Full-stack Applications with Open Source Technologies

ECE 2016, Ludwigsburg

@ZimMatthias, Matthias Zimmermann, BSI Business Systems Integration AG
Agenda

➡ Application Stack Goals
➡ Overview + Task App
➡ Spring Boot
➡ Eclipse Scout
➡ Demo «add a Field»
➡ Wrap-up
Goals for the Application Stack

1. Simplify the Lifes of the Developers
2. Allow for large and complex Apps
3. Provide Long Term Sustainability
4. The «right» Tool for the «right» Job

Requirements

- Java (Sustainability)
- Open Source (Sustainability)
- Proven in Practice (no Surprises)
- Small number of Frameworks/Components (Maintainability)
Overview of the Application Stack

- Browser
- Presentation
- Services, Persistence
- Database
Spring Boot
Spring Boot

- **Spring** is strong for Server-Side Programming
- **Spring Boot**: Radically faster ”getting started experience”
- Production ready: Auditing, health and metrics gathering ...
Spring Boot

Takes an opinionated view of building production-ready Spring applications. Spring Boot favors convention over configuration and is designed to get you up and running as quickly as possible.

QUICK START
Spring Boot «Hello World»

```java
package com.example;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @RequestMapping("/")
    public String index() {
        return "Hello Ludwigsburg 2";
    }
}
```
Spring Boot «Tasks»

Main Dependencies

- spring-boot-starter-web
- spring-boot-starter-data-jpa
- lombok (https://projectlombok.org/)

«Tour»

- Application (Spring Boot Application)
- Task (Model Class)
- TaskService (Service Interface)
- DefaultTaskService (Service Implementation)
- TaskRepository (Persistence: JPA)
- TaskController (API: REST)
public class Task implements Persistable<UUID> {

    private static final long serialVersionUID = 1L;

    @Id
    @Type(type = "uuid-char")
    @NonNull
    private UUID id = UUID.randomUUID();

    // this is a jpa entity
    @Entity
    // lombok annotations
    @Getter
    @Setter
    @NoArgsConstructor
    @RequiredArgsConstructor
    @EqualsAndHashCode(of = "id")
    @ToString
}
Eclipse Scout
Eclipse Scout

- **Eclipse Scout**: is strong for building UI’s
- Comes with an abstract Application Model
- Scout code is UI technology independent
- Multi-Device support
- Proven in Production - even longer than Spring ;-)
Create the Scout "Hello World" application
Create a new Scout maven-based archetype project.

Import the Scout demo applications
Import the Scout demo apps using Eclipse Oomph. In the installer search for "scout" and tick "Scout Demo App".

Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository.

Review IDE configuration settings
Review the IDE's most fiercely contested preferences.

Overview
Get an overview of the features

Tutorials
Go through tutorials

What's New
Find out what is new

Always show Welcome at start up
package helloworld;

public class ServerTimeField extends AbstractDateTimeField {
    @Override
    protected String getConfigurerLabel() {
        return TEXTS.get("ServerTime");
    }

    @Override
    protected int getConfigurerGridW() {
        return 2;
    }
}
Eclipse Scout «Tasks»

Main Dependencies

⇒ org.eclipse.scout.rt.ui.html
  − org.eclipse.scout.rt.client
  − org.eclipse.scout.rt.shared
  − org.eclipse.scout.rt.platform

«Tour»

⇒ HomeOutline  (Main Entry Point for Navigation)
⇒ AbstractTaskTablePage  (Parent Task Table)
⇒ InboxTaskTablePage  (User’s Inbox Table Page)
⇒ TaskForm  (Form to create and edit Tasks)
public class ModifyHandler extends AbstractDirtyFormHandler {

    @Override
    protected void execLoad() {
        setEnabledPermission(new UpdateTaskPermission());
        Task task = taskService.getTask(getTaskId());
        importFormFieldData(task);
        getForm().setSubTitle(calculateSubtitle());
    }

    @Override
    protected void execStore() {
        Task task = taskService.getTask(getTaskId());
        exportFormFieldData(task);
        taskService.saveTask(task);
    }
}
Eclipse Scout «Tasks»

Main Deviation form typical Scout Apps

- Client only Application
- Just a single Maven Module
- Some generic Infrastructure Classes to bridge to Spring
- @Inject for Services (not BEANS.get)
- Optionally: BEANS.get can be used to access Spring Services
Demo «Add a Field»
Add a Field

Spring
⇒ Add «private String category; » to Task

Scout
⇒ Add CategoryColumn to AbstractTaskTablePage
⇒ Add Mapping in importTableRowData
⇒ Add CategoryField to TaskForm
⇒ Add Mappings in {import|export}FormFieldData
Wrap-up
Summary

What have we seen?

➡ Combination of **Spring Boot** and **Eclipse Scout**
  - JPA Persistence *(Spring Data)*
  - Business Model *(Spring)*
  - REST API *(Spring Web)*
  - UI *(Scout)*

Tasks for «Tasks»

➡ Complete Role & User Management
➡ Fine-tuning here and there
➡ Documentation on Github
Take-home Message

- Spring and Scout play along nicely
- Only a small set of good OSS Technologies needed
- Developer can concentrate on Business Value

GitHub

Thanks
@ZimMatthias
@EclipseScout
Thanks
and please, cast your Vote

-eclipsecon Europe 2016-

Evaluate the Sessions
Sign in and vote at eclipsecon.org

-1 0 +1