SKILLS MATTER

Grails and Spring

2007

GRAILS eXCHANGE

http://www.grails-exchange.com | http://www.grails.org | http://skillsmatter.com

Dave Syer, Interface21

- What's new in Spring 2.5?
- Demo of MScript in Groovy
- Grails Spring integration features
- Spring Batch and Grails

Links:

http://www.springframework.org
http://www.interface21.com
http://blog.interface21.com
http://static.springframework.org/spring-batch

Dave Syer, Interface21

- **What's new in Spring 2.5?**
- Demo of MScript in Groovy
- Grails Spring integration features
- Spring Batch and Grails

What's New in Spring 2.5

- Annotation driven configuration (JSR250 + friends)
- MXBean support
- AspectJ extension: bean(name) pointcut
- JAX-WS (JSR224) support
- Spring Faces
- Spring Batch
- Spring Dynamic Modules for OSGi Service Platform

Annotations: JSR250 and friends

- Dependency injection
- Explicit wiring
- Component scanning
- Bean lifecycle
- Grails: useful for static middle tier configuration

Annotations: Example

component scanning

```
@Component("myService")
public class MyBean implements MyServiceInterface {

  @Resource(name="myDataSource")
  private DataSource dataSource;
  @Autowired
  public void injectServices(ServiceA a, ServiceB b) {...}

  @PostConstruct
  public void initialize() { ... }
  @PreDestroy
  public void shutdown() { ... }

}
```

explicit dependency injection

focused autowiring

bean lifecycle

MXBean Support

- Java 6 has native support for MXBeans (OpenMBean initiative)
- Spring MBeanExporter autdetects MXBeans
- Spring MBeanProxyFactoryBean also
  - Transparently wraps MXBeans
  - Converts arguments and return types (CompositeData etc.)

AspectJ Pointcut Extension

- Allow pointcut to target specific bean instance using AspectJ
- Natural extension point in AspectJ
- Example:

```
<aop:config>
    <aop:aspect>
        <aop:pointcut id="…"
            expression="bean(myBean) and execution(* *..start*(..))"/>
        …
    </aop:aspect>
</aop:config>
```

JAX-WS Support (JSR224)

- JAX-WS distributed with Java 6
- Built in HTTP Server (no need for Servlet container)
- Usual Spring remoting support in the form of
  - Server side Exporter
  - Client side ProxyFactoryBean

Spring Faces

- Currently released along with, but independent of, Spring Webflow
- Provides zero-configuration convenience for
  - transparent JSF integration to Spring-enabled projects
  - Spring Webflow integration if SWF is present
  - some client-side validation utilities
- Grails JSF plugin (Andreas Schmitt) is embryonic, but potentially very interesting

Spring Batch

- Framework for batch processing and transaction optimisation
- See later in this presentation for more details

Spring DMOSP

- Spring Dynamic Modules for OSGi™ Service Platorms
- Within a JVM: start, stop, upgrade, scale service components independently
- What a classloader could have been, but isn't
- Grails applications benefit the same way as any other: dynamic modifications to shared services

Dave Syer, Interface21

- What's new in Spring 2.5?
- **Demo of MScript in Groovy**
- Grails Spring integration features
- Spring Batch and Grails

MScript in Groovy

- MScript: dynamic scripting of remote MBeans
- Rob Harrop's original work in JRuby
- Very efficient and easy management scripting
- Operations, application management
- Developer productivity
- Dynamic classes (meta class data) used to present an MBean of any type
- Automatic conversion of OpenType attributes to Map

MScript Usage

- Three steps:
  1. create an MBeanServer - use one of the static factory methods platformMBeanServer() or connect(url)
  2. use it to create a new MBean
  3. use the MBean's attributes as if they were local Groovy properties, and the operations as if they were local methods.

