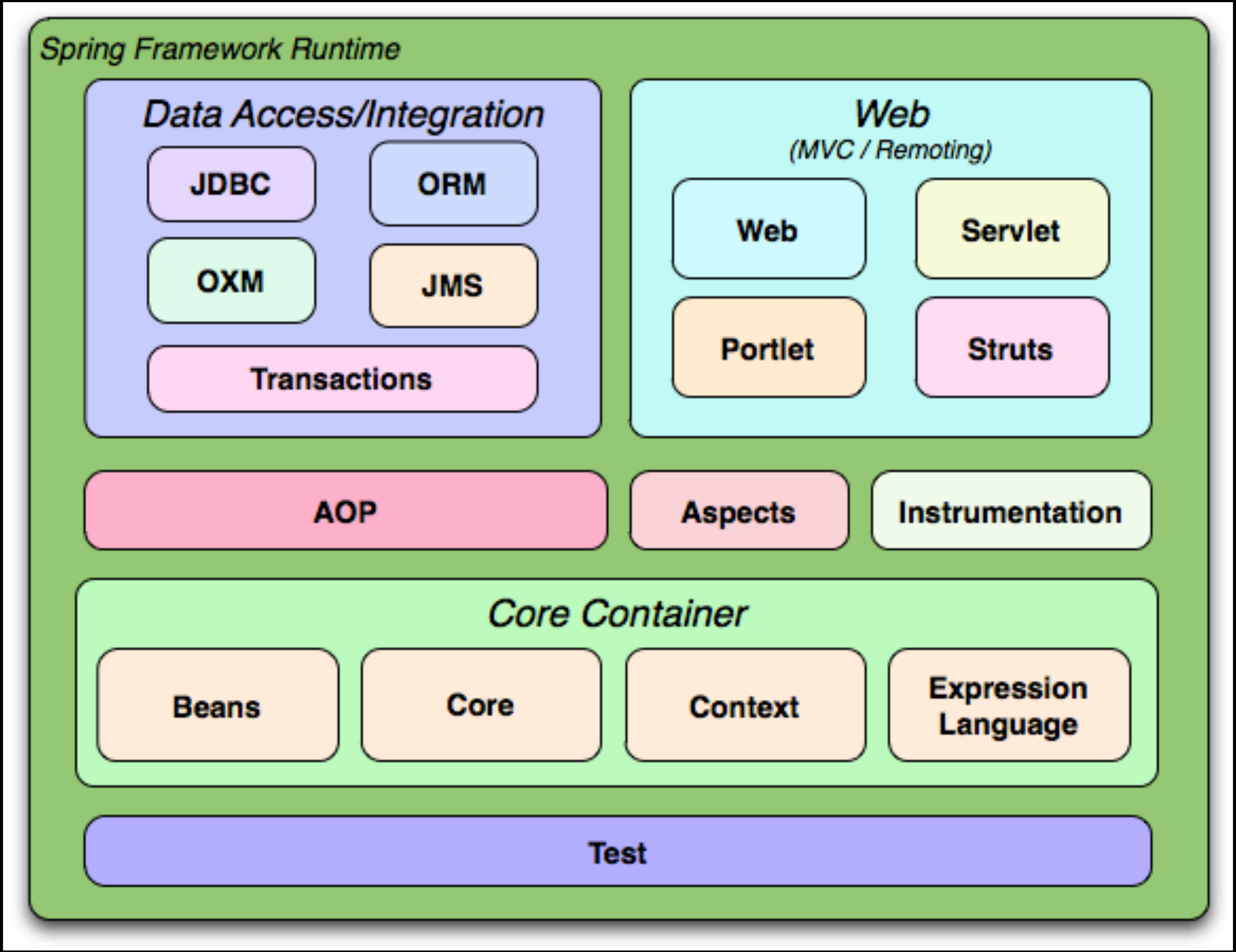
Spring Web MVC

Ji Hao

Features

- Flexible dependency injection with XML and annotation-based configuration styles
- Advanced support for aspect-oriented programming with proxy-based and AspectJ-based variants
- Support for declarative transactions, declarative caching, declarative validation, and declarative formatting
- Powerful abstractions for working with common Java EE specifications such as JDBC, JPA, JTA and JMS
- First-class support for common open source frameworks such as Hibernate and Quartz
- A flexible web framework for building RESTful MVC applications and service endpoints
- Rich testing facilities for unit tests as well as for integration tests

