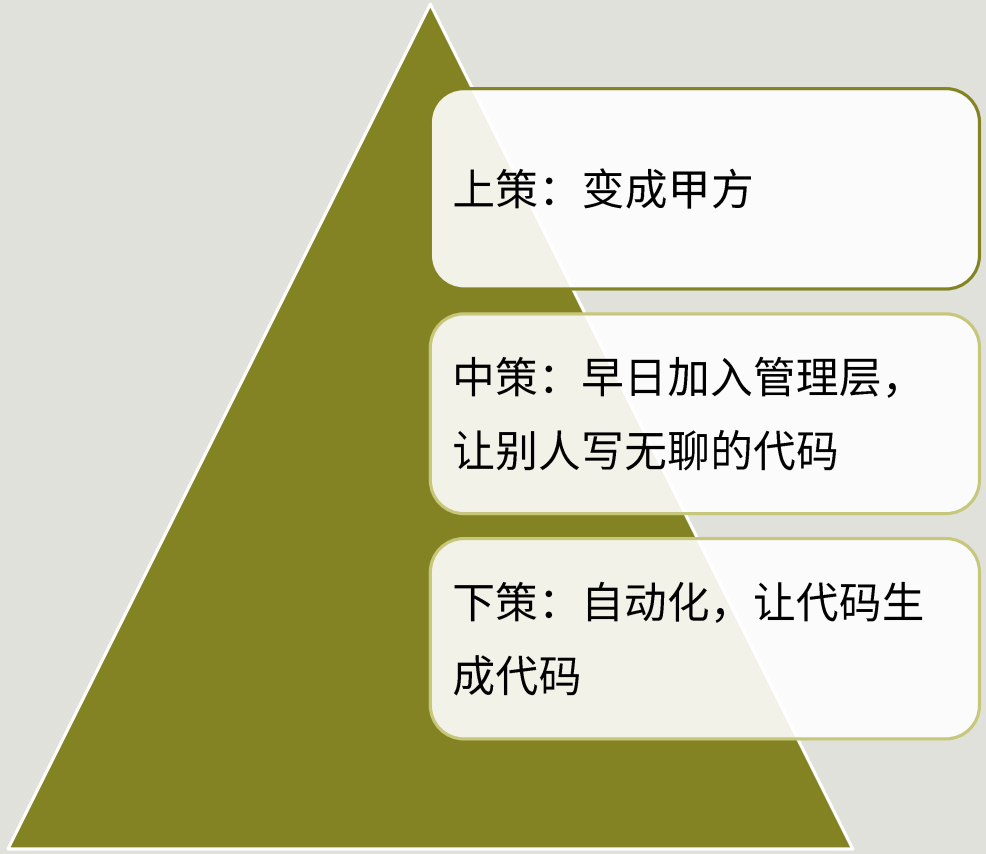
还有一件小事——前端

一个良好体验的单页 HTML 应用，前端的工作量大于后端

- 想想复杂字段：多对一、一对多、多对多做 UI 控件
- 列表要搜索、要组合条件搜索、要排序
- 这个要行内（inline）编辑，那个要弹出对话框
- …

手机应用

如何减小工作量?



元编程与代码生成器

生成代码的代码

元编程的实现方式

脚手架（一次性代码生成器）

- 某红色宝石铁路 Web 框架的脚手架
- 各种 IDE 的文件模板、项目模板功能

编译时静态代码生成

- 外部：Flex/Bison
- 内部：C++ 模板或者其他有宏的语言

运行时动态代码生成

- 各种动态语言
- Java/.NET 运行时生成字节码/IL 代码

脚手架——Evil Wizards

发射后不理的方式，只能用一次

自动帮你添加的代码越多越有害健康

只适合短小的代码片段，比如添加个类文件、添加个接口文件。

运行时动态代码生成

动态语言：天赋技能

静态语言：

1. 放弃了编译器检查、优化和类型安全
2. IDE 无法提供自动完成等功能
3. 人类充当编译器来生成“机器码”，难写难读
4. 一般用于代码量很少的简单情况：动态代理、反射处理等
5. 就算是把编译器做成库来调用，也无法回避 1. 与 2.

编译时代码生成

内部方式：Java 语言自身并无元编程能力，只能使用外部工具

外部方式：

- 通过某种来源生成特定语言的源代码文件
- 通过更改来源来修改生成后的结果
- 数据来源是唯一的单点定义

所以，

编译时代码生成器其实就是一种编译器……

Axelor 框架简介

开源模块化管理系统开发框架

Axelor 是什么?

Axelor Development Kit(ADK)：一个模块化 Java Web 数据库应用快速开发框架

Axelor Business Suite(ABS)：基于 ADK 开发的 ERP 套装模块

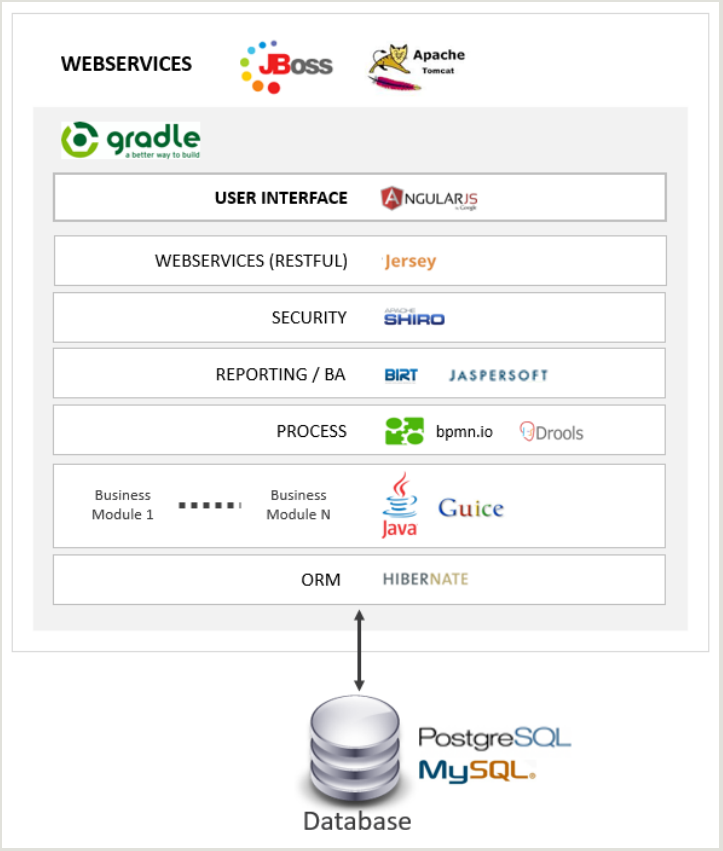
Axelor Process Studio(APS)：Axelor 的业务流程套件模块

