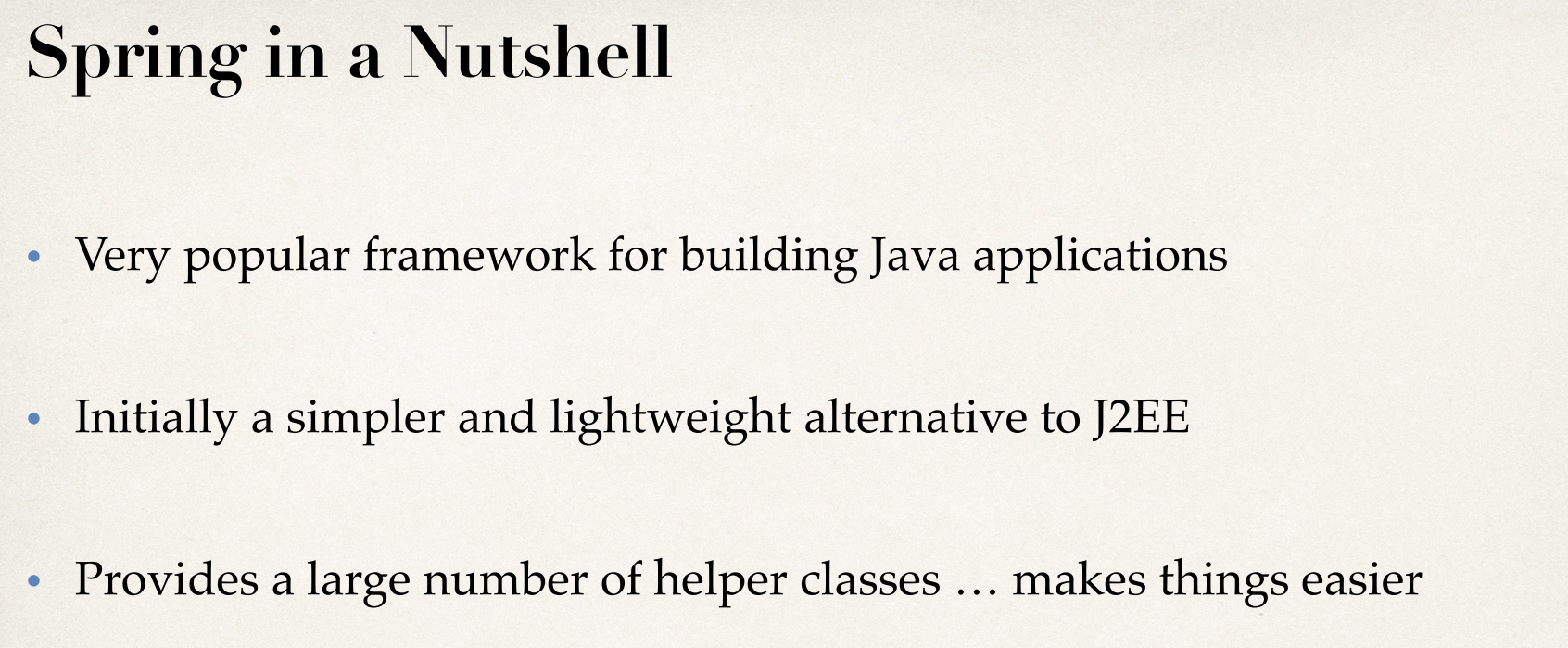
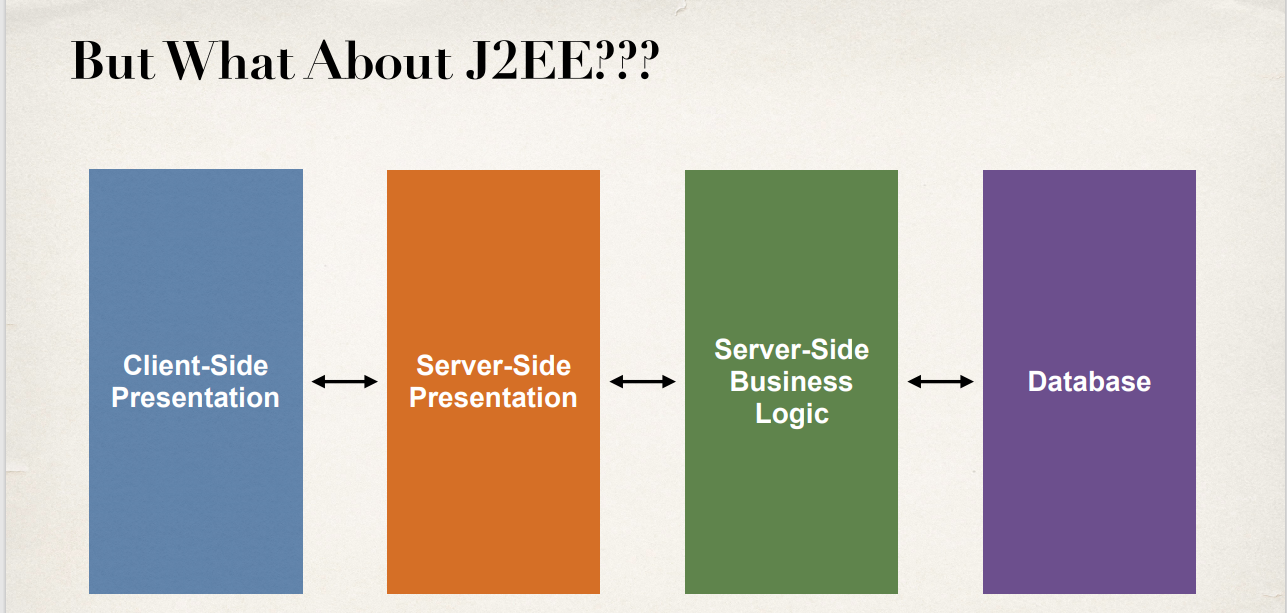
Spring in a Nutshell

