Advanced Topics in Spring

CodeMash 2016

Pre-Compiler

Java: Beginner to Intermediate

David Lucas
Lucas Software Engineering, Inc.
www.lse.com
ddlucas@lse.com

LSE

Spring Framework: Core Annotations

DZone Reference Cards (very handy)

Spring Configuration

http://refcardz.dzone.com/refcardz/spring-configuration

Spring Annotations

http://refcardz.dzone.com/refcardz/spring-annotations

About: David Lucas

Lucas Software Engineering, Inc
(www.lse.com)

https://www.linkedin.com/in/ddlucas
@DavidDLucas
https://github.com/lseinc
ddlucas@lse.com

- Developing with Java since 1998
- Focus on Enterprise Solutions
  - Large Scale and Volume
  - DevOPS
- Worked in various industries:
  military, insurance, financial, manufacturing, utilities
- Provide software engineering, performance tuning, mentoring and training
- Master Craftsman @ Manifest Bootcamp, working with Manifest Solutions since 2004

Assumptions

- This session is for beginners wanting to expand their basic knowledge of Spring Framework
- Will NOT go into scripting or domain specific languages
- Will NOT dig into Spring Batch, Spring Big Data, Spring MVC or Spring XD
- Will discuss JPA, JMS, JTA, and WebServices (SOAP and some REST-ish)
- Will build examples using maven and STS
- Examples are for informational purposes
- Unit tests are NOT exhaustive

A Few Ground Rules

You will NOT hurt my feelings if you …

ask questions.

take your phone or text conversation outside.

leave because you are bored or need a break.

want me to change pace or speed.

participate.

Agenda

Setup Verification

Spring Overview

Labs

Spring Boot

Labs

What's Next ?

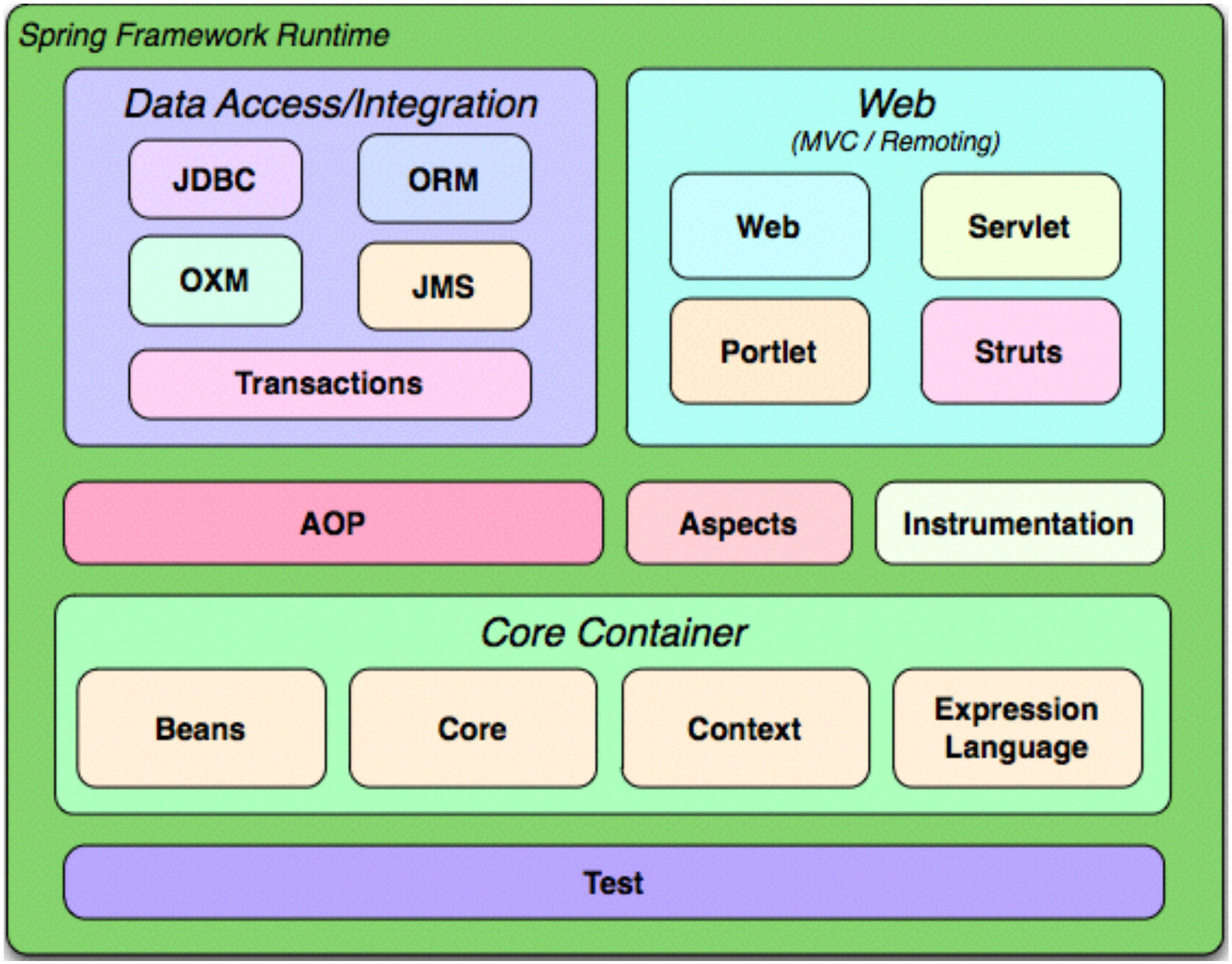
Setup Verification

- Java JDK 1.8
- Spring Tool Suite 3.7.2
- GitHub repo or Zip download
  - https://github.com/lseinc/spring-advanced
- import projects into STS
- Verify Setup (mvn clean install)


STOP

Spring Overview

- IOC / DI ?
- Containers: Why we don't like them?
- Frameworks: design patterns with default behaviors, life cycle
- Decoupled: easier to configure and test
- using since 2003, about ease of integration, replaced my own frameworks
- Driven new Java Standards and JEE profiles


Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
(MVC / Remoting)
Web
Servlet
Portlet
Struts
AOP
Aspects
Instrumentation
Core Container
Beans
Core
Context
Expression Language
Test

LAB 01

1) import project
2) mvn clean install
3) review some of Spring 3.x

Wait here until ready to move on…


STOP

3.x to 4.x

Wiring

<beans/>
<bean/>
<import />

Injections

<property/>
@Autowired
@Value

Component Beans

@Service
@Repository
@Component

Wiring

@Configuration
@Bean
@Import

Injections

@Autowired
@Value

Component Beans

@Service
@Repository
@Component

Spring Framework: 4.x

Bean Configurations

@Import
@Configuration
@ComponentScan
@PropertySource
@EnableAutoConfiguration
@EnableTransactionManagement

Spring Framework: Releases

Version	Released	End of Life
2.5	October 2008	January 2013
3.x	January 2010	December 2016
4.x	January 2014	2019 ?
5.x	4Q 2016 ?	2021 ?

Spring Framework: Core Annotations

DZone Reference Cards (very handy)

Spring Configuration

http://refcardz.dzone.com/refcardz/spring-configuration

Spring Annotations

http://refcardz.dzone.com/refcardz/spring-annotations

Spring Framework: Data

- Persistence
- JDBC
- Transactions

  -Local (JDBC)

  -Global (JTA)
- ORM

  -Hibernate

  -MyBatis

  -JPA (example: EclipseLink / Hibernate)

  -NoSQL

Spring Framework: Data

- JdbcTemplate uses DataSource provided by Spring Application Context
- DAO uses JdbcTemplate to execute JDBC type commands and manages closing of connections, statements, and result sets
- Results use a DataMapper to do manual mapping of each row to new entity object

Spring Framework: Data

- @Repository configures DAO with common exception translation
- JDBC Template (Spring JDBC)
- iBatis SqlMapClient support
- Hibernate Session API support
- JPA supported entity manager (EJB)
    - @PersistenceContext
    - @PersistenceUnit

Spring Framework: Data Transactions (TX)

- Spring Framework provides Transaction Management integration
- Leverages application container TMs:
  WebLogic, WebSphere, JBoss, etc
- Provides Local and Global support
  JTA: Java Transaction API (XA support)
- Spring provides consistent transaction management for multiple resources
  - database (JDBC, JPA)
  - messaging (JMS)
  - RMI / IIOP

Simple Spring Configuration

```
@Configuration
@EnableTransactionManagement
@ImportResource(locations="classpath:spring-minimal.xml")
public class SpringConfiguration {
    /* put special creation and wiring here */
}
```

LAB 02-A

1) import project
2) add SpringConfiguration class
2) mvn clean install
3) review some of Spring 4.x

Wait here until ready to move on…


STOP

Spring Framework: Data Transactions

- Simple DataSource TransactionManager
- Not Two Phase Commit
- Only Supports JDBC transactions
- Local Transaction Support (not Global)

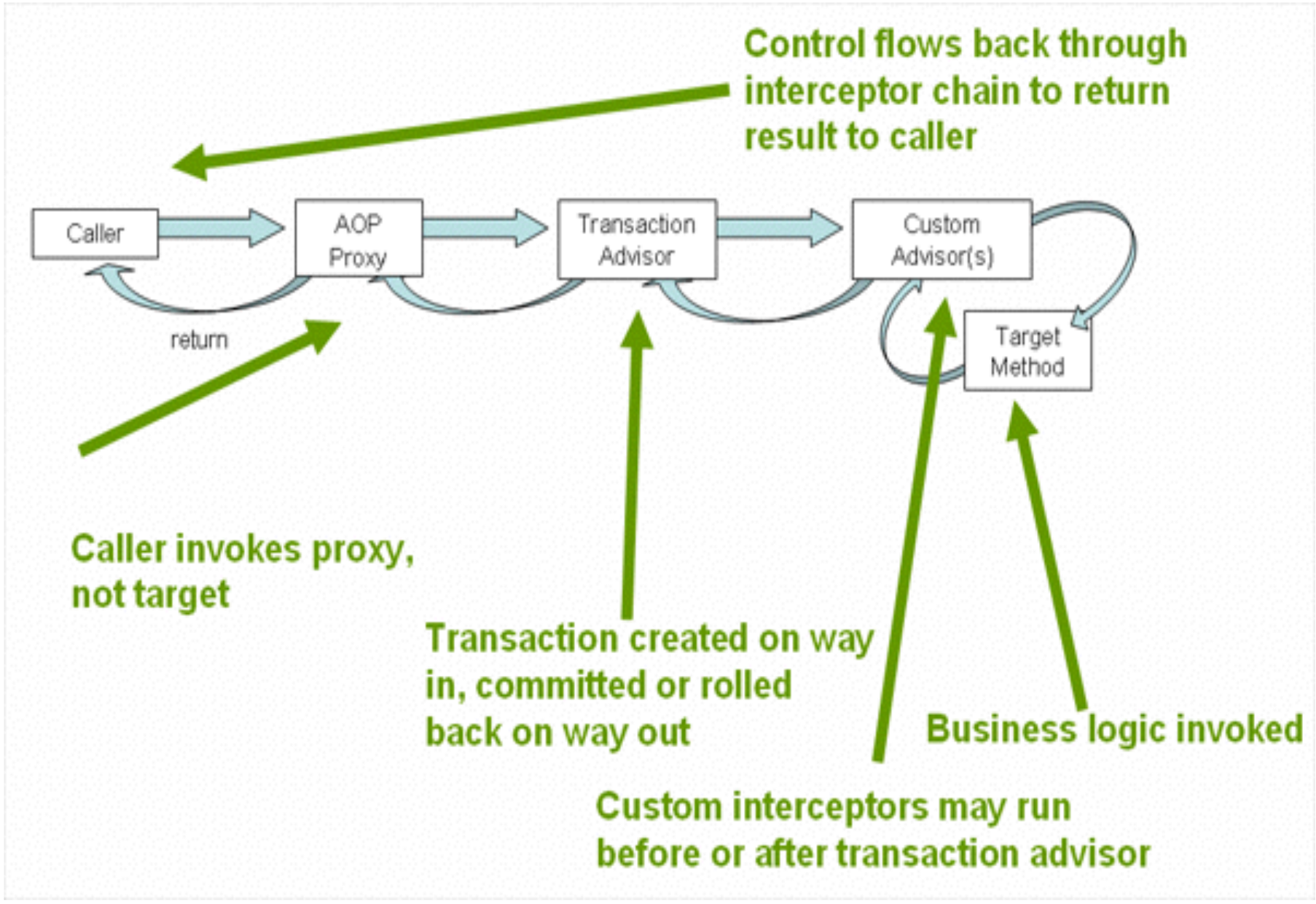
```
<!-- a PlatformTransactionManager is still required -->

<bean id="transactionManager"
  class="org.springframework.jdbc.datasource.DataSourceTransaction
  Manager">
  <!-- (this dependency is defined somewhere else) -->
  <property name="dataSource" ref="dataSource"/>

</bean>
```