Axelor ABS 是完整的下一代 ERP 套件，完全开源，不分企业版和社区版。ABS 着重发力于移动化、社交化和本地化——SoLoMo（Social, Local, Mobile）

如果你听过 Odoo…

AXELOR 就是把 ODOO 用 JAVA 重新实现的改进版

Axelor 技术架构


WEBSERVICES
JBoss
Apache Tomcat
gradle
a better way to build
USER INTERFACE
ANGULARJS
WEBSERVICES (RESTFUL)
Jersey
SECURITY
APACHE SHIRO
REPORTING / BA
BIRT
JASPERSOFT
PROCESS
bpmn.io
Drools
Business Module 1
Business Module N
Java
Guice
ORM
HIBERNATE
PostgreSQL
MySQL
Database

关键特性

纯模块化开发，模块可以安装、卸载

前端免开发，真正手机可用的响应式设计

全自动数据库迁移

集成数据导入导出、定时任务、权限管理等常用功能

多数据支持：PostgreSQL、MySQL、Oracle（5.0）

Axelor 是同名公司的唯一产品，不会出现类似某些企业为 KPI 开源的情况

And more…

设计哲学

框架牺牲一点灵活性，换来大大节约开发工作量

约定胜于配置，数据胜于代码：

- 代码和数据放到约定的目录里就可被自动发现及使用
- 种子数据、测试数据和演示数据数据均以文本格式（XML/CSV）跟随代码一起管理
- 运行阶段将文本文件的数据导入数据库

开发不依赖数据库状态，方便部署开发环境、测试环境及持续集成

由依赖注入组装各个部件，便于测试及替换

通过 Groovy 语言提供很多运行时用户可定制的功能

开发阶段的工具均集成在构建流水线中

Axelor 为什么开发速度快?

传统 JAVA SPRING WEB 应用开发

1. 领域模型：Java 代码
2. DTO：Java 代码
3. ORM 映射：XML 或者 Java 代码
4. 数据访问层：接口+实现
5. 业务逻辑层：接口+实现
6. 展现层 MVC：Java 代码+视图模板+…
7. 前端：比 Java 代码还多

AXELOR 应用开发

1. 领域模型：XML 定义
2. DTO：不需要
3. ORM 映射：自动生成
4. 数据访问层：自动生成（可继承覆盖）
5. 业务逻辑层：内置+按需实现
6. 展现层 MVC ：不需要
7. 前端：无需开发

一个 Axelor 实体的 XML 定义

```
<entity name="Customer" cachable="true">
  <string name="code" required="true" unique="true" title="客户编号"/>
  <string name="name" required="true" namecolumn="true" title="姓名"/>
  <string name="address" title="地址"/>
  <string name="notes" large="true" title="备注"/>
  <one-to-many name="orders" ref="com.axelor.orders.db.SalesOrder" mappedBy="customer" />
</entity>
```

有了Entity 的 XML 定义你就免费得到：

1. 领域模型的 Java 类定义
2. 数据访问层的特化 Repository 类定义
3. 数据库自动建表、自动迁移
4. 自动生成 Repository
5. JSON RESTFul Web API
6. XML 定义的实体源数据可通过 Web API 访问

Java 表现力比较低...

XML 定义

```xml
<entity name="Customer" cachable="true">
  <string name="code" required="true" unique="true" title="客户编号"/>
  <string name="name" required="true" namecolumn="true" title="姓名"/>
  <string name="address" title="地址"/>
  <string name="notes" large="true" title="备注"/>
  <one-to-many name="orders" ref="com.axelor.orders.db.SalesOrder" mappedBy="customer" />
</entity>
```

