

The new JEE world

Why are light frameworks better than traditional, heavy-weight JEE approaches

- Java is the best platform
- What is going to happen in the Java world
- The future of complex systems

- Develop using the Spring portfolio
- Run the application in light container

Why Java

- It is a mature language, with modern compilers and managed runtime
- Java's ecosystem is full of best quality of open source frameworks
- The Java platform is often the starting point of innovation (AOP, Groovy, Scala, ...)
- Sometimes, there is too much choice, but only the best practice brings good results

What is happening in Java

- Java's approach to complex applications has evolved from heavy approaches towards lean frameworks
- The aim is to reduce the accidental complexity to allow the developers to concentrate on the essential complexity
- Top engineers contribute their time & expertise to open source projects
 - The Spring Framework & Spring Portfolio
 - The Apache projects
 - The Eclipse projects

What is happening in Java

- Care about & use open source:
 - It is peer-reviewed, high quality code
 - It is open for anyone to see & use
 - It is free from licensing restrictions
 - It is very serious development, more so than in most commercial environments

The future systems

- Open source brought quality & clarity to the source code
- We now need to use it to create open information systems
- Use open standards to represent information
- Expose their functionality to other systems
- Do not hide essential complexity

The future systems

- In technical speak, open information systems
 - Typically provide RESTful APIs and use open communication methods (JSON, XML)
 - Use some type of messaging infrastructure to scale and be cloud-ready
 - Rigorously enforce the tier boundaries
 - Are scalable, integrate well, are easy to secure & review & verify, and usually bring excellent quality

Develop using Spring

- The Spring Framework sets out to eradicate the accidental complexity in JEE development
 - It makes it easy to follow the best development practices
 - It integrates almost all facets of JEE application development
 - It is extremely well-documented
 - The learning curve can be steep

What is in Spring?

- At its core, Spring Framework is a dependency injection container
- Spring uses XML and annotation-backed configuration to wire up the application
(Compare with Grails, Ruby on Rails or Django that use naming conventions to discover the purpose of the code)
- On top of the DI core, Spring Framework includes support for data access, services, web and integration tiers

Spring Portfolio

- The Spring Portfolio includes projects that:
- Abstract out message-driven systems (Spring Integration)
- Secure the systems (declarative standards-driven Spring Security)
- Simplify creation of resilient batch processes (Spring Batch)
- And many others

The Plumbing

- Code to create the components
- Code to look up the created components
- Code to implement typical functionality
 - Using JDBC / ORM
 - Transactional behaviour, security
 - Web tier
- Integration with typical enterprise components

The effects of plumbing

- It distract us from writing code that makes us money
- It increases complexity of our code (NIH syndrome)
- It introduces bugs
- It makes us abandon best practices because we just have to get it to work, we have no time to write pretty code

How does Spring help?

- It takes care of instantiating the components
- It wires up the components together
- It simplifies the typical tedious tasks by providing implementation of design patterns that cover
- JDBC, Hibernate, web, security, messaging, integration, open APIs
- It brings aspect-oriented programming, dynamic languages and many more

It's too good to be true...

- The price we pay is reduced feeling of control over your application
- There is some performance penalty at startup and at runtime, especially when using dynamic AOP
- Finally, the learning curve can be steep

Run in light containers

- Use closed source runtime platforms if and only if you absolutely require the features they bring!
- Avoid products that attempt to hide the essential complexity (ESBs!)
- Favour the lightest possible container; *compose, do not inherit*: favour combination of small tools that perform their tasks exceptionally well over complex system that includes all components

Run in light containers

- In our experience
Tomcat 6 or 7 running application written using
Spring Framework 3, Spring Integration 2, Spring Security 3,
Spring WebFlow 2.1
- far outperforms
 - the very same application in WebSphere, WebLogic and similar
 - application that implements the same features not using the Spring Portfolio

The evidence

```
try {
    Class.forName("org.hsqldb.jdbcDriver");
} catch (ClassNotFoundException e) {
    throw new RuntimeException(e);
}
Connection connection;
try {
    connection = DriverManager.getConnection("jdbc:hsqldb:mem:ps", "sa", "");
} catch (SQLException e) {
    throw new RuntimeException(e);
}
try {
    try {
        final PreparedStatement createTableStatement = connection.prepareStatement(
            "create table user (id int primary key, name varchar(200))");
        createTableStatement.execute();
        createTableStatement.close();

        final PreparedStatement insertStatement = connection.prepareStatement(
            "insert into user values (1, 'Jan')");
        int count = insertStatement.executeUpdate();
        insertStatement.close();

        System.out.println(count);
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    try {
        connection.commit();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
} finally {
    try {
        connection.close();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