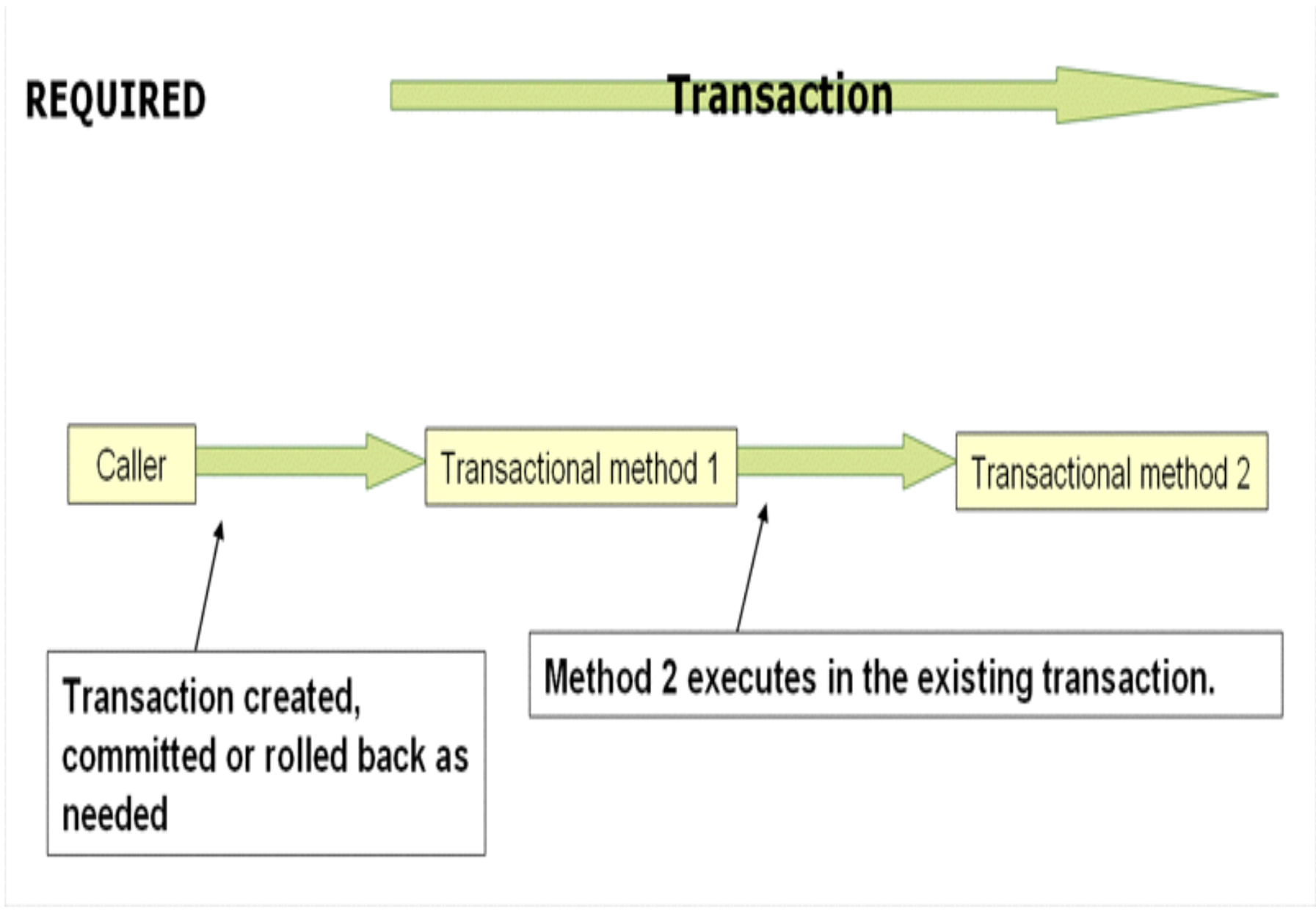
Spring Framework: Data Transactions

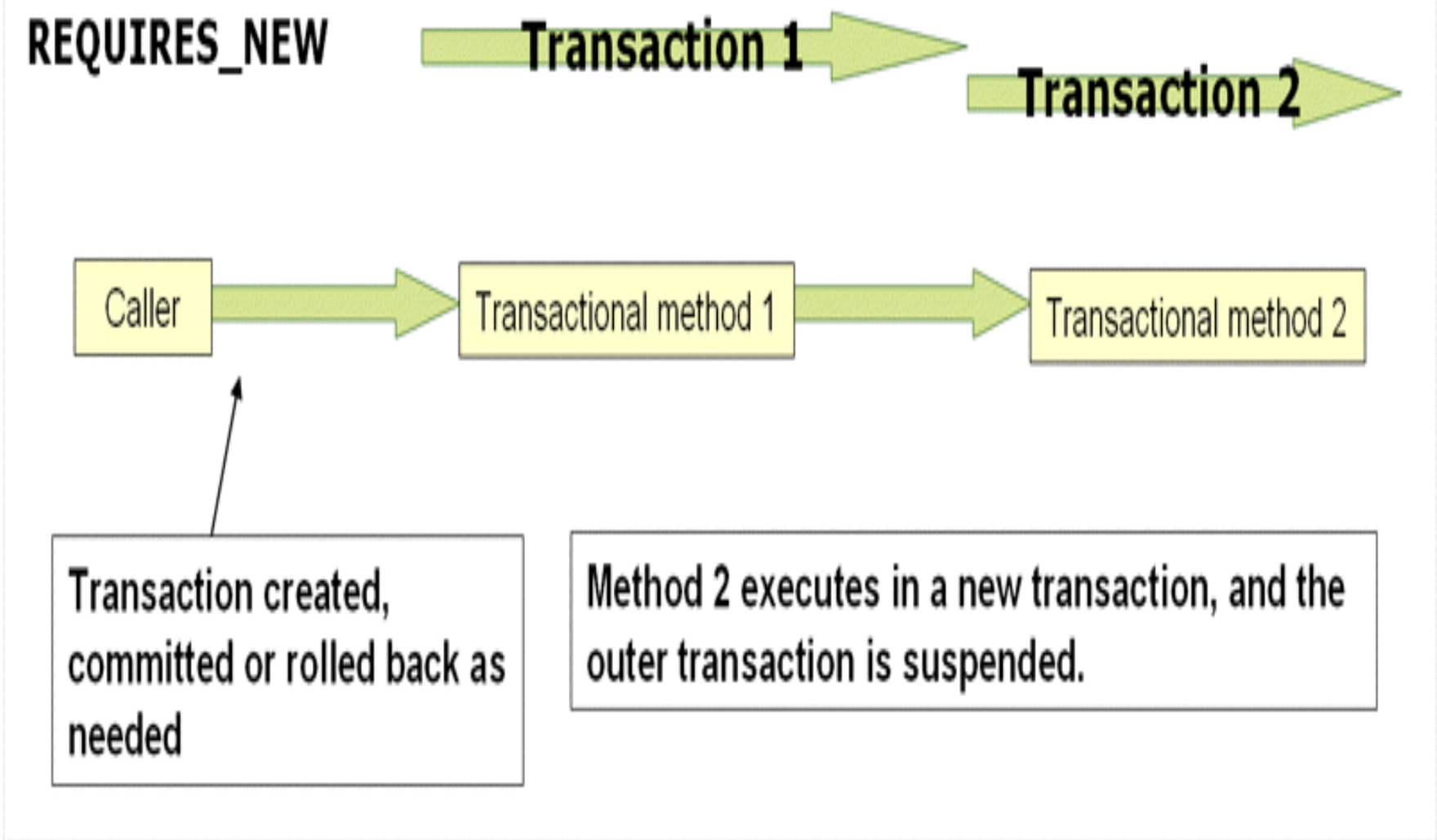
@Transactional annotation wraps methods with transaction control using transaction manager calls to begin / commit / rollback

- isolation and propagation level config
- read-only optimization option
- timeout (max time before rollback)
- applied at class or method level

```
@Transactional(readOnly = false, propagation =
  Propagation.REQUIRES_NEW)
  public void updateFoo(Foo foo) {
       // do something
  }
```


Control flows back through interceptor chain to return result to caller
Caller
AOP Proxy
Transaction Advisor
Custom Advisor(s)
Target Method
return
Caller invokes proxy, not target
Transaction created on way in, committed or rolled back on way out
Business logic invoked
Custom interceptors may run before or after transaction advisor


REQUIRED
Transaction
Caller
Transactional method 1
Transactional method 2
Transaction created, committed or rolled back as needed
Method 2 executes in the existing transaction.


REQUIRES_NEW
Transaction 1
Transaction 2
Caller
Transactional method 1
Transactional method 2
Transaction created, committed or rolled back as needed
Method 2 executes in a new transaction, and the outer transaction is suspended.

Spring Framework: Data Transactions

- Unit Testing can utilize a third party Transaction Manager
- Atomikos is open source and very stable
  - Transaction Essentials (Apache License 2)
  - DataSource wrappers for XA and non-XA connections
  - Support for JMS

Spring Framework: Data Transactions

Example XML Config

```
<bean id="dataSource" name="dataSource,datasource"
        class="com.atomikos.jdbc.nonxa.AtomikosNonXADataSourceBean"
        lazy-init="true" init-method="init" destroy-method="close" >