生成的 JAVA 实体类（~ 200 行）

Axelor 开发演示

先定义个需求

肉铺订单管理系统

- 客户信息维护
- 订单信息维护

三个表：

- 用户表：Customer
- 订单表：Order，包含多笔订单明细
- 订单明细表：OrderDetail

准备工作

安装配置 Java 8 环境

安装配置 Java 构建工具 Gradle

安装配置 Axelor-Development-Kit

安装配置 PostgreSQL 9.5+ 数据库

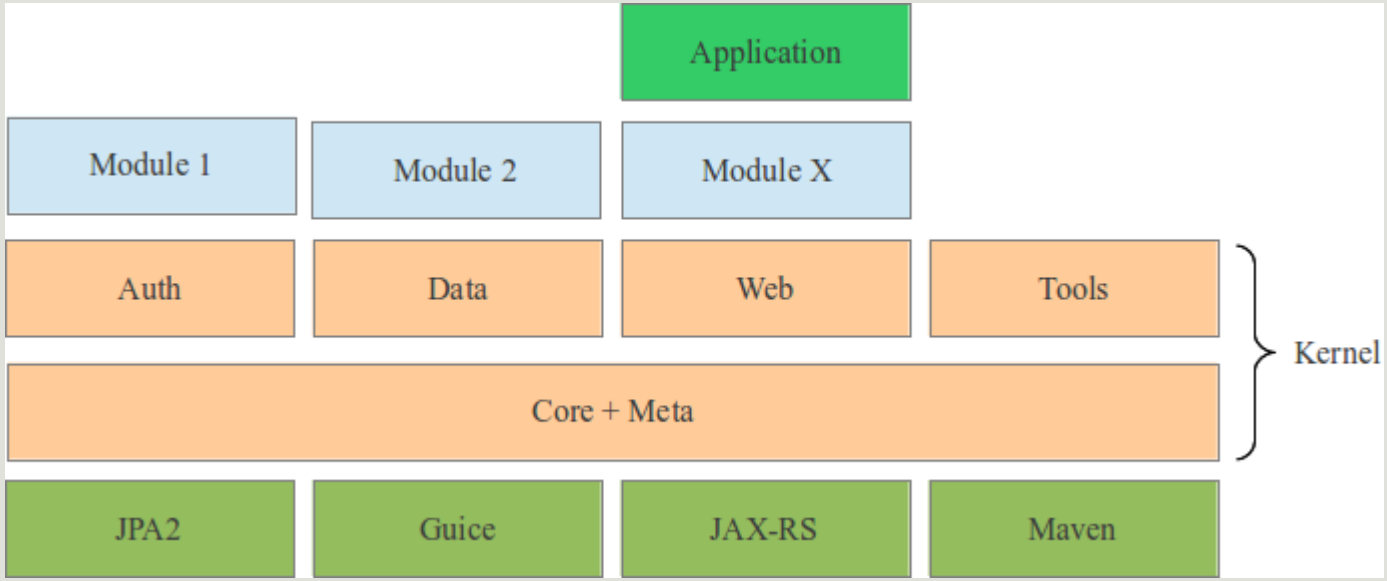
创建 PostgreSQL 用户，并创建该用户拥有的空白测试数据库

脚手架登场了！

1. 使用 Axelor 命令行工具创建新 Java 项目
2. 项目名称为 “my-axelor-app”
3. 创建以后可以导入 Eclipse/IntelliJ 等
4. 进入“my-axelor-app”目录可看到创建的项目结构
5. 创建订单管理模块“orders”

```
$axelor --new my-axelor-app
$cd my-axelor-app
$axelor new-module --name orders
```