Modules



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

BeanFactory & ApplicationContext

- The BeanFactory provides the underlying basis for Spring's IoC functionality but it is only used directly in integration with other third-party frameworks and is now largely historical in nature for most users of Spring.

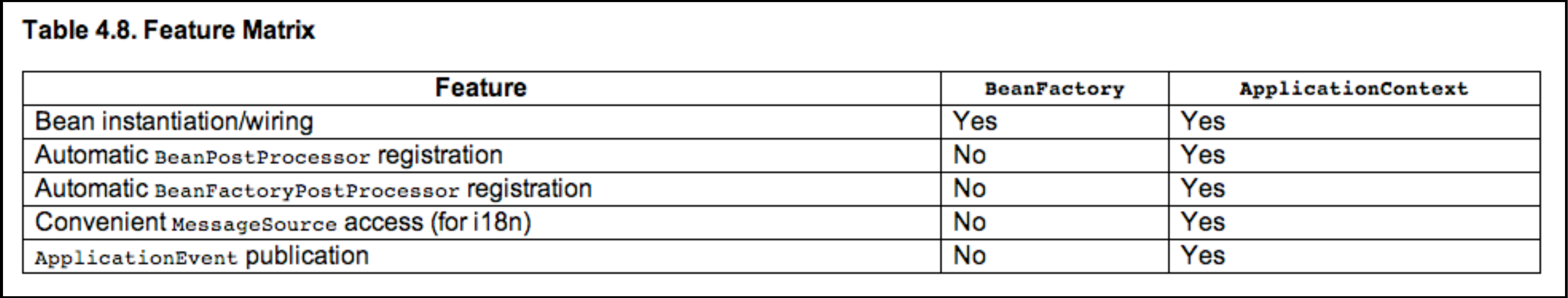


Table 4.8. Feature Matrix

Feature	`BeanFactory`	`ApplicationContext`
Bean instantiation/wiring	Yes	Yes
Automatic `BeanPostProcessor` registration	No	Yes
Automatic `BeanFactoryPostProcessor` registration	No	Yes
Convenient `MessageSource` access (for i18n)	No	Yes
`ApplicationEvent` publication	No	Yes

Demo - first-spring

IoC Background

- "The question is, what aspect of control are [they] inverting?" Martin Fowler posed this question about Inversion of Control (IoC) on his site in 2004. Fowler suggested renaming the principle to make it more self-explanatory and came up with Dependency Injection.
- For insight into IoC and DI, refer to Fowler's article at http://martinfowler.com/articles/injection.html.

Assignment

- Implement a Simple IoC container
  - be able to load the following xml file
  - be able to wire injection
  - be able to get bean instance from context

```xml
<bean id="foo1" name="foo1" class="x.y.Foo">
    <constructor-arg ref="baz1"></constructor-arg>
    <constructor-arg ref="bar1"></constructor-arg>
</bean>

<bean id="bar1" class="x.y.Bar"></bean>
<bean id="baz1" class="x.y.Baz"></bean>
```