MScript Example

```
> def bean =
        mscript.MBeanServer.platformMBeanServer()
        .newMBean("java.lang:type=Memory")

> println bean.properties
["HeapMemoryUsage":["committed":21483520, "init":0, "max":66650112,
"used":12472608], "NonHeapMemoryUsage":["committed":35454976,
"init":33751040, "max":121634816, "used":22441504],
"ObjectPendingFinalizationCount":0, "Verbose":false]
> println bean.Verbose
false
> bean.Verbose = true
```

dynamic method

attribute getter

CompositeData

attribute setter

MScript

- Code review and live demo (time permitting)

Dave Syer, Interface21

- What's new in Spring 2.5?
- Demo of MScript in Groovy
- **Grails Spring integration features**
- Spring Batch and Grails

Spring Integration in Grails

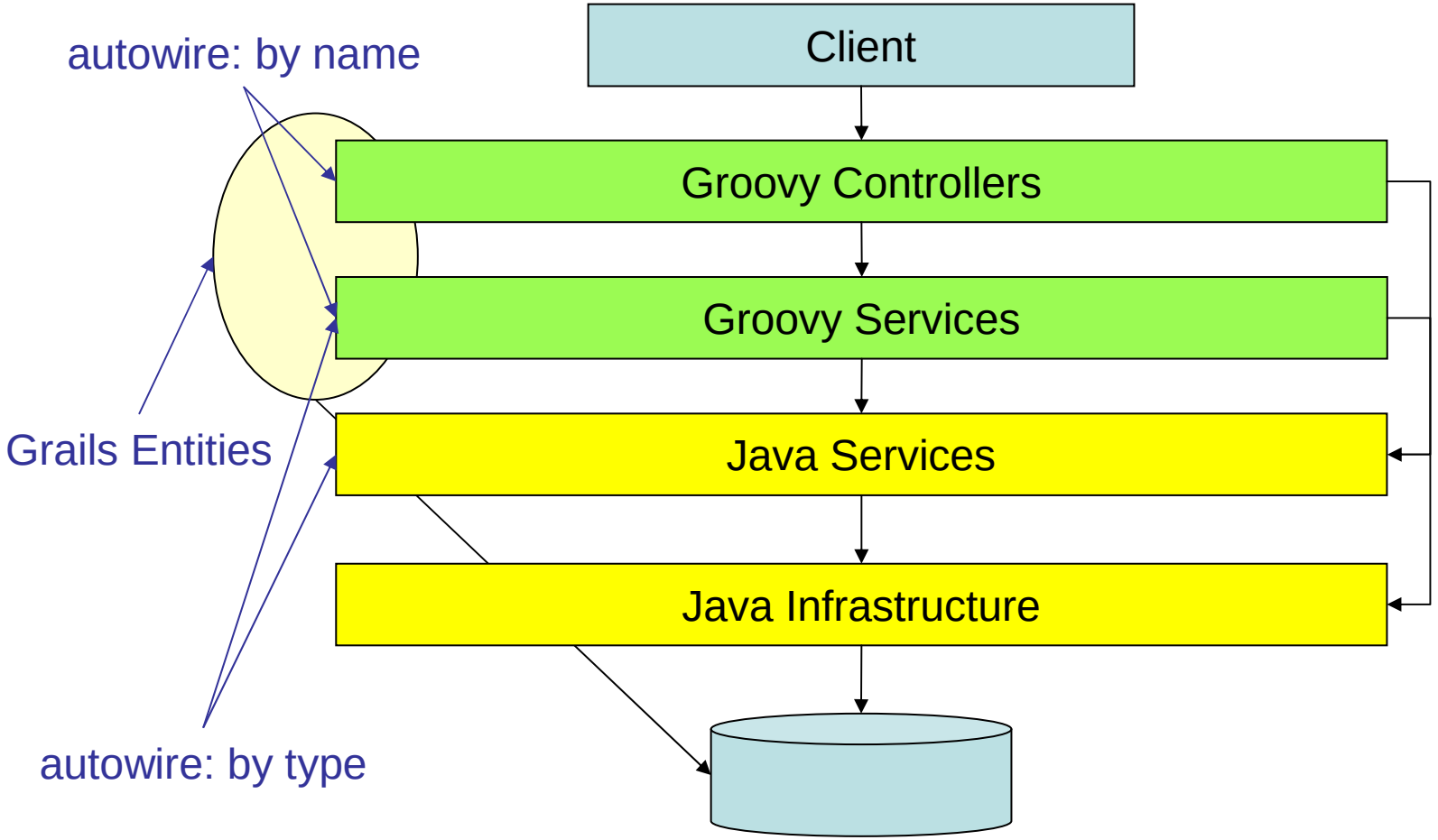
Grails is a heavy user of Spring, and uses some of the least often visited parts of the framework, and other Spring Portfolio products…