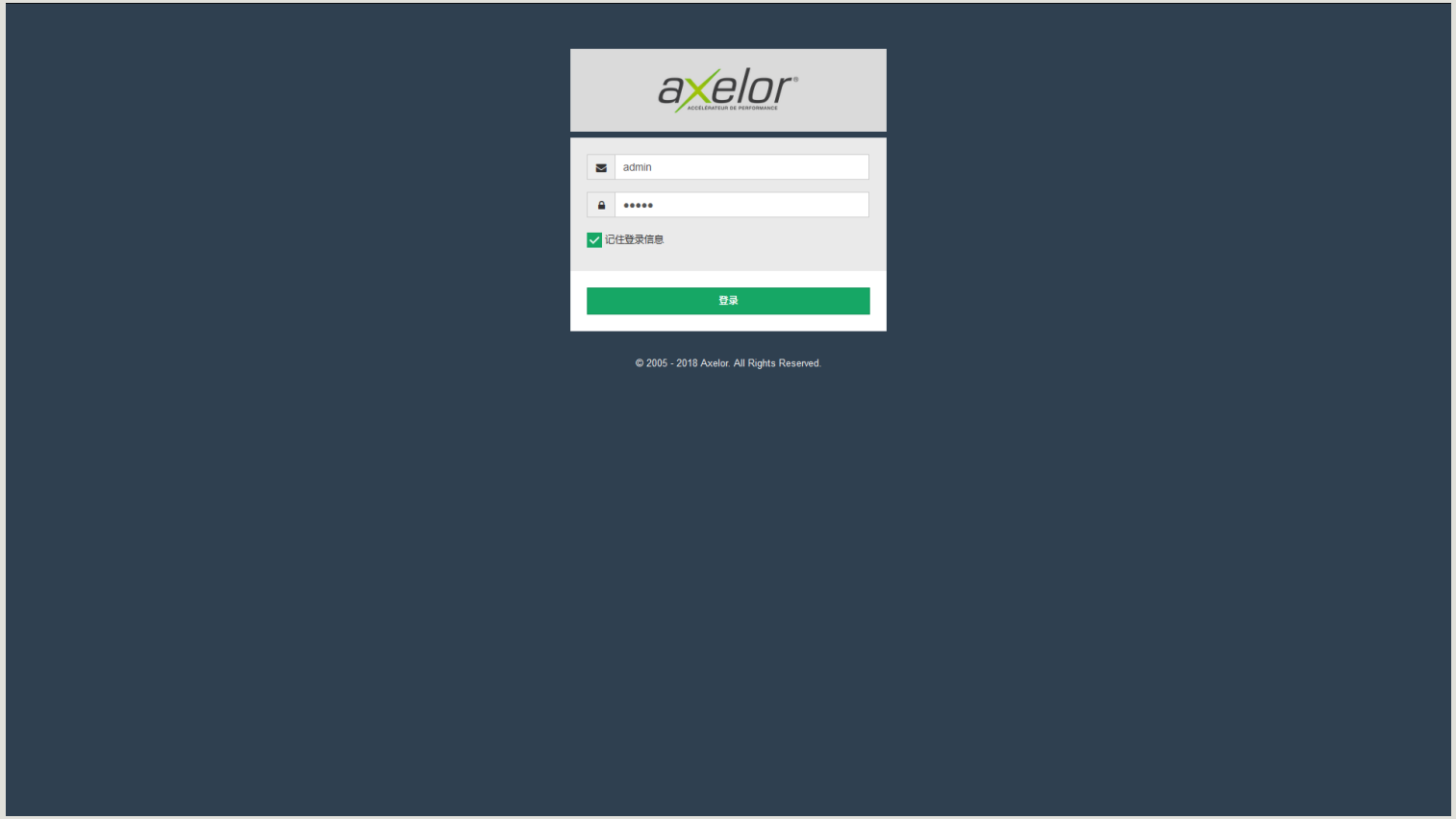
Axelor 应用的架构



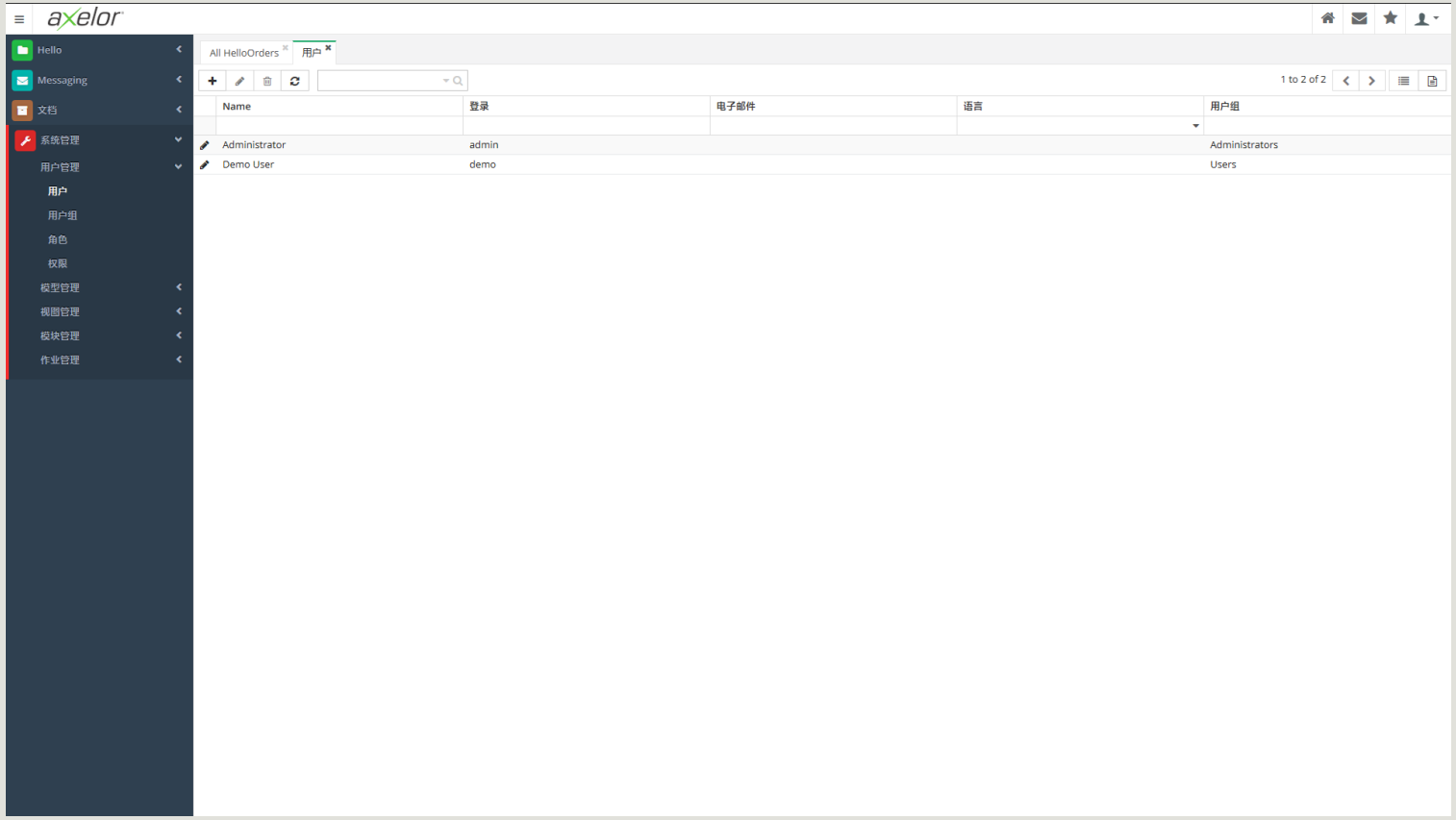
进行必要的修改及启动 Axelor

1. 修改 `src/main/resources/application.properties` 文件中的数据库连接信息
2. 修改 `src/settings.gradle`，增加包含新建的模块：`include ‘modules:orders ‘`
3. 修改 `src/build.gradle`，增加注册新建的模块：`module ‘modules:orders’ `
4. 在项目根目录中运行 `$gradle tomcatRun` 启动 Axelor
5. 通过浏览器打开 http://localhost:8080/my-axelor-app
6. 配置无误即可看到 Axelor 的登录界面。