Spring is a very popular framework for building enterprise java applications. With the initial release of spring it looked more like an lightweight alternative to J2EE where spring was providing a lot of helper methods to ease the enterprise development.

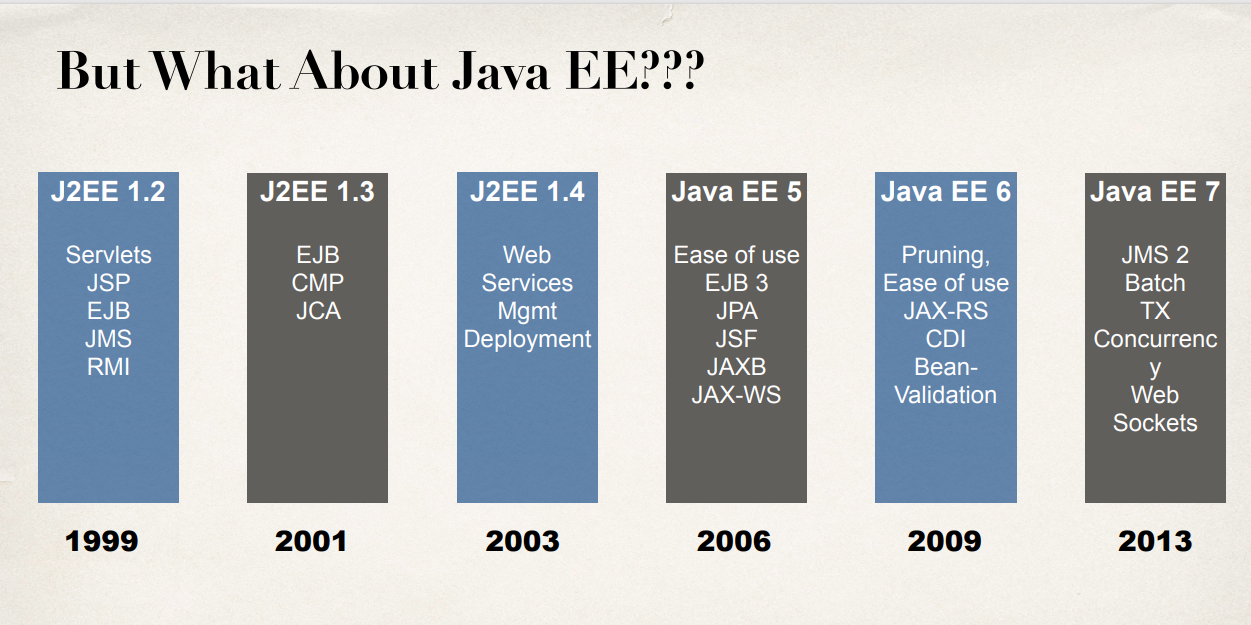


What About J2EE

In j2ee we have client side presentation, server side presentations (jsp, servlet, XML and web services.). In j2ee we also have server side business logic using Enter prize java beans. They also provide means of communicating with the data base.