- Bootstrap
- Dynamic class loading and proxy creation
- Dynamic configuration
- Security
- Page navigation

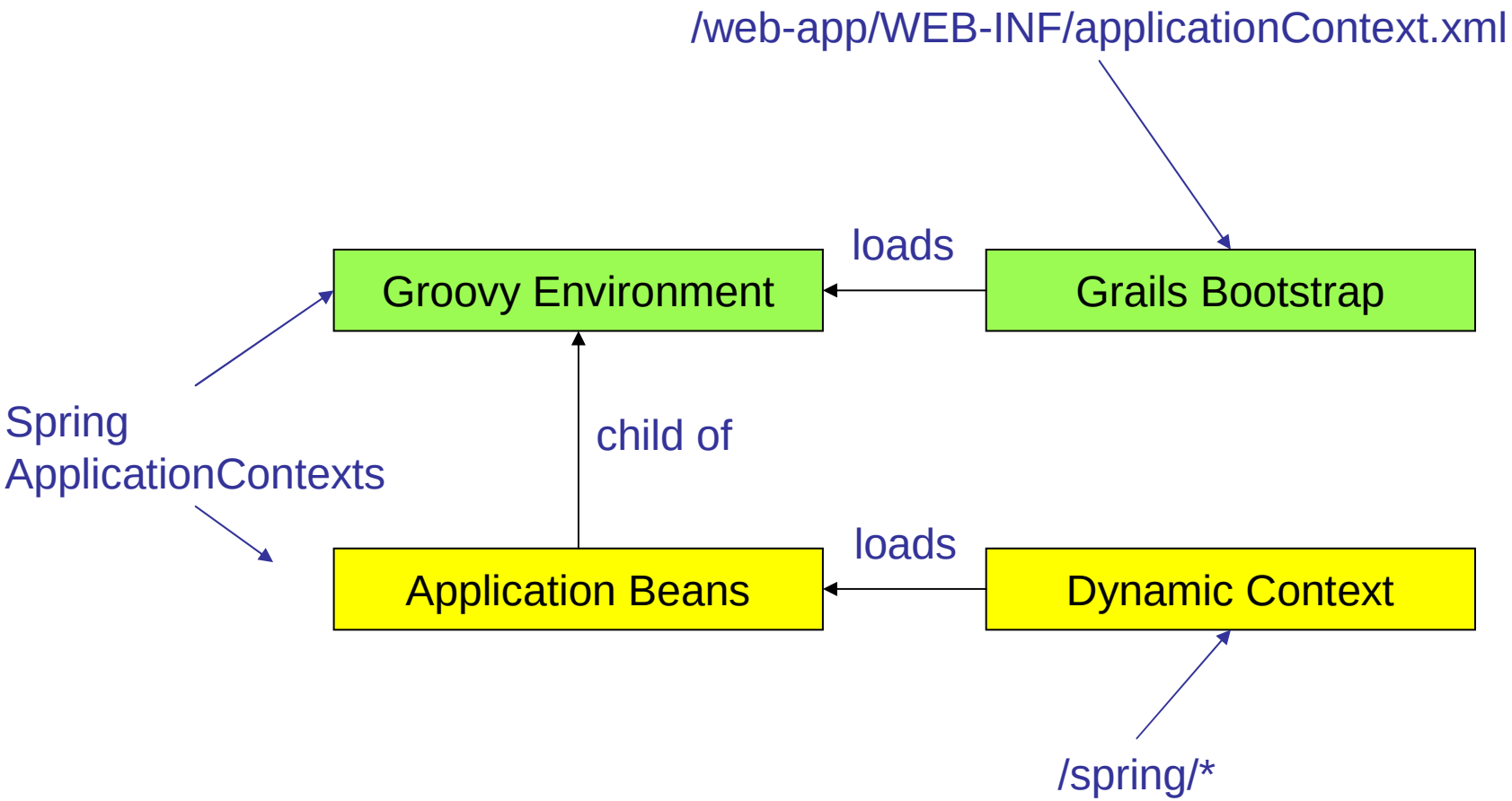
Grails + Spring highlights

- XML configuration of middle tier
- Groovy configuration of middle tier
- Navigation and state management - Spring Webflow DSL
- Namespace support
- Other things…

Configuration of Middle Tier



Grails Spring Context Hierarchy



XML Configuration

Reloaded dynamically

/spring/resources.xml

Import from app-specific local file

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="…" …>

   <import resource="grail-app.xml"/>
   <import resource="classpath*:spring-context.xml"/>