初步体验下 Axelor 的界面：登录



初步体验下 Axelor 的界面：主界面



Axelor 里实现我们的业务需求

1. 添加实体定义 XML
2. 添加视图
3. 添加菜单及动作
4. 没了

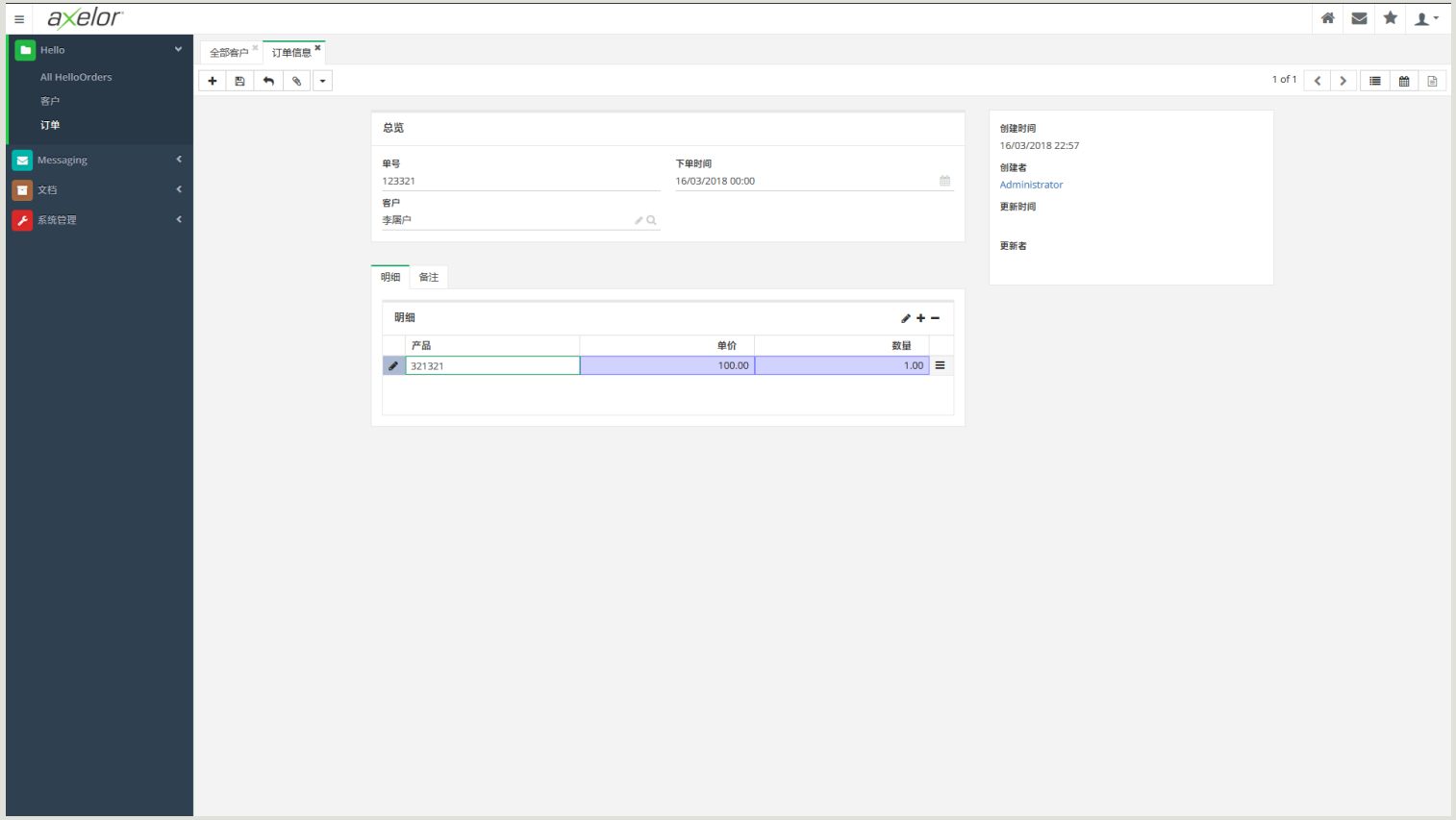
实体定义：客户表

```
<entity name="Customer" cachable="true">
  <string name="code" required="true" unique="true" title="客户编号"/>
  <string name="name" required="true" namecolumn="true" title="姓名"/>
  <string name="address" title="地址"/>
  <string name="notes" large="true" title="备注"/>
  <one-to-many name="orders" ref="com.axelor.orders.db.SalesOrder" mappedBy="customer" />
</entity>
```

实体定义：订单及明细表

```
<entity name="SalesOrder" cachable="false">
  <string name="name" required="true" unique="true" title="单号" sequence="orders.order.seq"/>
  <datetime name="orderDate" title="下单时间" />
  <many-to-one name="customer" ref="com.axelor.orders.db.Customer" required="true" title="客户" />
  <one-to-many name="details" ref="com.axelor.orders.db.SalesOrderDetail" mappedBy="salesOrder" title="明细"/>
  <string name="notes" large="true" title="备注"/>
</entity>

<entity name="SalesOrderDetail" cachable="false">
  <many-to-one name="salesOrder" ref="com.axelor.orders.db.SalesOrder" required="true" title="订单" />
  <integer name="sequence" />
  <string name="product" required="true" title="产品"/>
  <decimal name="unitPrice" required="true" title="单价"/>
  <decimal name="quantity" required="true" title="数量"/>
</entity>
```

接下来是展示层


axelor
Hello
All HelloOrders
客户
订单
Messaging
文档
系统管理
全部客户
订单信息
1 of 1
总览
单号
123321
下单时间
16/03/2018 00:00
客户
李属户
创建时间
16/03/2018 22:57
创建者
Administrator
更新时间
更新者
明细
备注
明细
产品
单价
数量
321321
100.00
1.00

视图

视图是前端展示 UI 的布局

视图种类：列表、树状列表、日历、图表、看板、甘特图

视图使用 XML 定义布局，由前端决定如何渲染成界面

视图只是种特殊的种子数据，在安装模块时导入数据库

运行时使用数据库中的视图数据

视图允许继承、动态替换

视图文件必须放在 `your_app/modules/your_module/src/resources/views/` 中。

客户视图定义

```
<grid name="customer-grid" title="客户列表" model="com.axelor.orders.db.Customer">
        <field name="code"/>
        <field name="name"/>
        <field name="address"/>
</grid>

<form name="customer-form" title="客户信息" model="com.axelor.orders.db.Customer">
        <panel title="基本信息">
                <field name="name" />
                <field name="code" />
                <field name="address" x-span="12"/>
        </panel>
        <panel title="备注">
                <field name="notes" showTitle="false" x-span="12"/>
        </panel>
        <panel sidebar="true">
                <field name="createdOn"/>
                <field name="createdBy"/>
                <field name="updatedOn"/>
                <field name="updatedBy"/>
        </panel>
</form>
```

订单及明细相关视图定义

```
<grid name="salesorder-grid" title="订单列表" model="com.axelor.orders.db.SalesOrder">
      <field name="name"/>
      <field name="customer"/>
  </grid>

  <form name="salesorders-form" title="订单信息" model="com.axelor.orders.db.SalesOrder">
      <panel title="总览">
          <field name="name" />
          <field name="orderDate"/>
          <field name="customer"/>
      </panel>
      <panel-tabs>
          <panel-related title="明细" field="details" editable="true" orderBy="sequence" canMove="true" >
              <field name="product"/>
              <field name="unitPrice"/>
              <field name="quantity"/>
          </panel-related>
          <panel title="备注">
              <field name="notes" showTitle="false" colSpan="12" widget="html" />
          </panel>
      </panel-tabs>
      <panel sidebar="true">
          <field name="createdOn"/>
          <field name="createdBy"/>
          <field name="updatedOn"/>
          <field name="updatedBy"/>
      </panel>
  </form>

  <calendar name="salesorder-calendar" title="订单日历" model="com.axelor.orders.db.SalesOrder"
            editable="false" eventStart="orderDate" eventLength="8"
            colorBy="customer">
      <field name="name" />
  </calendar>
```