Everything looked good on the proposal when j2ee was proposed. But when it was time for implementation it became a different story.



JMS : java message service

RMI: Remote method invocation

CMP: Component managed Persistance

JCA: Java Connector Architecture

JPA : java persistent API

JSF: Java server faces

JAXB: java API for XML binding

JAX-WS : java web services (SOAP)

JAX-RS : java web services (REST)

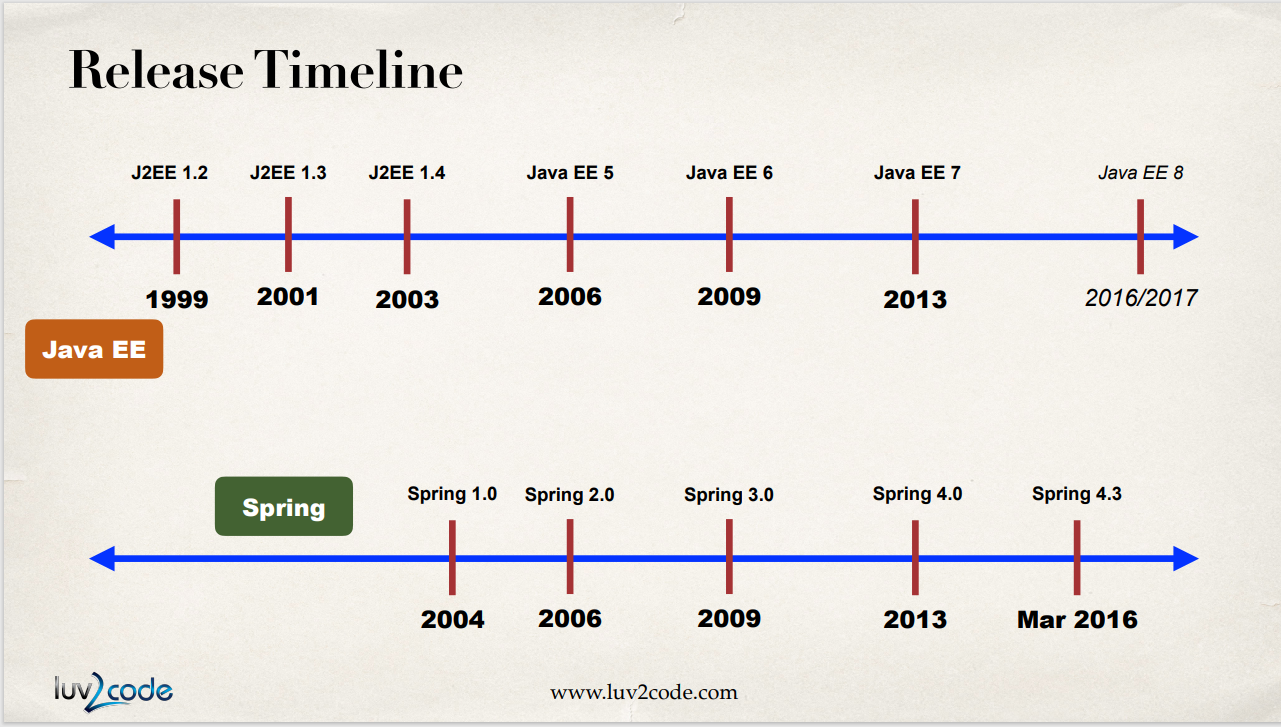
CDI : Context Dependency Injection (IoC)

This is how J2EE grow up over the years.