DI

- Constructor-based dependency injection

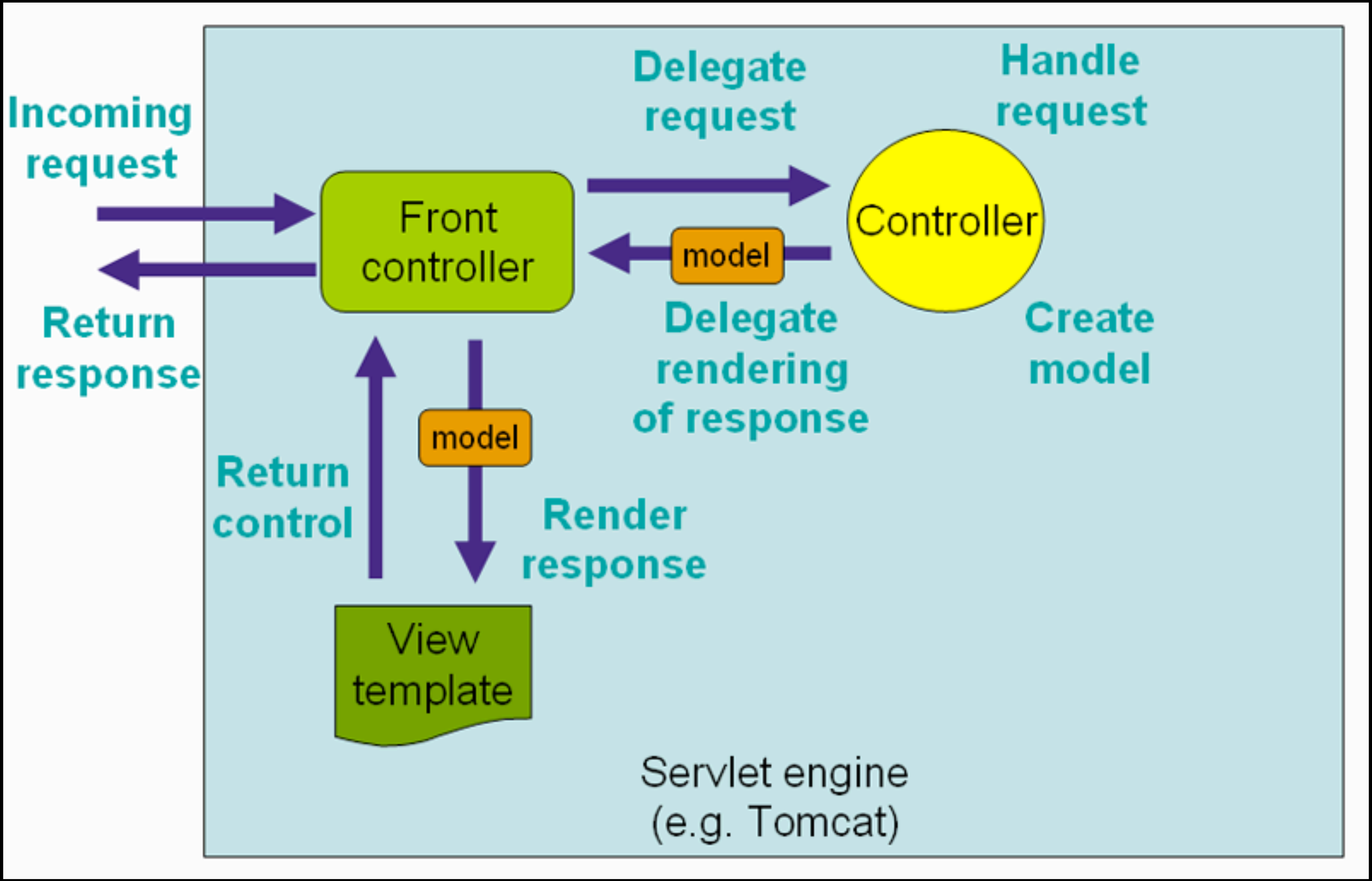
```
<bean id="foo1" name="foo1" class="x.y.Foo">
    <constructor-arg ref="baz1"></constructor-arg>
    <constructor-arg ref="bar1"></constructor-arg>
</bean>

<bean id="bar1" class="x.y.Bar"></bean>
<bean id="baz1" class="x.y.Baz"></bean>
```