菜单与动作

菜单及动作均定义在 XML 文件中，也是种特殊的种子数据，模块安装时导入数据库

菜单关联到动作，动作定义要执行的操作

可用的操作：

- 打开实体的视图
- 弹出对话框
- 服务器端动作
- 调用 Java 方法
- 执行 Groovy 脚本
- …

客户菜单及动作

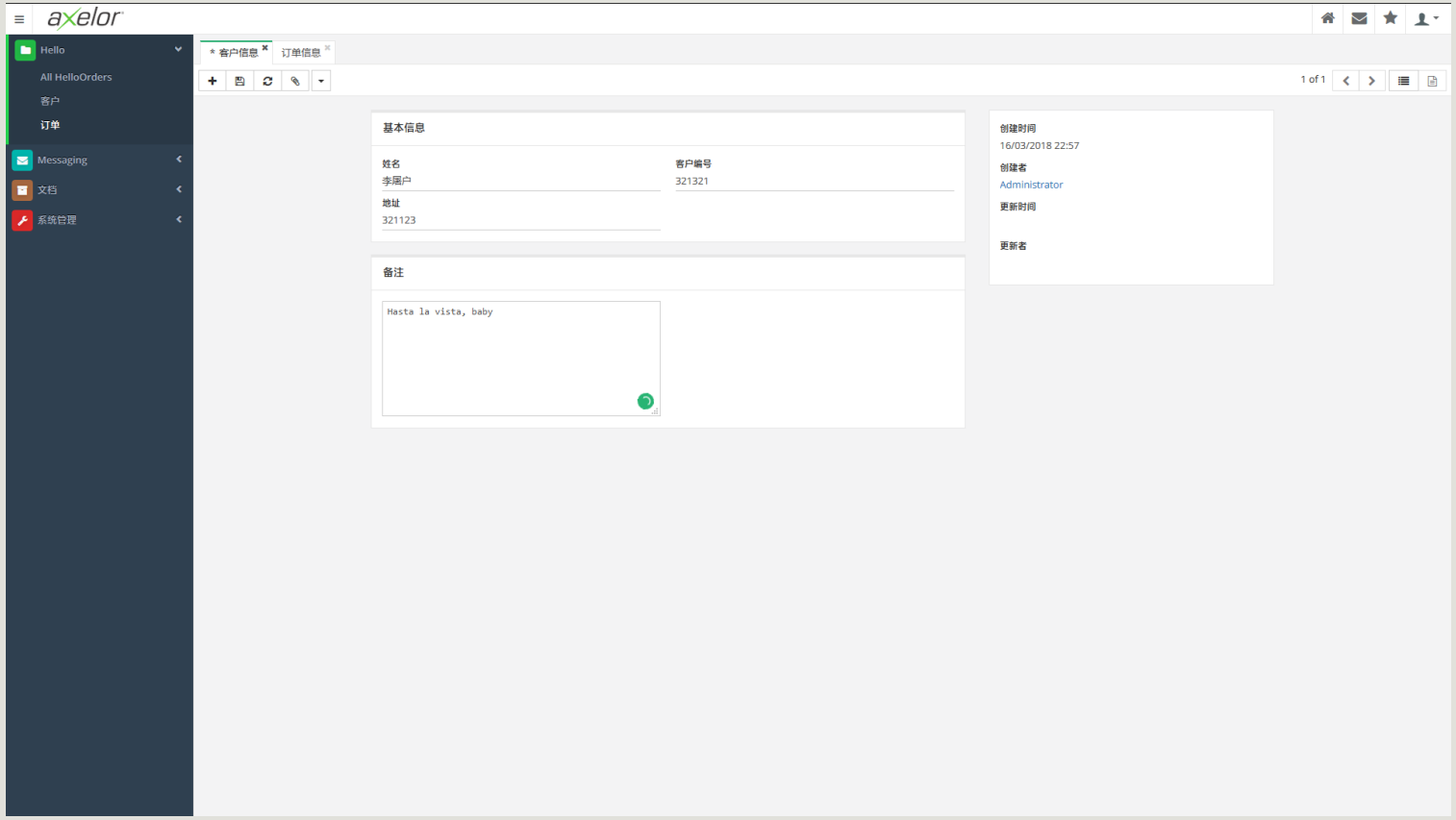
```
<menuitem name="menu-customer-all" parent="menu-hello"
title="客户"
action="customer.all"/>

<action-view name="customer.all" title="全部客户" model="com.axelor.orders.db.Customer">
<view type="grid" name="customer-grid"/>
<view type="form" name="customer-form"/>
</action-view>
```

