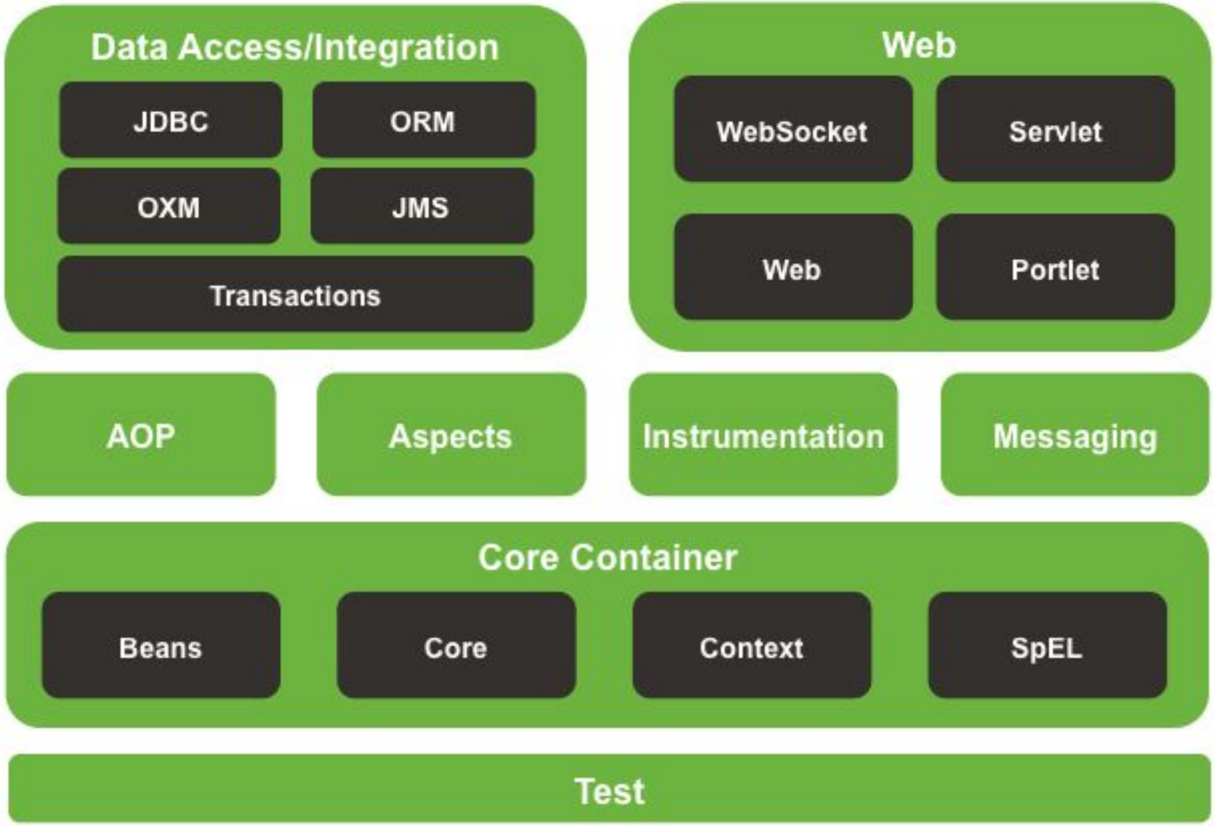
Spring MVC

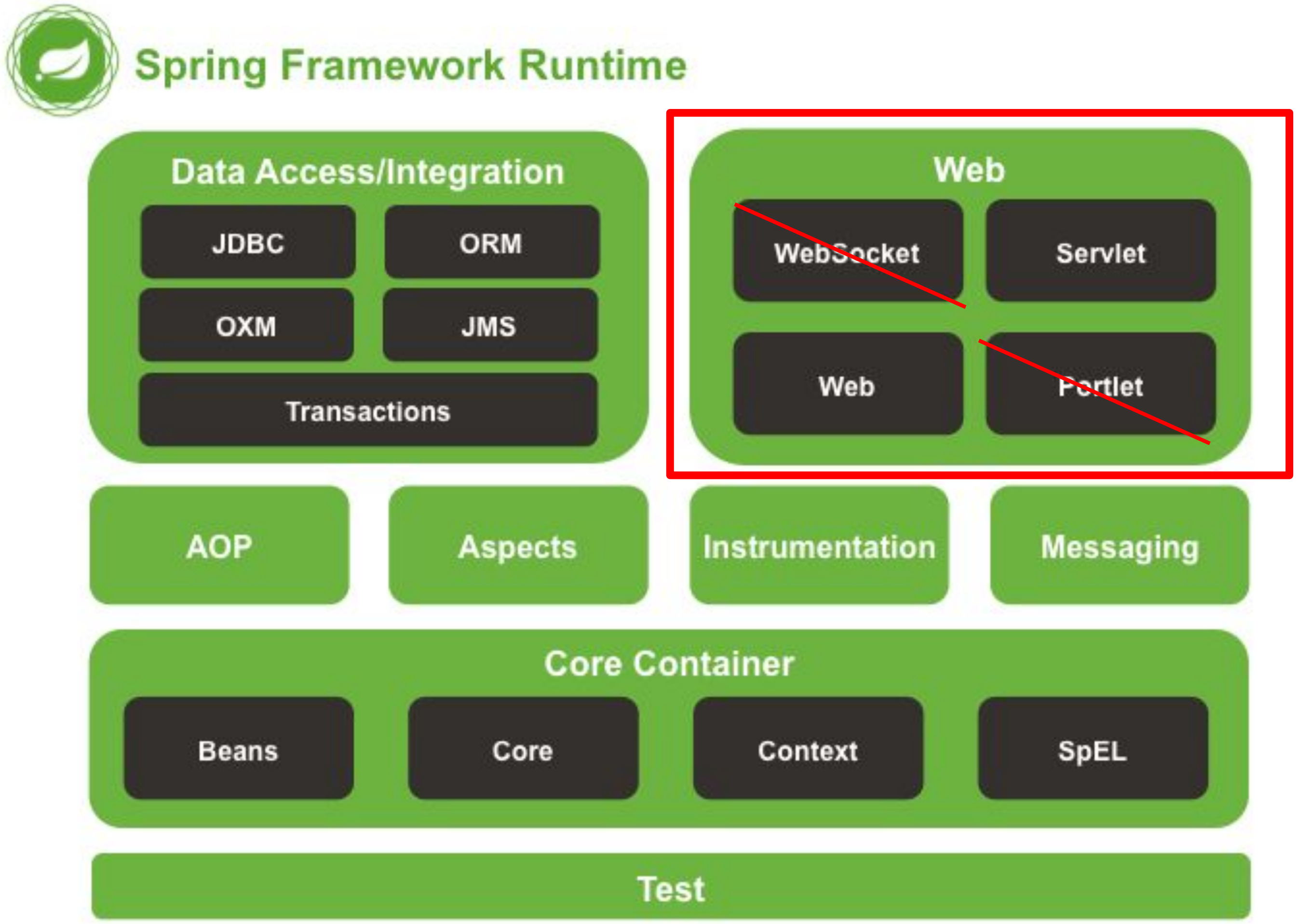
For Spring 4.3.x

Review: MVC Design Pattern

- Model: The data being passed, rendered, and manipulated
- View: What will be displayed, usually as html
- Controller: Handles logic, routing