- Setter-based dependency injection

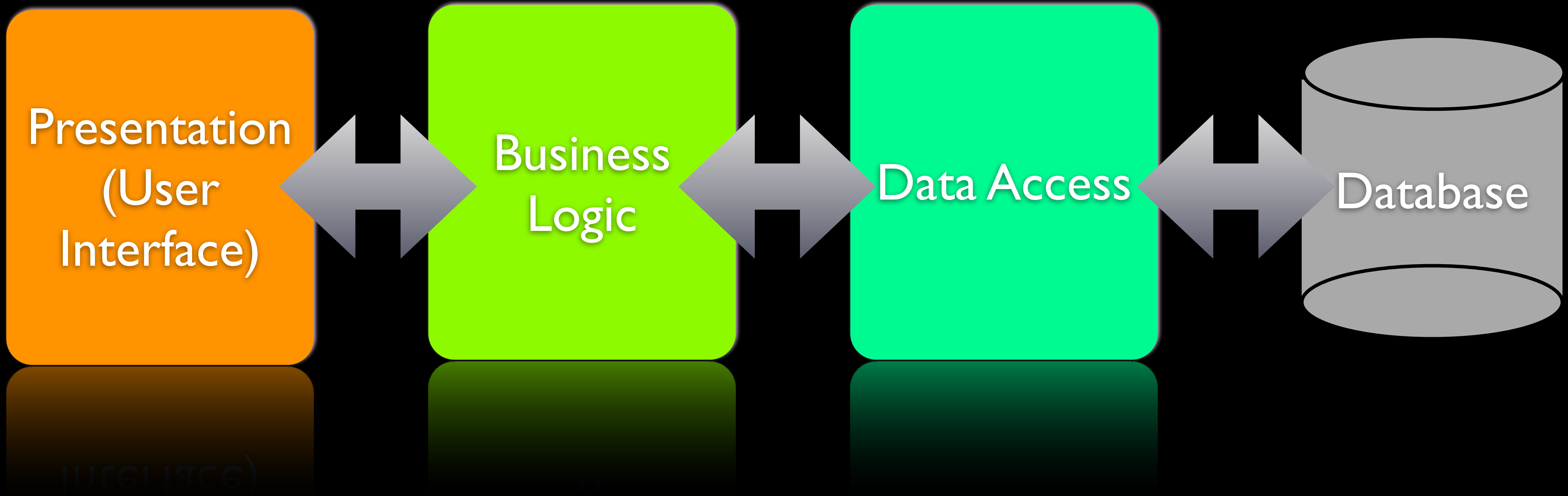
```
<bean id="foo1" name="foo1" class="x.y.Foo">
    <property name="baz" ref="baz1"></property>
    <property name="bar" ref="bar1"></property>
</bean>

<bean id="bar1" class="x.y.Bar"></bean>
<bean id="baz1" class="x.y.Baz"></bean>
```