订单菜单及动作

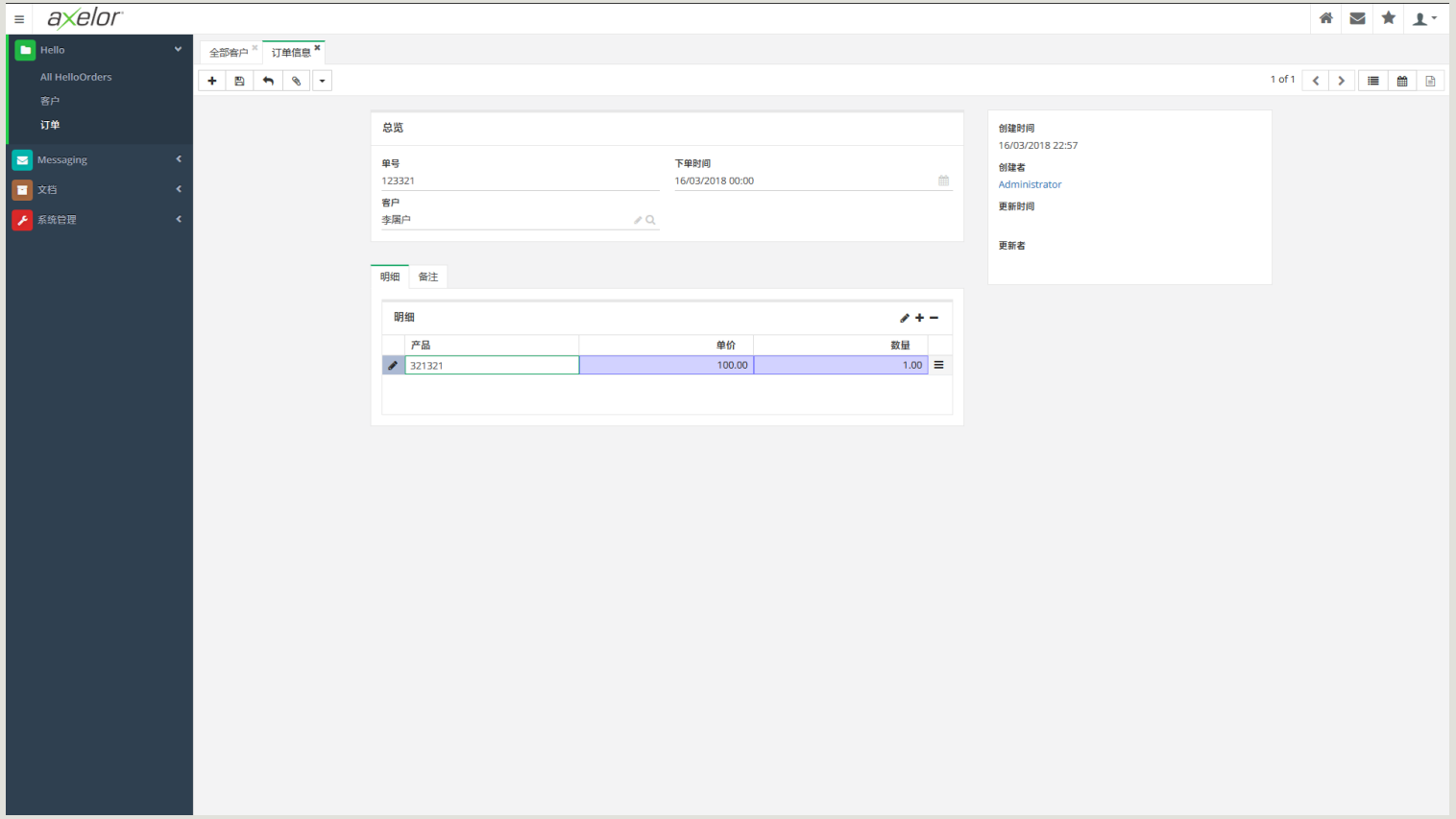
```
<menuitem name="menu-order-all" parent="menu-hello"
title="订单" action="order.all" />

<action-view name="order.all" title="全部订单" model="com.axelor.orders.db.SalesOrder">
<view type="grid" name="salesorder-grid"/>
<view type="form" name="salesorder-form"/>
<view type="calendar" name="salesorder-calendar"/>
</action-view>
```

最终效果

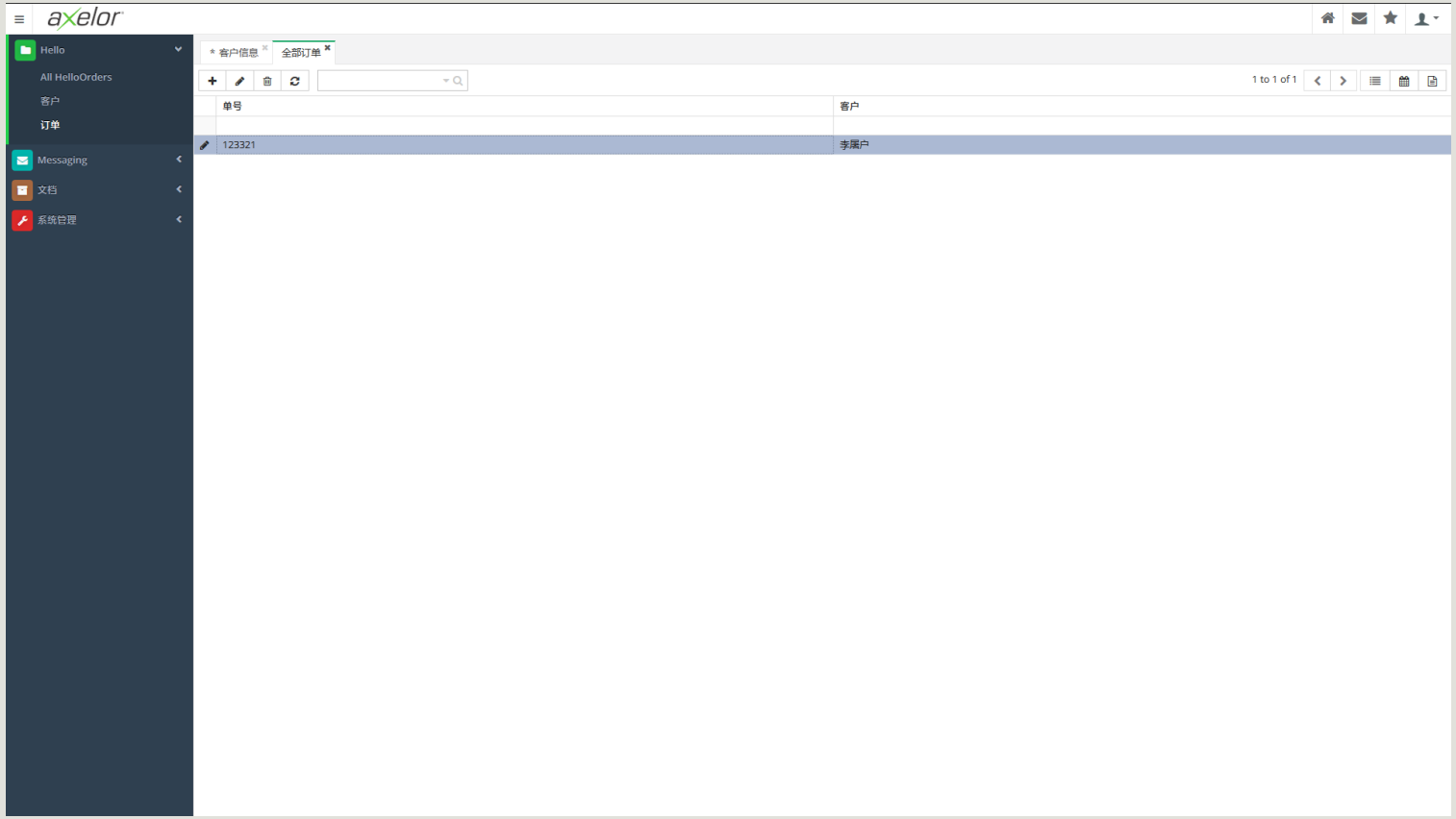
客户表单


axelor
Hello
All HelloOrders
客户
订单
Messaging
文档
系统管理
* 客户信息
订单信息
1 of 1
基本信息
姓名
李屠户
客户编号
321321
地址
321123
创建时间
16/03/2018 22:57
创建者
Administrator
更新时间
更新者
备注
Hasta la vista, baby