        <property name="uniqueResourceName" value="NONXADBMS" />
        <property name="driverClassName" value="${jdbc.driverClassName}" />
        <property name="url" value="${jdbc.url}" />
        <property name="user" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
        <property name="readOnly" value="false" />
        <property name="poolSize" value="1" />
        <property name="maxPoolSize" value="4" />
        <property name="minPoolSize" value="0" />
        <property name="testQuery" value="select 1 from dual" />
</bean>

<bean id="atomikosTransactionManager"
      class="com.atomikos.icatch.jta.UserTransactionManager"
      init-method="init" destroy-method="close">
      <property name="forceShutdown" value="true"/>
      <property name="transactionTimeout" value="${transaction.timeout}" />
</bean>
```

Spring Framework: Data Transactions

Example XML Config (cont)

```
   <bean id="atomikosUserTransaction"
          class="com.atomikos.icatch.jta.UserTransactionImp">
         <property name="transactionTimeout" value="${transaction.timeout}"
   />
   </bean>

<!-- Configure the Spring framework to use JTA transactions from Atomikos
   -->
<bean id="transactionManager"
       class="org.springframework.transaction.jta.JtaTransactionManager">
      <property name="transactionManager">
            <ref bean="atomikosTransactionManager" />
      </property>
      <property name="userTransaction">
            <ref bean="atomikosUserTransaction" />
      </property>
</bean>


<!-- enable the configuration of transactional behavior based on
annotations -->
<tx:annotation-driven transaction-manager="transactionManager"/>
```

LAB 02-B

1) modify Bank to create/update a transfer method
2) create an exception between credit and debit
3) what happened and how to fix it ?