```
ApplicationContext ac = new ClassPathXmlApplicationContext("/META-INF/spring/module-context.xml");
JdbcTemplate template = ac.getBean(JdbcTemplate.class);

template.execute("create table user (id int primary key, name varchar(200))");
System.out.println(template.update("insert into user values (1, 'Jan')"));
```

The evidence

BPB Service Node

Call statistics

Hits

Count 302
Average call time 179 μ s
Median call time 116 μ s
Maximum call time 3 ms
Minimum call time 69 μ s

Misses

Count 100
Average call time 8 ms
Median call time 4 ms
Maximum call time 230 ms
Minimum call time 3 ms

Hits, Misses and Unadvised

Count 402
Average call time 2 ms
Median call time 142 μ s
Maximum call time 230 ms
Minimum call time 69 μ s

Segment statistics

Segment contact

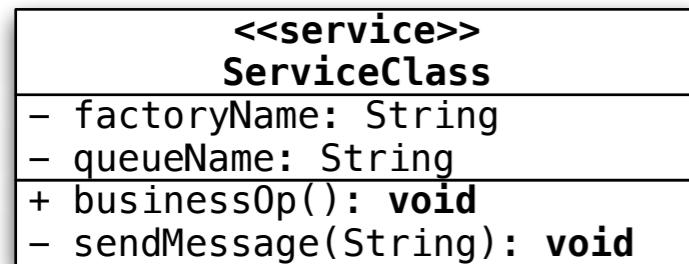
The evidence

```
try {
    QueueConnectionFactory qConnectionFactory = ServiceLocatorManager
        .lookupQueueConnectionFactory("qfactory");
    Queue queue = ServiceLocatorManager.lookupQueue("queue");
    QueueConnection qConnection = qConnectionFactory.createQueueConnection();
    QueueSession qSession = qConnection.createQueueSession(
        false, Session.AUTO_ACKNOWLEDGE);
    QueueSender sender = qSession.createSender(queue);

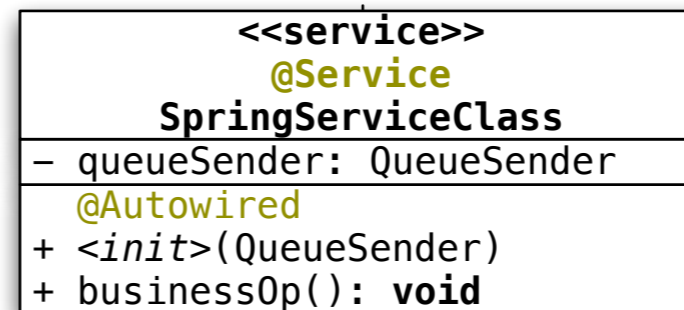
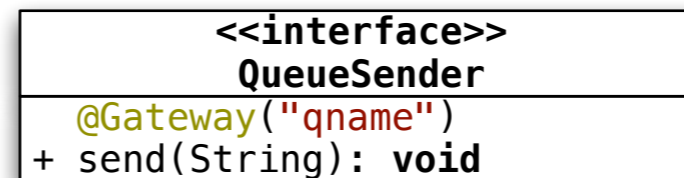
    TextMessage message = qSession.createTextMessage();
    message.setText("some really long string");
    sender.send(message);

    sender.close();
    qSession.close();
    qConnection.close();
} catch (JMSException ex) {
    throw new RuntimeException(ex);
}
```