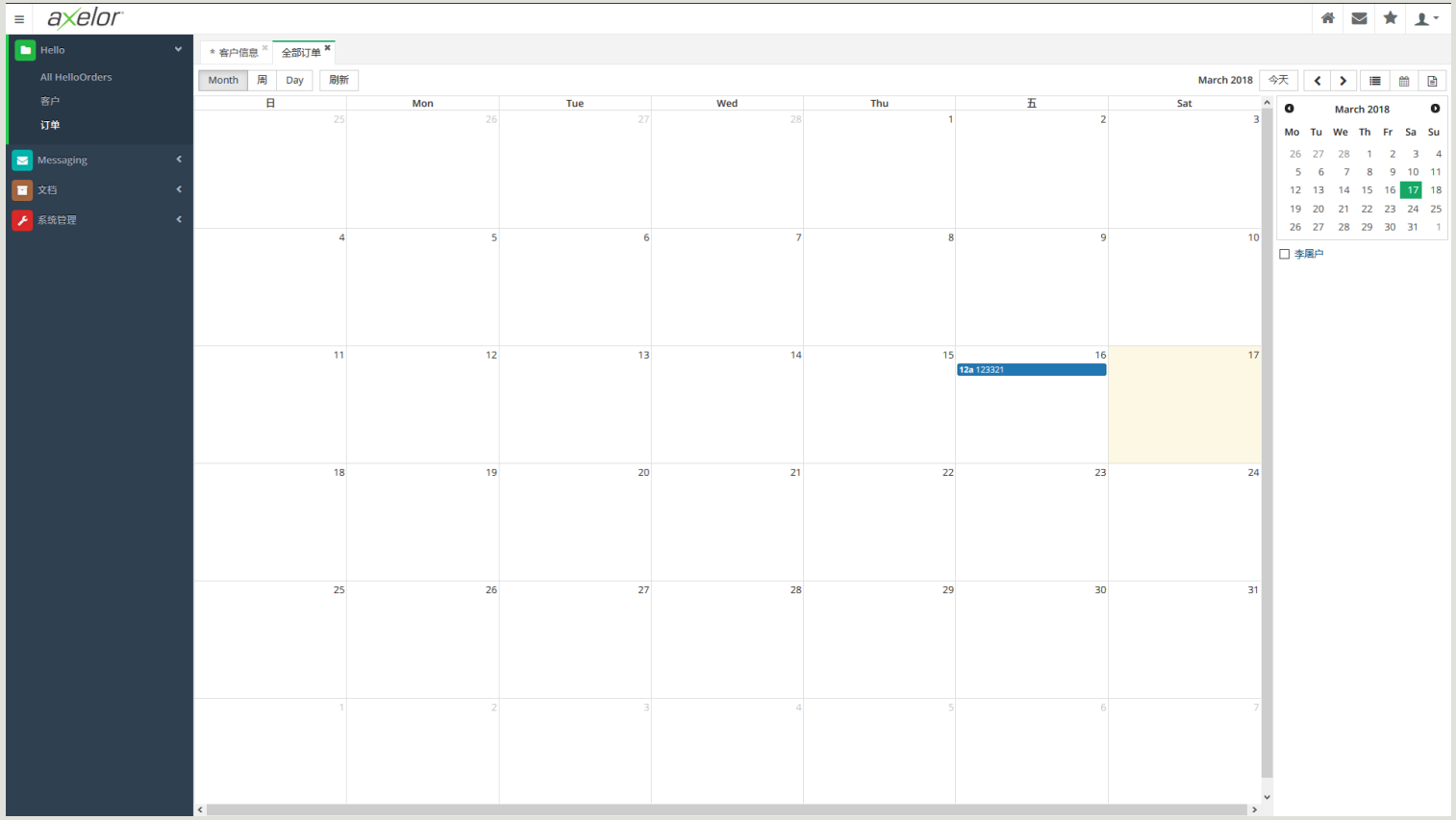
订单表单


axelor
Hello
All HelloOrders
客户
订单
Messaging
文档
系统管理
全部客户
订单信息
1 of 1
总览
单号
123321
下单时间
16/03/2018 00:00
客户
李属户
创建时间
16/03/2018 22:57
创建者
Administrator
更新时间
更新者
明细
备注
明细
产品
单价
数量
321321
100.00
1.00

订单列表


axelor
Hello
All HelloOrders
客户
订单
Messaging
文档
系统管理
* 客户信息
全部订单
1 to 1 of 1
单号
客户
123321
李黑户

订单日历视图


axelor
Hello
All HelloOrders
客户
订单
Messaging
文档
系统管理
* 客户信息
全部订单
Month
周
Day
刷新
March 2018
今天
日
Mon
Tue
Wed
Thu
五
Sat
12a 123321
March 2018
Mo Tu We Th Fr Sa Su
李雨户

高级主题

级联字段

自定义 JSP 页面：登录、管理后台等

权限配置

自定义业务层服务、自定义控制器

定时任务

Scala/Kotlin 支持

工作流

社交化

…

实际演示时间！

参考

Axelor 官网：http://www.axelor.com

Axelor 开发者文档： http://docs.axelor.com

Axelor 代码库： https://github.com/axelor

Axelor 第三方文档： https://sandwind.gitbooks.io/axelor-doc/

本文演示程序下载：https://github.com/oldrev/yncoder-2018-spring

感谢！