Wait here until ready to move on…


STOP

Spring Framework: JPA Annotations

DZone Reference Cards (very handy)

Getting Started with JPA
http://refcardz.dzone.com/refcardz/getting-started-with-jpa

What's new with JPA 2.0 (2.1 is latest, but useful)
http://refcardz.dzone.com/refcardz/whats-new-jpa-20

Spring Framework: JPA

Major Players: Hibernate / EclipseLink

- Spring provides hooks for each
- Hibernate has more history and maturity
- EclipseLink (OpenSource) came from TopLink, Oracle owned
- Both allow entity integration

Spring Framework: JPA

- Add some dependencies
  - spring-data-commons
  - spring-data-jpa
  - hibernate-jpa-2.1-api
  - hibernate-entitymanager
  - cglib

Spring Framework: JPA

- @EnableJpaRepositories(basePackages = "com.lse.spring.example")
- @Entity / @Table(name="ACCOUNT")
- @Id / @Column(name = "ACCT_NUMBER")
- @Transactional(dontRollbackOn = {EmptyResultDataAccessException.class})

LAB 03

1) copy lab 2 over to lab 3 named example-atm-jpa
2) edit pom.xml, change artifactId / name
   add dependencies
3) modify Account as new Entity
4) add AccountDaoJPA JPARepository

```
public interface AccountDaoJPA extends AccountDao,
                JpaRepository<Account,String> {
}
```

5) Profiles ???

Wait here until ready to move on…

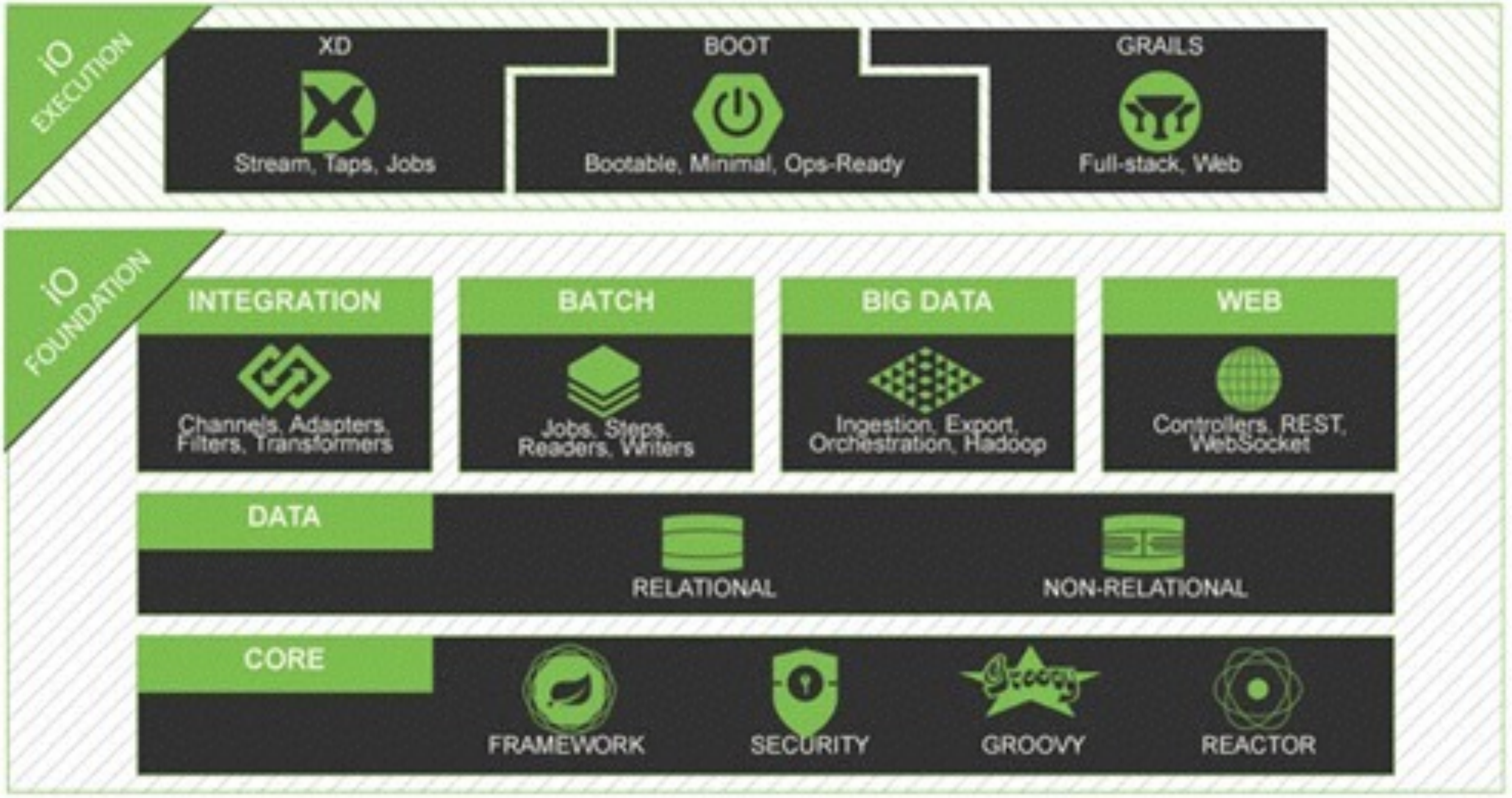

STOP

Spring Boot

Builds upon Spring Framework success

http://docs.spring.io/spring-boot/docs/1.3.x/reference/htmlsingle/

- Create stand-alone, production-grade applications that just run
  Opinioned view of Spring, best of bread setup
- Auto-configurations
- Easy to get running !!!
- Alot of free features by using starter framework and templates
- NO XML Required !!!


