Whoops! Where did my architecture go?

Approaches to architecture management for Java and Spring applications

Oliver Gierke
Oliver Gierke
SpringSource Engineer
Spring Data

ogierke@vmware.com
olivergierke
www.olivergierke.de
Background

5 years of consulting
Lots of code reviews
Eoin Woods‘ talk on InfoQ
If you think architecture is expensive, try no architecture.
Macro VS. Micro Architecture
Macro VS. Micro Architecture
Sample Code

http://github.com/olivergierke/whoops-architecture
Roadmap

Divide and conquer
Of layers and slices
A plain Java based approach
Architecture 101
Know your dependencies
Explicit / Visible dependencies
Granularity

- Modules
- Layers
- Vertical slices
- Subsystems
Granularity

Java ARchive
Package
Class
Divide and conquer
Component
Component

Single unit to understand
Component

Single unit to change
Component

Scope of risk of change
Cost of separation
Definition and maintenance of dependencies
Smaller unit to understand
Reduced risk of change
Of layers and slices...
<table>
<thead>
<tr>
<th>Presentation</th>
<th>Service</th>
<th>Data Access</th>
</tr>
</thead>
</table>

The diagram shows a grid with labels for Presentation, Service, and Data Access. The grid's cells are connected with arrows, indicating a flow or relationship between the different sections.
<table>
<thead>
<tr>
<th></th>
<th>Account</th>
<th>Customer</th>
<th>Core</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Presentation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Service</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Data Access</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Layers

Well understood
Known to developers
Less important to business
Slices

Hardly understood
New to developers
Key for business req
How to implement an architecture inside a codebase?
Architecture VS. Codebase
How to implement an architecture inside a codebase?
How to implement an architecture inside a codebase?
How to maintain an architecture inside a codebase?
Code analysis

JDepend
Sonar
Demo
Sonargraph
Formerly known as SonarJ
Demo
A plain Java based approach
“How far can we get with **plain Java** means only?”
<table>
<thead>
<tr>
<th>Presentation</th>
<th>Account</th>
<th>Customer</th>
<th>Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Access</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Packages
....layer.slice
....slice.layer
....slice
....web.core
....service.core
....repository.core
core.web

core.service

core.repository
core
customer
account
"Why the f#$k should I even care?"
Does it make a difference?
Dependency management
"You only need to manage, what you can refer to..."
Layers first
Leaks slice internals
Lower layers visible to everyone
Slices first/only

Start with package per slice
Expose interfaces and domain types
Keep implementations private
Slices first/only

Encapsulates business module

Internals understood anyway
"Start with less packages and the least visibility possible..."
Demo
Take-aways

Know your dependencies
On every granularity
Start as strict as possible
Get lenient where necessary
Resources

Spring Data JPA @ GitHub
Sonargraph
Blogpost
Thanks & credits

Eoin Woods - Talk @ InfoQ

Uwe Friedrichsen - Slides @ Slideshare