</beans>
```

Import all spring configuration from project jars

Groovy Configuration

Not reloaded dynamically (currently)

/spring/resources.groovy

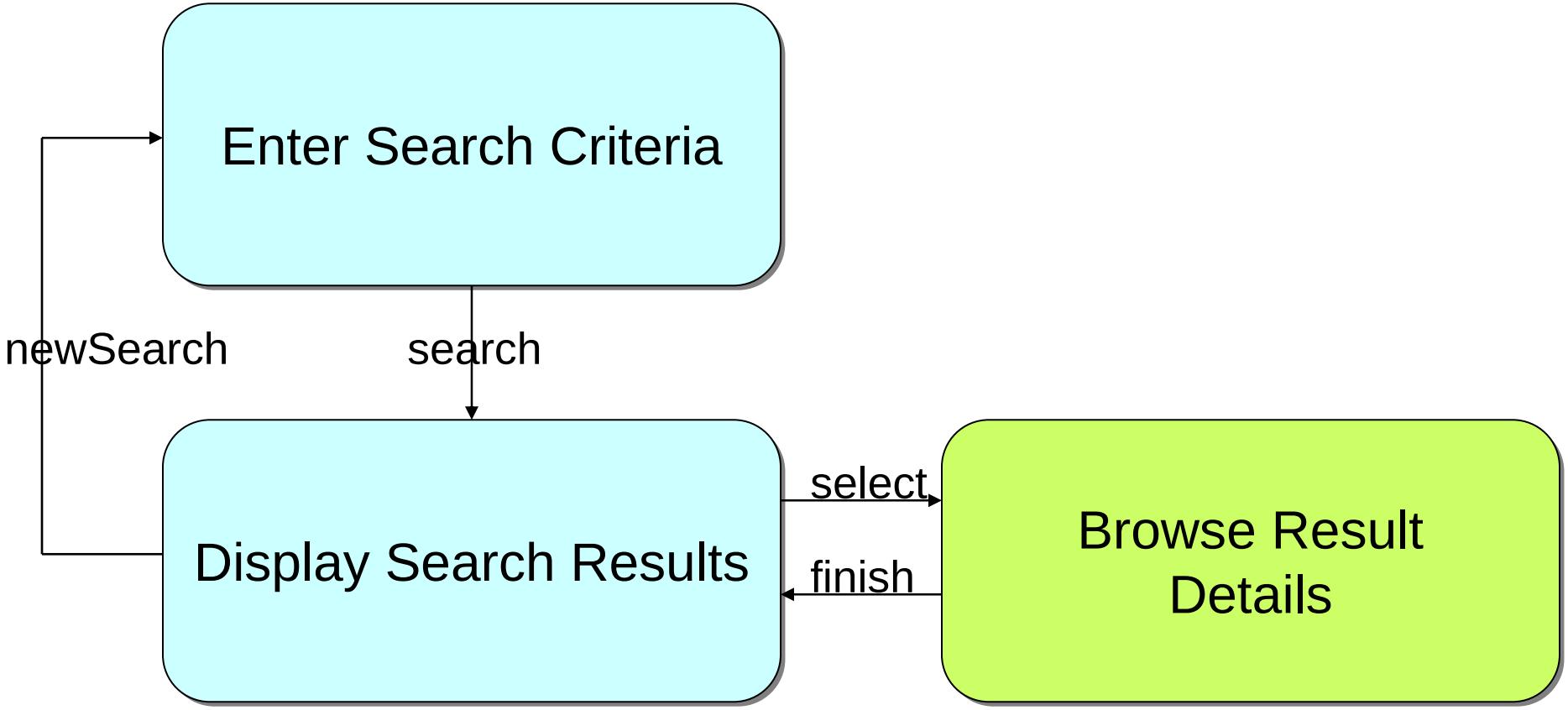
```
beans {
    warehouseJobConfiguration() { bean ->
        bean.parent = simpleJob
        steps = [ … ]
    }
}
```

No equivalent of <import resource="…"/> (currently).

Spring Webflow Integration

- Navigation
  - Finite state machine
- State-management:
  - Conversation scope -> ORM benefits
- Modularity
- Post-Redirect-Get
  - Browser button support
- Grails: Groovy DSL
  - Identify flow definitions with Grails controllers

Example Flow



Grails Controller