IO
EXECUTION
XD
Stream, Taps, Jobs
BOOT
Bootable, Minimal, Ops-Ready
GRAILS
Full-stack, Web
IO
FOUNDATION
INTEGRATION
Channels, Adapters,
Filters, Transformers
BATCH
Jobs, Steps,
Readers, Writers
BIG DATA
Ingestion, Export,
Orchestration, Hadoop
WEB
Controllers, REST,
WebSocket
DATA
RELATIONAL
NON-RELATIONAL
CORE
FRAMEWORK
SECURITY
GROOVY
REACTOR

WONKA'S

GOLDEN TICKET

GREETINGS TO YOU, THE LUCKY FINDER OF THIS GOLDEN TICKET FROM MR. WILLY WONKA!...

PRESENT THIS TICKET AT THE FACTORY GATES AT TEN O'CLOCK IN THE MORNING OF THE FIRST DAY OF OCTOBER AND DO NOT BE LATE. YOU MAY BRING WITH YOU ONE MEMBER OF YOUR OWN FAMILY... AND ONLY ONE... BUT NO ONE ELSE.....

In your wildest dreams you could not imagine the marvelous SURPRISES that await YOU!

Spring Starter / InitializR

Provides easy boiler plates for starting apps

Support from STS
File->New->Spring Starter Project

http://projects.spring.io/spring-boot

Easy getting started with InitializR
select types of options (Web, JMS, JPA, etc)
generates boiler plate zip to download
http://start.spring.io

Spring Starter / InitializR

WAIT, LETS LOOK AT IT…

Spring Tool Suite Starter Project

http://start.spring.io

[Generates a project based on requirements]

Spring Framework: Initializr

Let's ~~build~~ generate a project together…

- maven project, spring boot 1.3.1
- groupId: com.lse.spring.example
- artifactId: example-audit-jms
- actuator + actuator docs
- artemis (JMS ActiveMQ)
- data jpa
- jta atomikos
- Remote Shell CRASH
- WS (Web Services)

Spring Framework: JMS

- Now that you downloaded it, what next ?
- Copy the zip to lab-04-jms or leave in STS workspace if done via eclipse
- Oh, and now lets talk about JMS

JMS Client

JMS Listener (MDB)

Spring Framework: Core Annotations

DZone Reference Cards (very handy)

Spring Configuration

http://refcardz.dzone.com/refcardz/spring-configuration

Spring Annotations

http://refcardz.dzone.com/refcardz/spring-annotations

Spring Framework: JMS

Players: ActiveMQ / RabbitMQ / JMS++ more

- Spring provides hooks for major configs
- Others provide own integration artifact for spring
  - ActiveMQ has an embedded mode, not alone
  - JMS implementations should work with JTA

Spring Framework: JMS

- Let's build a simple audit notification listener
- Send a simple String message to a listener so it can be logged using sl4j logging

```
private static final Logger log =
LoggerFactory.getLogger(<theclass>.class);
```

Spring Framework: JMS

//JEE Message Drive Bean like using a “listener” pattern

```
@Component
public class AuditMessageListener {
   public static final String MESSAGE_QUEUE = "example.audit.queue";

    @JmsListener(destination = MESSAGE_QUEUE)
    public void processMessage(String auditMessage) {
      //do stuff
    }
}
```

Spring Framework: JMS

```
//Spring Client Template (again  :-) )

@Component
public class AuditClient  {
   @Autowired
    private JmsMessagingTemplate jmsMessagingTemplate;

    @Autowired
    @Qualifier("auditQueue")
    private Queue queue;

   public void audit(String msg) {
      this.jmsMessagingTemplate.convertAndSend(this.queue, msg);
    }
}
```

Spring Framework: JMS

```
//Spring Config

@Configuration
public class JMSConfig {
    @Bean
    public ActiveMQConnectionFactory jmsConnectionFactory() {
        ActiveMQConnectionFactory amqFactory =
            new ActiveMQConnectionFactory(
                "vm://localhost?broker.persistent=false&broker.useJmx=false");
        amqFactory.setSendTimeout(30000); //example of a factory setting
        amqFactory.setMaxThreadPoolSize(10);
        return amqFactory;
    }
    @Bean
    public Queue auditQueue() {
        return new ActiveMQQueue(AuditMessageListener.MESSAGE_QUEUE);
    }
}
```

Spring Framework: JMS

```
//Spring Data Config for example
@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(basePackages = "com.lse.spring.example")
public class JPAConfig {
    @Value("${spring.datasource.driverClassName}")    String dsDriverClassName;
    @Value(“${spring.datasource.url}") String dsUrl;
    @Value(“${spring.datasource.username}") String dsUsername;
    @Value(“${spring.datasource.password}") String dsPassword;

    @Bean(initMethod="init", destroyMethod="close")    DataSource dataSource() {
        System.setProperty("derby.system.home","./logs/derby-data");
        AtomikosNonXADataSourceBean dataSource =
                        new AtomikosNonXADataSourceBean();
          dataSource.setUniqueResourceName(
                        "JTAResource-"+System.currentTimeMillis());
        dataSource.setDriverClassName(dsDriverClassName);
        dataSource.setUrl(dsUrl);
        dataSource.setUser(dsUsername);
        dataSource.setPassword(dsPassword);
        dataSource.setMaxPoolSize(4);
        dataSource.setTestQuery("VALUES(1)");
        return dataSource;
    }
}
```

LAB 04-A

1) after unzipping starter project…
2) copy over src/main/resources folder from setup
3) add datasource setup in a JPAConfig
4) add a JMS Listener and Client
5) Create a simple Unit Test to send a request without exception

Wait here until ready to move on…


STOP

Spring Boot: Starter

```
//Spring Boot Starter
@Component
public class AuditStartup implements CommandLineRunner {
   @Override
    public void run(String... args) throws Exception {
        StringBuilder message = new StringBuilder();
        if (args!=null && args.length>0) {
            for (String arg : args) {
                message.append(arg);
            }
        }
       client.audit(message.toString());
    }
}
```