EJB v1 and v2 complexity

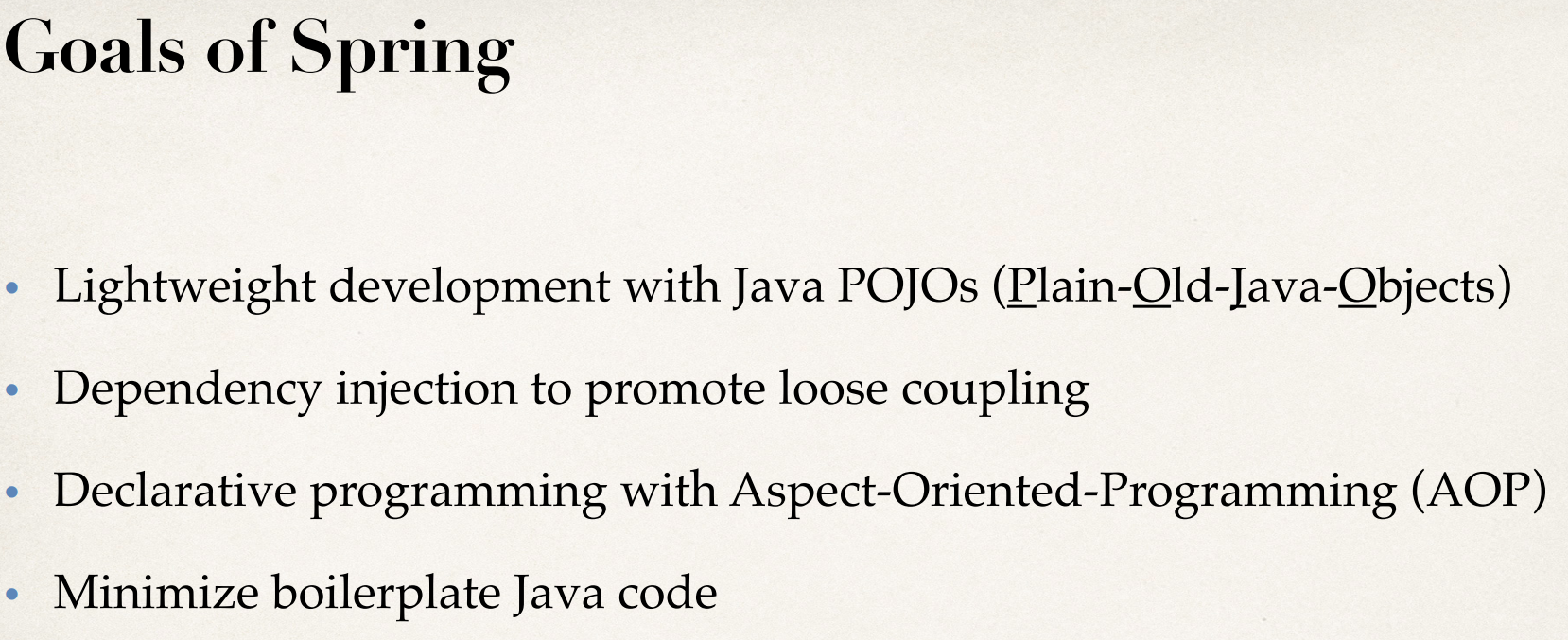
While j2ee was evolving one thing they added in the way was enterprise java beans (EJB). Creating EJB in the v1 and v2 were extremely complex. Development of EJBs in the earlier versions involved creation of multiple XML deployment descriptors, creation of multiple interfaces. Hence the entire process of development was tedious and error prone.

And once these beans were developed and deployed into production, they slowed down everything else. So it also so happened, developers had to pull there code out from the production and re write the entire code without the EJBs.