Demo - 2 dependency injection

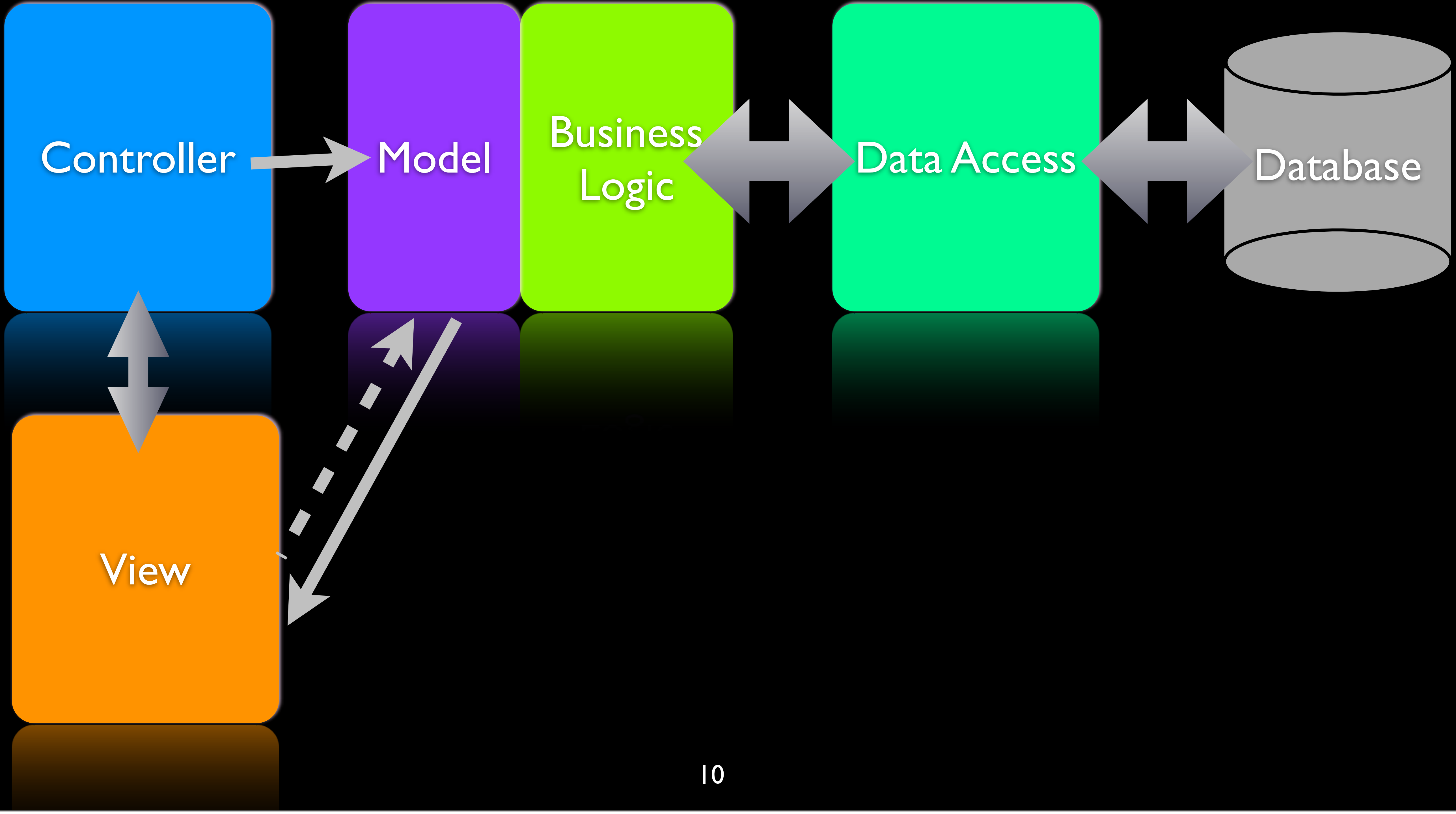
Spring MVC


Incoming request
Return response
Front controller
Delegate request
Handle request
Controller
model
Delegate rendering of response
Create model
model
Return control
Render response
View template
Servlet engine (e.g. Tomcat)

Loose Couple



MVC


Controller
Model
Business Logic
Data Access
Database
View

Q 0

- How to prepare your design environment?
  - SpringSource Tool Suite
  - http://www.springsource.org/downloads/sts

Demo - first-springmvc

Q 1

- how to mapping from url to java code?

Q 2

- how do we respond to the HttpRequest?

Q 3

- how to store data in session?

Q 4

- how to send data to an JSP? since we already learned that we can respond and http request with an JSP

Q 5

- how to handle form post data?

Q 6

- how to do form data validation?

Q 7

- what about the RESTful support?

Q 8

- Any reference application like Struts mailreader example?
    - https://github.com/SpringSource/spring-mvc-showcase

Q&A

References

- http://martinfowler.com/articles/injection.html
- < J2EE Development without EJB >