LAB 04-B

1) now let’s create our own auto command line starter
2) Have it execute an audit notification that we started

Wait here until ready to move on…


STOP

Spring Actuator

Let’s take a tour of Actuator…

DEMO

http://localhost:8080/docs

http://localhost:8080/info

http://localhost:8080/health

http://localhost:8080/metrics

http://localhost:8080/mappings

http://localhost:8080/configprops

http://localhost:8080/autoconfig

http://localhost:8080/beans

http://localhost:8080/info

http://localhost:8080/dump

http://localhost:8080/env

http://localhost:8080/trace

LAB 04-C

1) now let's create our own Audit entity and apply what we learned with JPARepository and store the message when the listener receives it
2) Audit should have a UUID for the @Id
3) Audit should have a Date createdTs
4) Audit should have a String message
5) Create a JPARepository using Audit as the templated class and String as the Id

Wait here until ready to move on…


STOP

Spring Framework: JAX-WS

- Spring supports multiple JAX-WS implementations
- CXF is most popular, supports jax-ws and jax-rs implements (XML + JSON)
- We will create a web service using standard JEE annotations

Spring Framework: JAX-WS

Need an interface to define the method and types

```
@WebService
@SOAPBinding(style= Style.DOCUMENT, use= Use.LITERAL)
public interface AuditWS  {
    @WebMethod
    String audit(@WebParam(name = "message") String
message);
}
```

Spring Framework: JAX-WS

Need an interface to define the method and types

```
@WebService
@SOAPBinding(style= Style.DOCUMENT, use= Use.LITERAL)
public interface AuditWS  {
    @WebMethod
    String audit(@WebParam(name = "message") String
message);
}
```

Spring Framework: JAX-WS

```
@EnableWs
@Configuration
public class WSConfig {
   @Bean
    public ServletRegistrationBean servletRegistrationBean(ApplicationContext
context) {
        CXFServlet jaxwsServlet = new CXFServlet();
        jaxwsServlet.setBus(springBus());
        return new ServletRegistrationBean(jaxwsServlet, "/jaxws/*");
    }

    @Bean(name = "cxf")
    public SpringBus springBus() {
        SpringBus bus = new SpringBus();
        return bus;
    }
```

Spring Framework: JAX-WS

```
@Bean public AuditWS auditWSService() {
    AuditWSImpl svc = new AuditWSImpl();
    return svc;
}
 @Bean  public Endpoint auditWSEndpoint() {
    EndpointImpl endpoint =
        new EndpointImpl(springBus(),
        auditWSService());
    endpoint.publish("/auditWS");
    return endpoint;
}
```

Spring Framework: JAX-WS

```
@Bean  Jaxb2Marshaller auditWSMarshaller() {
    Jaxb2Marshaller marshaller = new Jaxb2Marshaller();
    marshaller.setContextPath("com.lse.spring.example.ws");
    return marshaller;
}
@Bean WebServiceTemplate auditWSClient() {
    WebServiceTemplate wsc = new WebServiceTemplate();
        wsc.setMarshaller(auditWSMarshaller());
        wsc.setUnmarshaller(auditWSMarshaller());
        return wsc;
    }
}
```

Spring Framework: JAX-WS

For the client we need some maven dependencies:

```
<dependency>
    <groupId>org.apache.cxf</groupId>
    <artifactId>cxf-bundle</artifactId>
    <version>3.0.0-milestone2</version>
</dependency>
<dependency>
    <groupId>javax.xml</groupId>
    <artifactId>jaxws-api</artifactId>
    <version>2.0</version>
</dependency>
```

Spring Framework: JAX-WS

For the client we need some maven magic:

```
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <configuration>
     <executable>true</executable>
     <layout>ZIP</layout>
  </configuration>
</plugin>
```

Spring Framework: JAX-WS

More maven magic:

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>build-helper-maven-plugin</artifactId>
    <version>1.7</version>
    <executions>
      <execution>
        <id>add-source</id>
        <phase>generate-sources</phase>
        <goals>
          <goal>add-source</goal>
        </goals>
        <configuration>
          <sources>
            <source>${project.build.directory}/generated-sources</source>
          </sources>
        </configuration>
      </execution>
    </executions>
</plugin>
```

Spring Framework: JAX-WS

More maven magic:

```
<plugin>
    <groupId>org.jvnet.jaxb2.maven2</groupId>
    <artifactId>maven-jaxb2-plugin</artifactId>
    <version>0.11.0</version>
    <executions>
      <execution>
        <goals>
          <goal>generate</goal>
        </goals>
      </execution>
    </executions>
    <configuration>
      <verbose>true</verbose>
      <debug>true</debug>
      <!-- <packageName></packageName>  -->
      <generateDirectory>${project.build.directory}/generated-sources</generateDirectory>
      <schemaDirectory>${project.basedir}/src/test/resources/wsdl</schemaDirectory>
      <schemaIncludes>
        <schemaInclude>**/*.wsdl</schemaInclude>
        <schemaInclude>**/*.xsd</schemaInclude>
      </schemaIncludes>
      <bindingIncludes>
        <bindingInclude>**/*.xjb</bindingInclude>
      </bindingIncludes>
   </configuration>
</plugin>
```

LAB 04-D

1) now lets expose our JMS Client via a Web Service
2) we need to first create the WSConfig
3) next we create the AuditWSService
4) now we need to add some maven magic
   to generate JAXB objects for our unit test
5) Our Integration Unit test will need to use the
   WebServiceTemplate

Wait here until ready to move on…


STOP

CraSH

- Spring allows us to integrate an SSH server embedded in our Spring Boot app
- Allows for setting of username / password values or use generated password stored in logs (reset each restart)
- Allows for viewing internals of JVM
- Allows creation of our own commands

CraSH

DEMO

CraSH

- Crash allows for scripting (like groovy) to customize the experience

```
package crash.commands
import com.lse.spring.example.jms.AuditClient
import org.crsh.cli.*
import org.springframework.beans.factory.BeanFactory
import org.springframework.beans.factory.annotation.Configurable