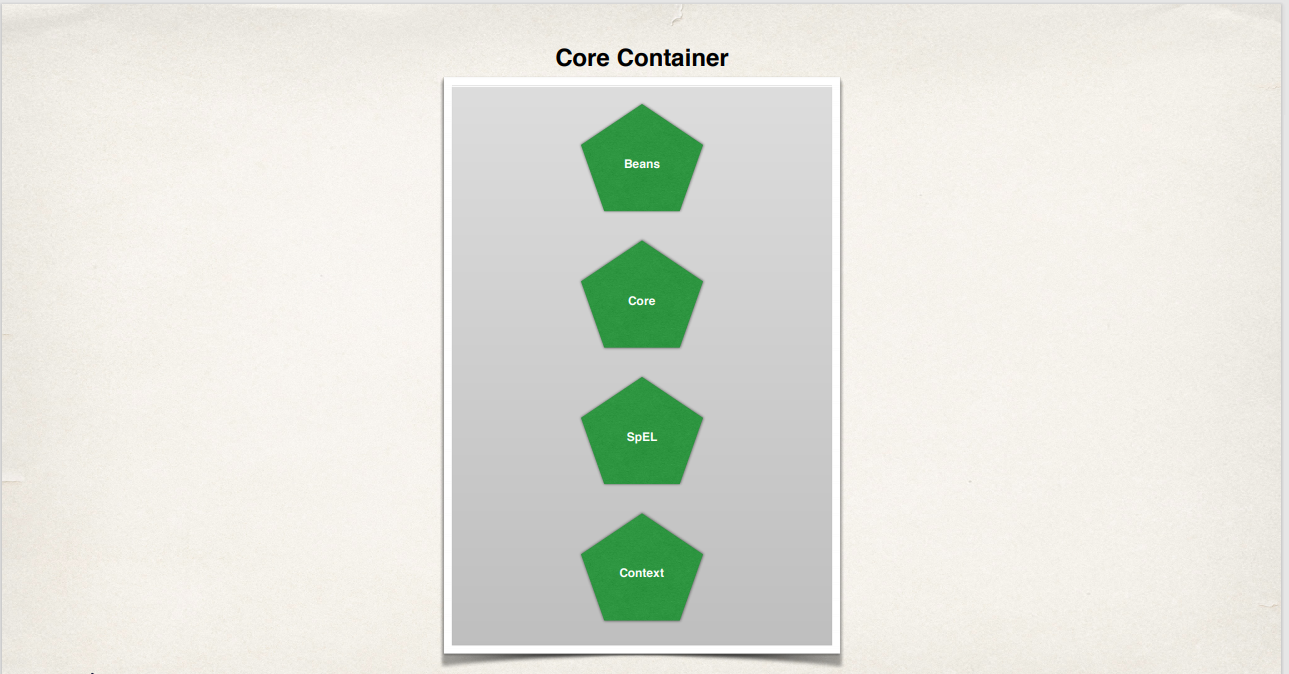


**Goals of Spring**

* Light weight development with POJO (Plain-Old-Java-Objects).
* Promoting loose coupling by using dependency injection. Instead of hardwiring our objects together we can specify the same within the configuration files or annotations. //more on this later.
* Declarative programming with Aspect-Oriented-Programming (AOP), where it is possible to have application wide services.
* Minimizing boiler plate java code. Back in the days of j2ee there was a lot of code to be written. Therefore, people at spring provided helper classes and they worked towards minimizing this boiler plate code.

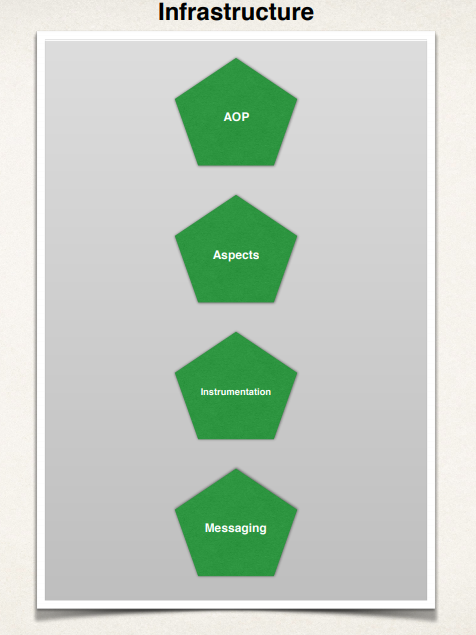
****

**Core container:** This is the heart or main item of spring. This manages how beans are created. It has a bean factory for creating the beans. This core container can read config files for setting properties and setting dependencies. ‘Context’ over here can be defined as the spring container that hold the beans in the memory. Bottom line is, core container is used to create beans and making those beans available.



