Hands-on: Developing Scout Business applications

Spring, Hibernate, Maven
Agenda

Eclipse Demo Camp 2013, Zürich

• Intro
• Use Cases
• Architecture
• Front-end: Scout
• Back-end: Spring and Hibernate
• Industrialization of build process (Maven)
• Outlook
Status of presented concept

Intro

• Development during last 8 months
• Going live 2014 and 2015
• Work in progress
Customer projects

Use Cases

• requirements
  o Rich Client, RCP/SWT
  o Multi Layer Backend
  o Multi Module, base project and dependent projects
    Core (shared), product A, product B
  o CI, Maven
  o Mainframe (run down -> DB independency)
  o Deployment environment: Citrix (RCP Client), IBM Websphere
    Application Server, IBM DB2 database
  o Maintainability (>20 man years of development)
Verification with other projects

Use Cases

• requirements
  ○ Replace home made UI abstraction in RCP and RAP (Scout)
  ○ Replace home made persistence abstraction (Spring)
  ○ Industrialization of build process (Maven)
Classical Multi Tier Architecture

Architecture

Front-end
- RCP Client
- Web Client
- Batch Client
- ...

Crosslayer / Shared (Model, DTO, Utilities)

Back-end
- Service Layer
- Business Layer
- Persistence Layer

Mainframe
- IMS
- Transaction
- SAP
- ...

DB
Scout...we like

Front-end: Scout

- efficiency
- SWT, Swing, Web, Mobile
- Client Server communication
- Security
Scout, but... 

**Front-end: Scout**
Backend considerations

• Technology: EJB, WebService, ... ?
• OSGI bundles together with Scout Server
• Standard JAR projects with manual MANIFEST.MF
• Problem: location of MANIFEST.MF – maven vs. eclipse
Dependency Injection Spring

Back-end

- The Scout way to call a Scout Service:
  SERVICES.getService(IStandardOutlineService.class)

- Our way to call a backend Service
  BACKEND.getService(IBasicDataService.class)

- BACKEND.java
  ApplicationContext ctx = new ClassPathXmlApplicationContext(new String[] {"...", "...");
  bean = ctx.getBean(serviceInterfaceClass);

- Service bean instantiated by Spring -> deeper layers can use injection: @Service, @Resource ...
Hibernate – Database access

Back-end

• JPA generic vs. Hibernate specific
• Annotation vs. XML config
• OSGI...Classloader issues
• “normal” issues with legacy databases (composite-id,...)
• Transaction boundaries, lazy loading
Scout...but 2\textsuperscript{nd}

Back-end

@Override
public void execLoad() throws ProcessingException{
    ICompanyService service = SERVICES.getService(ICompanyService.class);
    CompanyFormData formData = new CompanyFormData();
    exportFormData(formData);
    formData = service.load(formData);
    importFormData(formData);
    setEnabledPermission(new UpdateCompanyPermission());
}

![Diagram of New Service Operation](image)
Maven

Industrialization of build process

• Tons of projects
  ○ Parents
  ○ Targets
  ○ Features (+ Test)
  ○ jar / OSGI bundles
  ○ Plugins (+ Test)
  ○ Products, Repositories

• Sequence
  ○ Build Parents, Targets -> Maven Repo
  ○ Build Backend OSGI bundles (POM) -> Maven Repo
  ○ Build Remaining (Tycho) -> P2 Repo

• P2 Repo
Problem areas and outlook

Lessons learned and outlook

• Positive
  ○ Efficiency
  ○ Flexibility

• Problems
  ○ OSGI
  ○ Target
  ○ P2 Repo

• Next steps
  ○ Finalize Hibernate lazy loading
  ○ Check elimination of Scout form data, Object[][]
  ○ Spring – Gemini blueprint
  ○ Client notifications – multi node