```
class ContactController {
    …
    def searchFlow = {
        enterCriteria {
            on("search").to "displayResults"
        }
        displayResults {
            on("newSearch").to "enterCriteria"
            on("select").to "browseDetail"
        }
        browseDetail {
            subflow(browseDetailsFlow)
            on("finish").to "displayResults"
        }
}
```

Naming convention: *Flow

Missing business logic:
call down to service layer

BeanBuilder: Namespace Support?

```
beans {

    dataSource(jee.jndiLookup) { jndiName = "jdbc/petstore" }

}
```

special argument

OR…

```
beans {

    jee.jndiLookup(id:'dataSource', jndiName:'jdbc/jpetstore')
}
```

special prefix

BeanBuilder: additional DSLs?

```
beans {

    aop.aspect() {
        pointcut.setOperation() {
            value = “execution(void *..set*(*))”
        }
        advice.before(setOperation()) {
            method = myBean.logMethod
        }
    }

    myBean(SomePlainOldJavaObject) {
        …
    }

}
```

an aspect

advice definition

a bean with a method called logMethod

Other Spring Integration Features

- What would the Grails community find the most useful?
- What is difficult or unnatural today?

Dave Syer, Interface21

- What's new in Spring 2.5?
- Demo of MScript in Groovy
- Grails Spring integration features
- **Spring Batch and Grails**

Spring Batch