```
ApplicationContext ac = new ClassPathXmlApplicationContext("/META-INF/spring/module-context.xml");
QueueSender sender = ac.getBean(QueueSender.class);
sender.send("some really long string");
```



Business code and
the long JMS code



Just the business code

The evidence

- I will leave unit and integration testing of the `ServiceClass` and the `SpringServiceClass` as exercise for your team... They will:
 - Find it difficult to reliably test the `ServiceClass` outside the container
 - Discover that it is difficult to simulate unusual scenarios (various `JMSEExceptions`, errors in message transport, etc)
 - Be able to unit test the `SpringServiceClass` completely, including all exceptional situations
- What if you later on discover that you need to support another messaging infrastructure? Will you re-implement the `ServiceClass`? Will you subclass it?

The evidence

- Spring Portfolio isolates you from the tedious details of your infrastructure
- Your programmers can concentrate on the important code
- They can use the latest approaches: OOP, AOP, DDD, BDD, ...enough buzzwords to fill the whole slide!
- The metadata you provide at development time gives the runtime more information about the purpose of the code—you can get better error reporting and common-sense behaviour
- It allows you to bring in other rapid development tools and methods

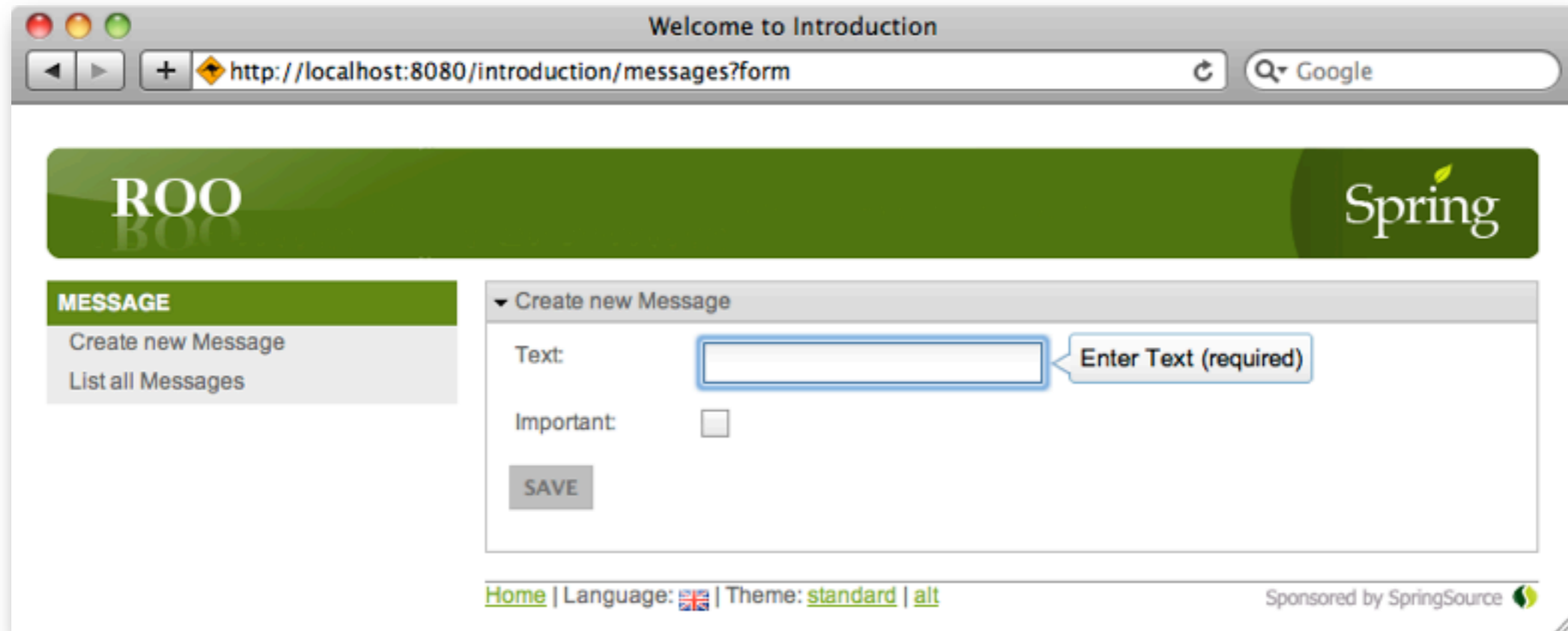
The evidence

```
~/Minefield/mg$ cat mg.roo
project --topLevelPackage net.cakesolutions.pr.introduction
persistence setup --provider HIBERNATE --database HYPERSONIC_IN_MEMORY
entity --class ~.domain.Message
field string --fieldName text --notNull
field boolean --fieldName important --notNull
controller all --package ~.web
quit
```

```
~/Minefield/mg$ roo script mg.roo
```

...

```
~/Minefield/mg$ mvn tomcat:run
```



Welcome to Introduction

http://localhost:8080/introduction/messages?form

ROO

Spring

MESSAGE

Create new Message


List all Messages


Create new Message

Text: Enter Text (required)

Important:

SAVE

Home | Language:  | Theme: [standard](#) | [alt](#)

Sponsored by SpringSource 

Spring Support

- We can offer complex support package:
 - Developer training
 - Mentoring, peer code reviews
 - Architectural assistance
 - Agile management support
- We will never accept mediocrity; we will always push you to create the best possible code: our services are excellent, but we expect excellence back.

Training & mentoring

- Cake Solutions and Skills Matter co-operate to deliver exceptional training courses covering:
 - The Spring Framework (DI, AOP, JDBC, Hibernate, transactions, EL, Spring MVC, REST & others)
 - Spring WS (server implementation of XML WS endpoints and clients)
 - Coming soon: Spring Roo, Hibernate
- We are happy to design custom training & mentoring package for you. Our clients had custom courses covering:
 - Modern & light Java
 - Spring Integration (EIP & scalable messaging)
 - Spring Batch (reliable, large-scale data processing; including large-scale Hibernate data access)

Questions