**Infrastructure:**

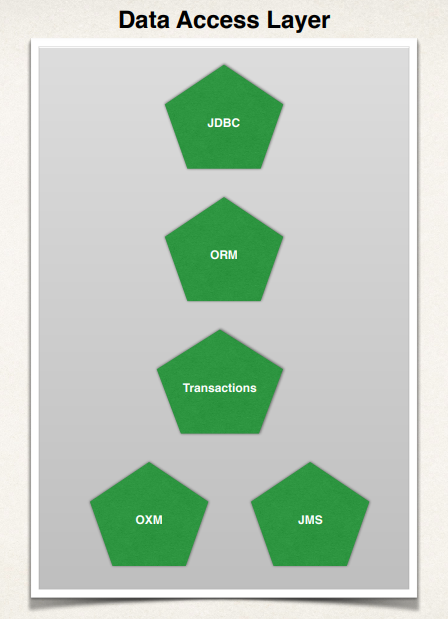
Here we have support for AOP (Aspect Oriented Programming). With the help of AOP we will be able to create application wide services (like logging, security, transactions). We can apply these services to our objects in a declarative manner by using annotation or config files.



**Data Access Layer:**

This layer is for communicating with the database. The database being targeted can either be a relational database or no-sql database or using a system like message queue.

Spring provides some helper JDBC classes using which our code can be reduced by 50%. The ORM is actually the object-relational-mapping which helps integrate spring with hibernate.



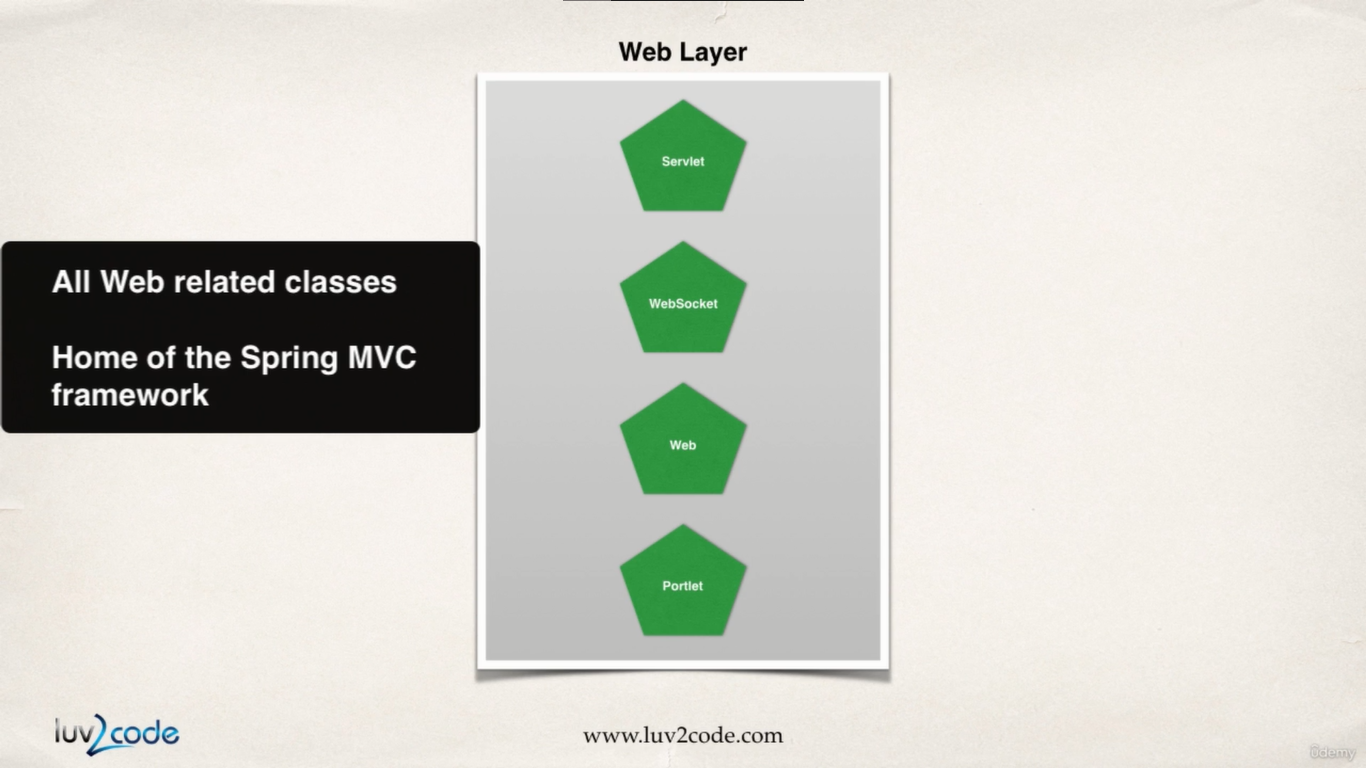
Using JMS we will be able to send async messages to message queue. Spring provides helper classes for JMS.