- Batch jobs are part of most IT projects and currently no commercial or open source framework provides a robust, enterprise-scale solution/framework
- Batch processing is an Application Style for data processing pipelines (e.g. payment and settlement systems)
- The lack of a standard architecture has led many projects to create their own custom architecture at significant development and maintenance costs

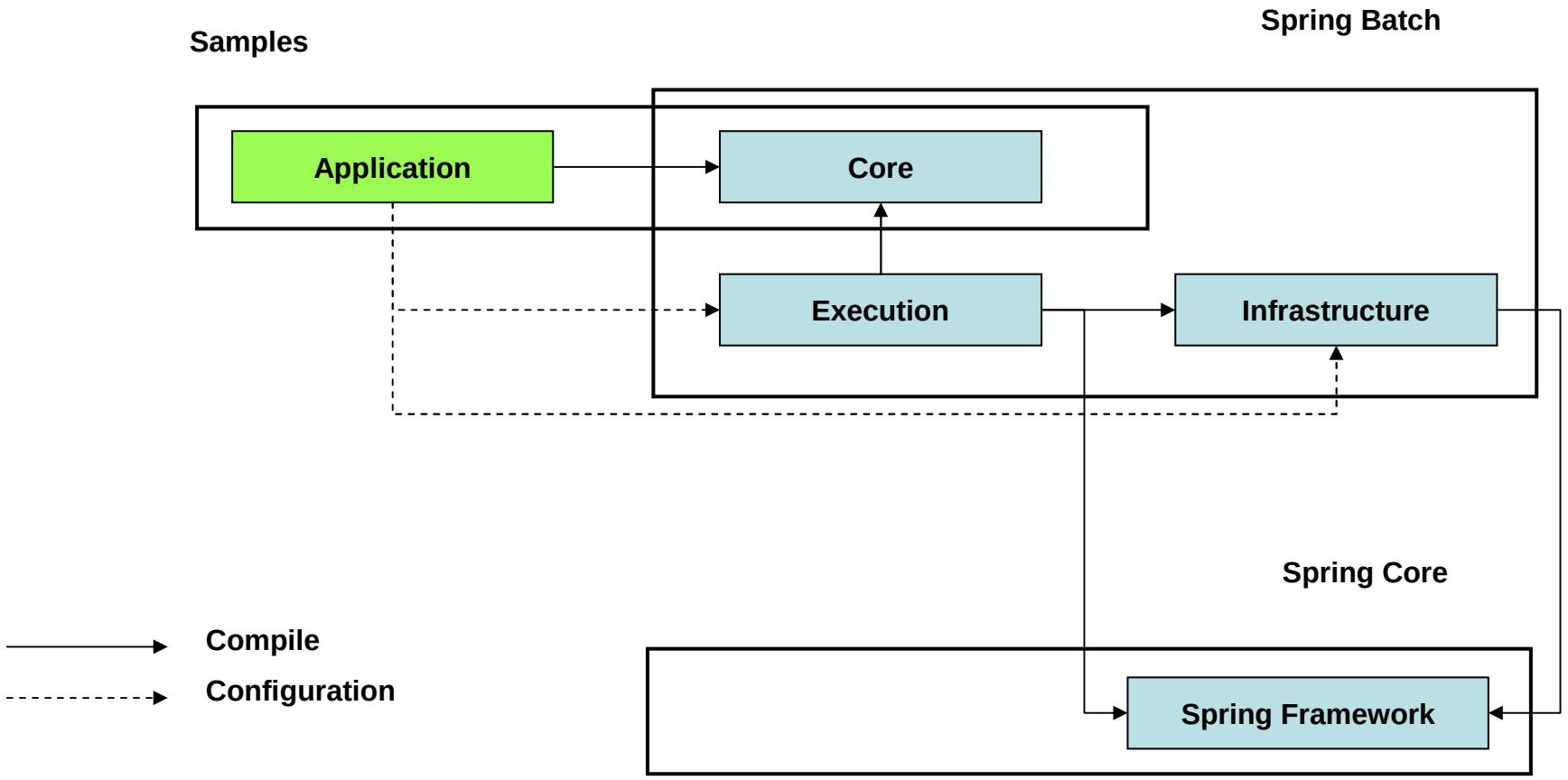
Spring Batch Design Objectives

- Clear separation of concerns – application developer concentrates on business logic
- Provide common, architecture layer services as interfaces
- Provide simple and default implementations of the layers that are easy to configure, customize, and extend