@Configurable
@Usage("provide a way send an audit message event")
class audit {
    @Command  Object main(
            @Man ("audit \"<message>\" ")
            @Usage("the audit message to send, use quotes for whitespace.")
            @Argument String message) {
        if (message == null) {
            message = "<empty>";
        }
        AuditClient client = fetchClient();
        client.audit(message);
        return "sent";
    }
//…
```

CraSH

```
  AuditClient fetchClient() {
      BeanFactory beanFactory =
            context.attributes['spring.beanfactory']
      try {
         return beanFactory.getBean(AuditClient)
      } catch (Exception) {
         return null;
      }
   }
}
```


LAB 04-E

1) add a crash command
2) log in via ssh
3) verify command runs

Wait here until ready to move on…


STOP

Spring Framework: Initializr

Let's ~~build~~ generate a project together…

- maven project, spring boot 1.3.1
- groupId: com.lse.spring.example
- artifactId: example-atm-app
- actuator + actuator docs
- data jpa
- jta atomikos
- Remote Shell CRASH
- Jersey (jax-rs)

Spring Framework: JAX-RS

- Now that you downloaded it, what next ?
- Copy the zip to lab-05-jaxrs or leave in STS workspace if done via eclipse
- Oh, and now lets talk about JAX-RS

REST Annotations

DZone Reference Cards (very handy)

REST

https://dzone.com/storage/assets/4019-rc129_010d-rest_0.pdf

Spring Framework: JAX-RS

- Spring supports multiple JAX-RS implementations (CXF & Jersey)
- Jersey is most popular from my point of view
- We will create a web service using standard JEE annotations

Spring Framework: JAX-WS

```
@Component
@ApplicationPath("/rest")
public class ATMRestApplication extends ResourceConfig
{
    private static final Logger log =
LoggerFactory.getLogger(ATMRestApplication.class);

    public ATMRestApplication() {
        register(ATMRestService.class);
        register(JacksonFeature.class);
        log.info("+++ ATMRestApplication started...");
    }
}
```

Spring Framework: JAX-RS

```
@Path("/atm")
@Component
@Transactional
public class ATMRestService {
    private static final Logger log = LoggerFactory.getLogger(ATMRestService.class);

    @Inject
    AccountDao dao;

    @GET
    @Path("/health")
    @Produces(MediaType.APPLICATION_JSON)
    public Response health() {
        Response response = null;
        try {
            Map<String,String> info = new HashMap<>();
            info.put("account_count",String.valueOf(dao.count())  );
            response = Response.ok(info).build();
        }
        catch(Throwable t) {
            log.error("balance error: {}",t.getMessage(),t);
            response = Response.status(Status.INTERNAL_SERVER_ERROR).build();
        }
        return response;
    }
}
```

LAB 05

1) lets implement an ATMRestService that will…
2) return a health check that it is running
3) return the balance amount
4) return an account
5) make sure you copy setup source for this one too
   (example unit test awaits you)
5) Our Integration Unit test will need to use the
   RestServiceTemplate to make a call

HINT: use the artifact example-atm-sf4 built earlier

Wait here until ready to move on…


STOP

Spring Boot
and
Micro-Services

DEMO Spring Cloud


STOP

Advanced Spring 2016 Summary

- Spring Framework has a lot of nice features you don't have to build !
- Spring Boot makes it easier to create opinioned solutions, but tuning will need to improve (lots of extras that you might not need, cpu/memory use)
- There is so much we just scratched the surface
- Read more at spring.io guides and documents
- IT may not be for everyone, but if you want to focus on business solutions, you may want to give Spring and Spring Boot a try !
- Watch out for Docker and Spring Boot microservices

Spring Framework: Resources

http://spring.io/docs

http://spring.io/guides

http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/

http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework

http://www.javabeat.net/tutorials/8-spring-framework-beginners-tutorial.html

Books:

- Spring In Action, Walls
- Pro Spring, Harrop and Machacek

Spring Framework: Resources

http://refcardz.dzone.com/refcardz/spring-configuration
http://refcardz.dzone.com/refcardz/spring-annotations
http://refcardz.dzone.com/refcardz/core-spring-data
http://refcardz.dzone.com/refcardz/eclipse-tools-spring
http://refcardz.dzone.com/refcardz/spring-web-flow
http://refcardz.dzone.com/refcardz/expression-based-authorization
http://refcardz.dzone.com/refcardz/getting-started-with-jpa
http://refcardz.dzone.com/refcardz/whats-new-jpa-20
http://refcardz.dzone.com/refcardz/eclipselink-jpa
http://refcardz.dzone.com/refcardz/spring-integration
http://refcardz.dzone.com/refcardz/spring-batch-refcard
http://refcardz.dzone.com/refcardz/mockito

Spring Framework: Resources

https://dzone.com/storage/assets/487673-rc024-corejava.pdf
https://dzone.com/storage/assets/572027-rc221-docker.pdf
https://dzone.com/storage/assets/293353-rc215-microservices.pdf
https://dzone.com/storage/assets/3012-rc003-eclipse_online.pdf
https://dzone.com/storage/assets/4506-rc094-020d-git_2.pdf
https://dzone.com/storage/assets/4019-rc129_010d-rest_0.pdf
https://dzone.com/articles/debug-and-maintain-your-spring-boot-app

https://dzone.com/storage/assets/4388-rc050-010d-scrum_2.pdf

https://dzone.com/storage/assets/4595-rc008-designpatterns_online.pdf
https://dzone.com/storage/assets/413450-rc218-cdw-jenkins-workflow.pdf


STOP

Thank You !!!

Any feedback appreciated to help
the next sojourner !!!

I can’t get any better if you don’t tell me
what you like and dislike. :-)

David Lucas
Lucas Software Engineering, Inc (www.lse.com)


https://www.linkedin.com/in/ddlucas
@DavidDLucas
https://github.com/lseinc
ddlucas@lse.com