**Web Layer:**

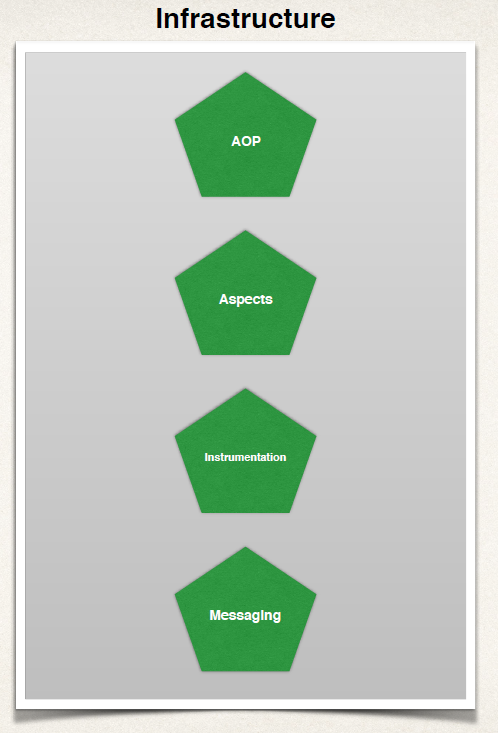
This is a home for spring MVC framework. We can build web application using spring core, spring controllers and spring view.

In this layer we have other modules which can be used to integrate other web technologies like JSF or Struts. If we are already using spring we can go ahead and use spring mvc in a straight forward manner but we can still integrate with other web technologies.

In here we get support for remoting, which means we can can have ‘web-remoting’ where we can have ‘external clients’ make calls into the spring container.



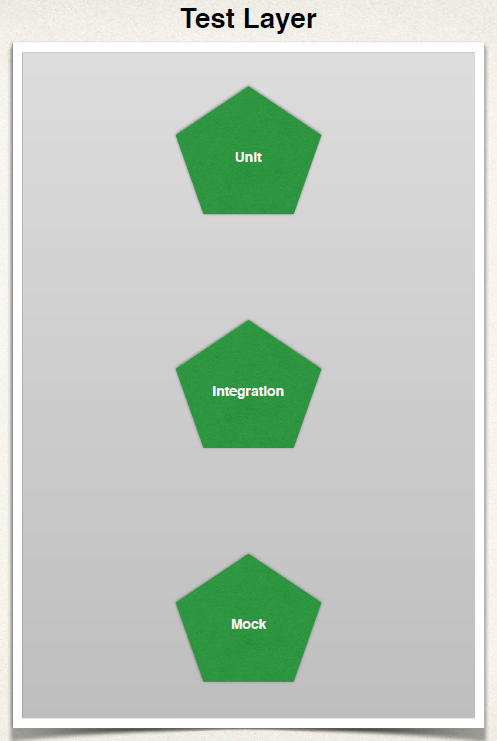
**Infrastructure:**

****

Spring does have some sophisticated technologies behind the scenes. In this section we can make use of class loader implementations to work with different apps service. For example it can be used to create java agent so we can remotely monitor our application using JMX.

As a developer we would not normally build an java agent by ourselves rather we can use the agents provided by the spring team or our app server vendor.

**Test Layer:**

****

Spring provides support for test driven development and the framework includes mock service and data services.