- Batch services should be easy to replace or extend, without impact to the infrastructure or application layers

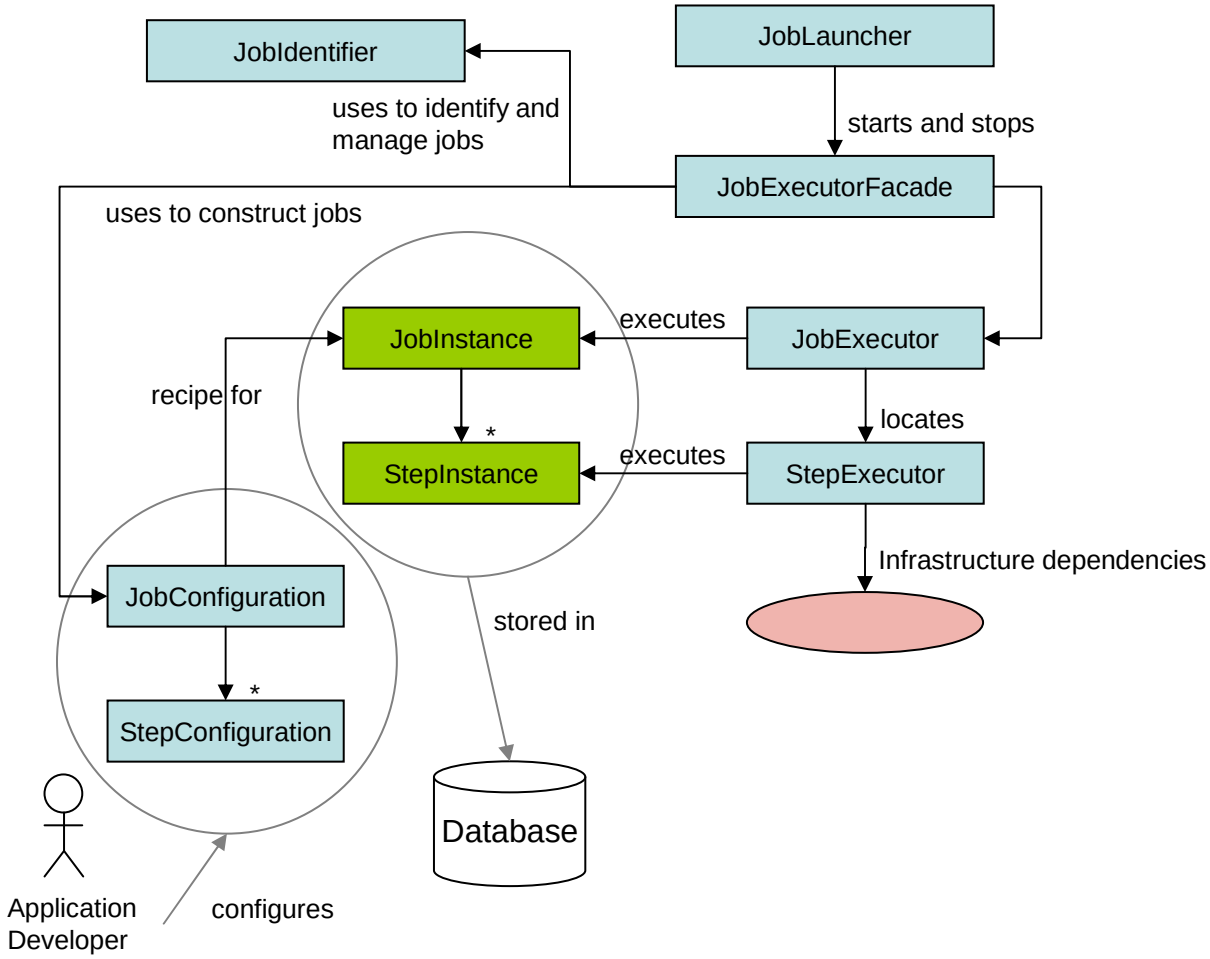
Spring Batch Scenarios

- Commit batch process periodically
- Concurrent batch processing: parallel processing of a jobs
- Manual or scheduled restart after failure
- Partial processing: skip records (e.g., on rollback)
- Sequential processing of dependent steps
- Staged, enterprise message-driven processing
- Whole-batch transaction for simple data models or small batch size
- Massively parallel batch processing

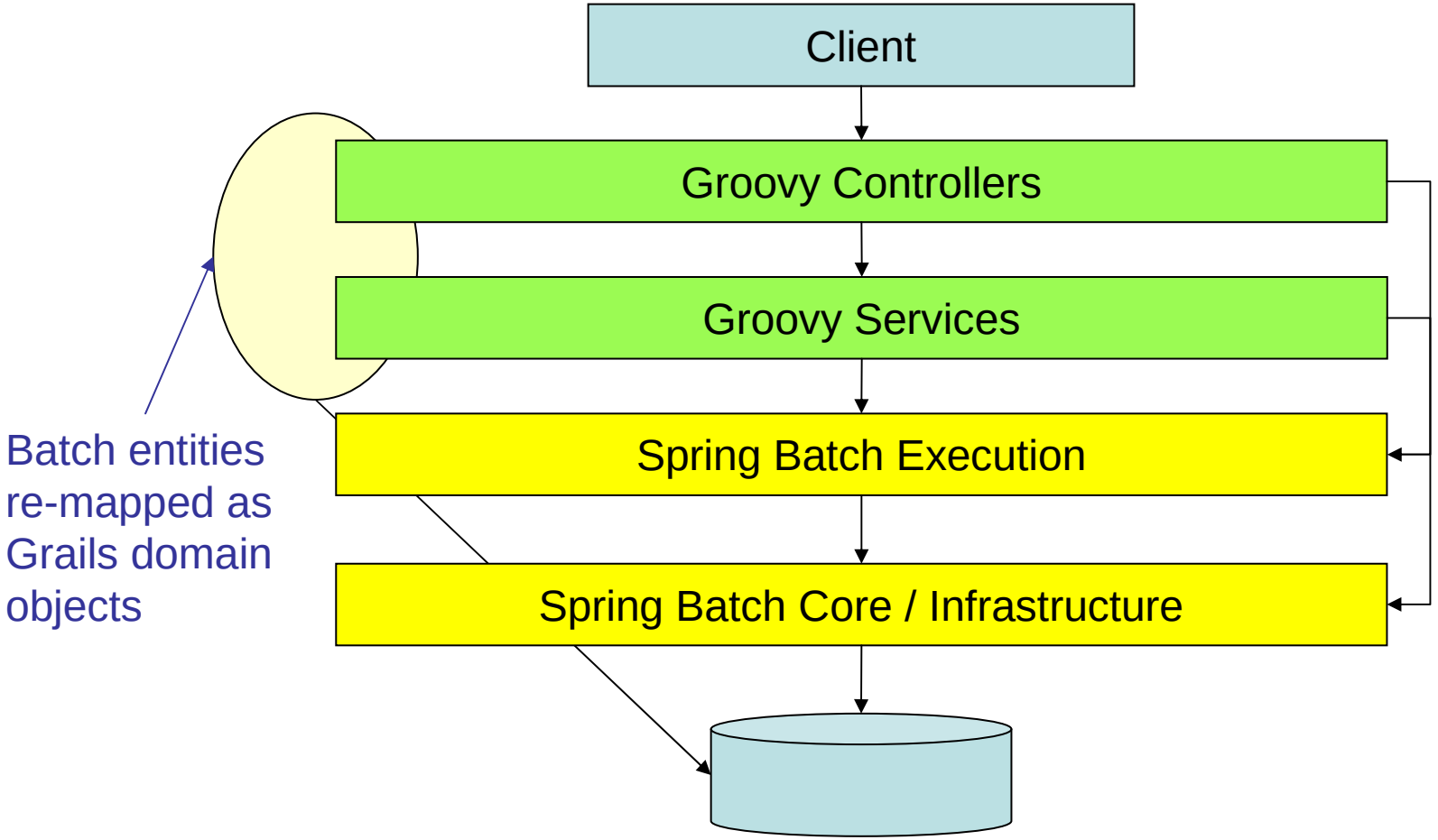
Runtime Dependencies



Spring Batch Domain Overview



Spring Batch Console



Spring Batch Console

- Code review and demo (time permitting)