Spring Framework Runtime



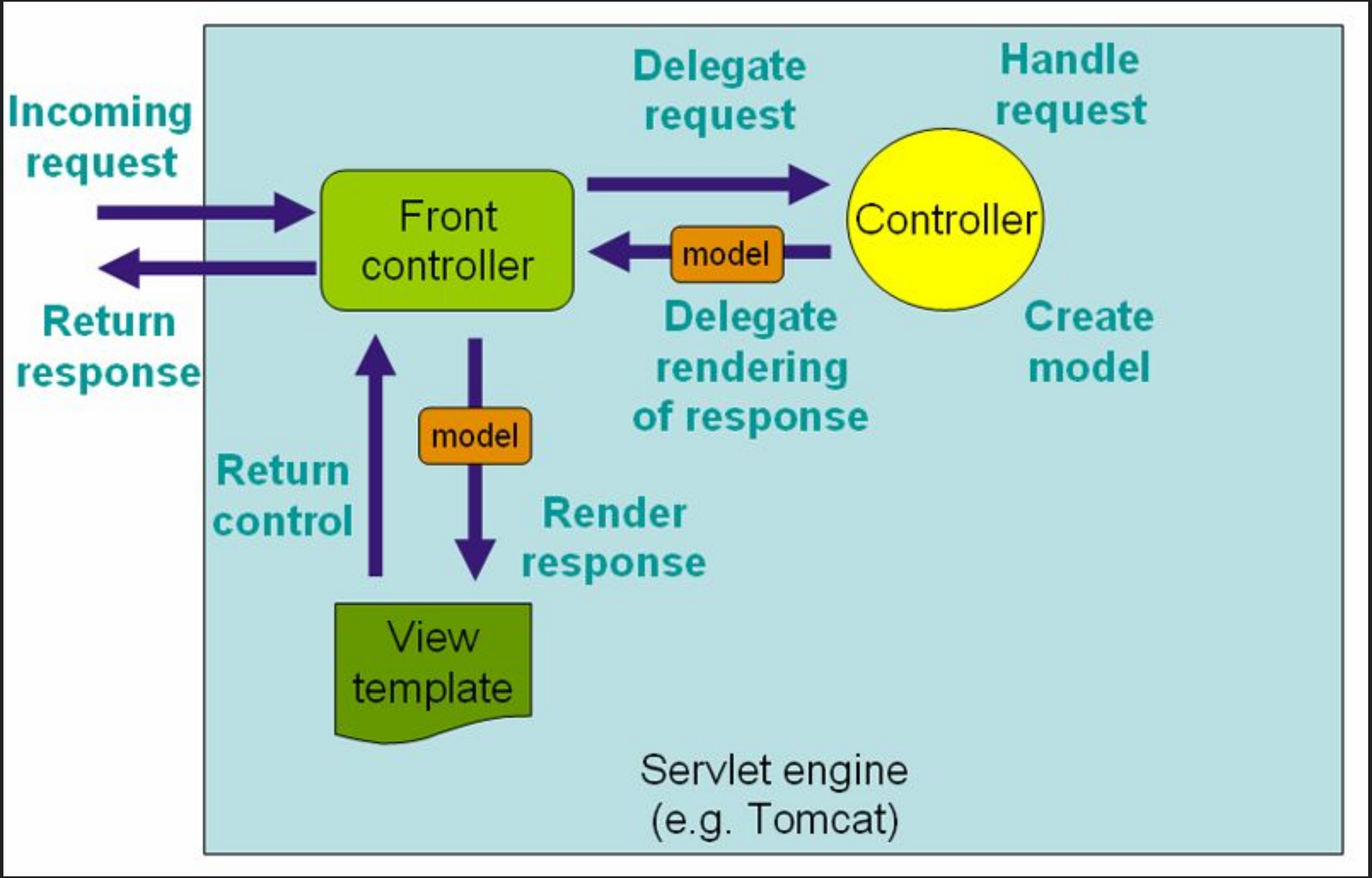

Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
WebSocket
Servlet
Web
Portlet
AOP
Aspects
Instrumentation
Messaging
Core Container
Beans
Core
Context
SpEL
Test

Overview

- Centered around a DispatcherServlet that implements the Front Controller design pattern
    - DispatcherServlet subclasses HttpServlet and uses the classic web.xml
- DispatcherServlet has its own WebApplicationContext, similar to the classic ApplicationContext
- The Model (from MVC=Model,View,Controller) is a Map interface
- Highly flexible data binding and view resolution
- Like everything in Spring Framework, configurable and modular

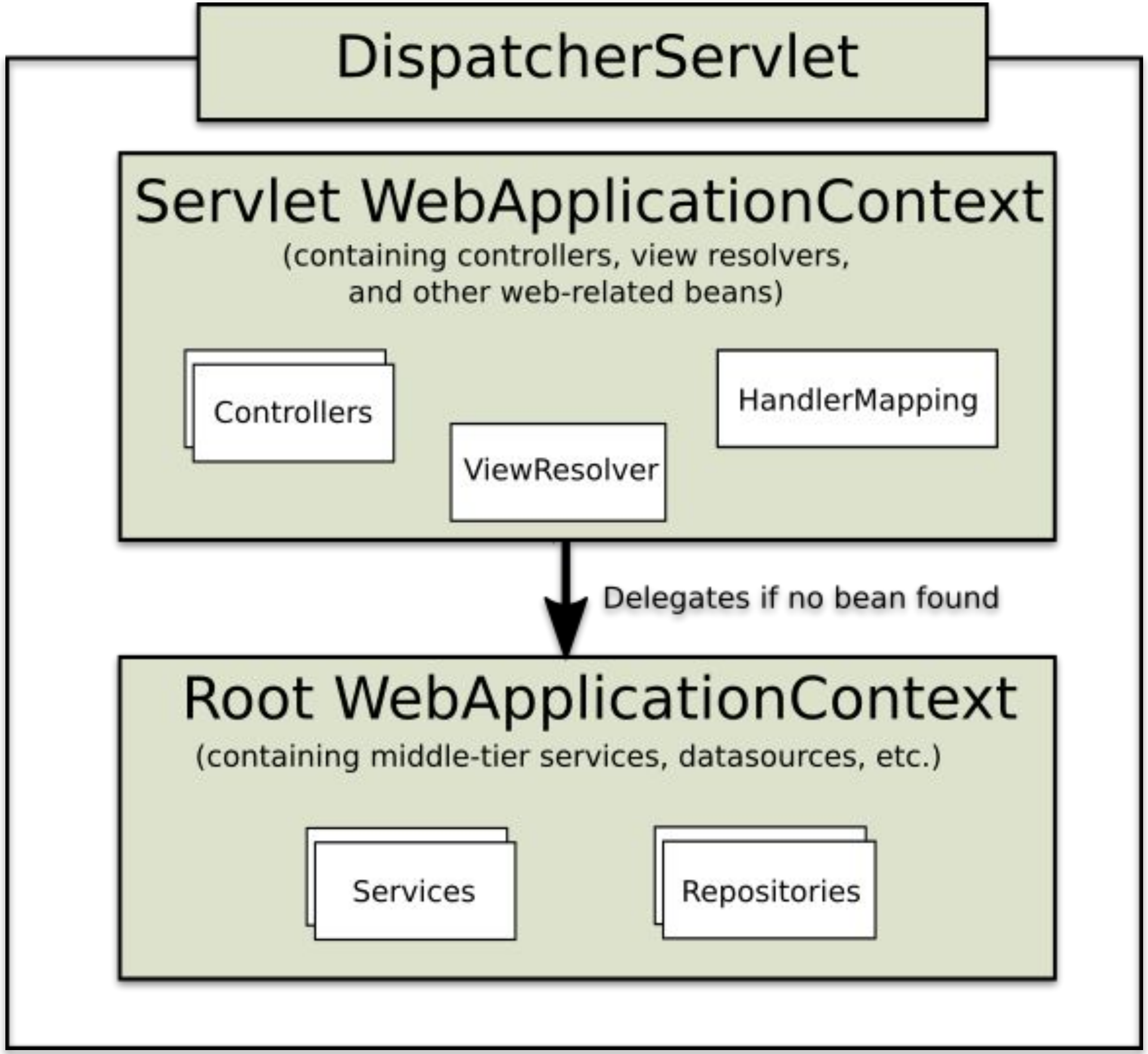
DispatcherServlet

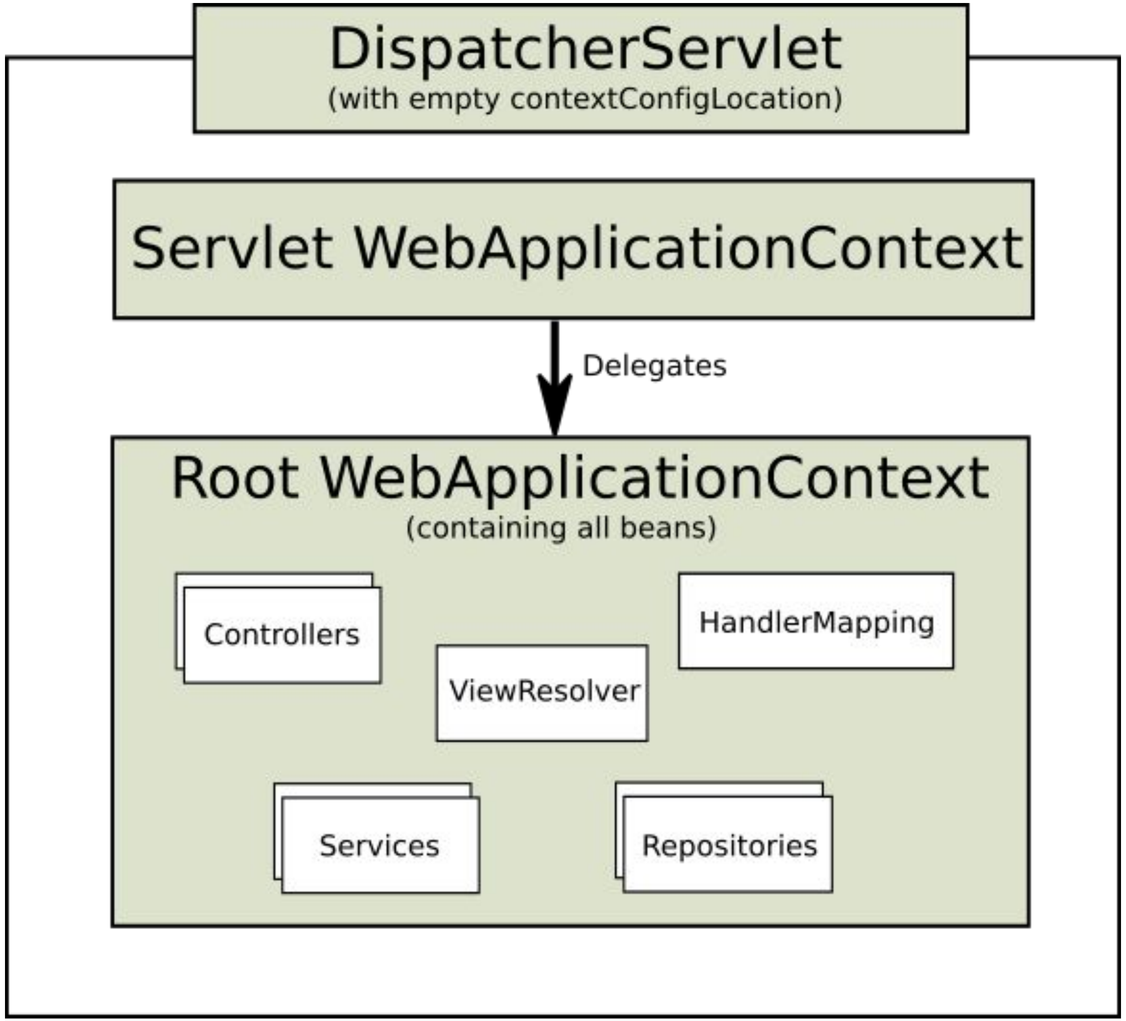
- Spring MVC is request-driven, and each request is managed by the DispatcherServlet
- Integrated with the IoC container
- Implementation of Front Controller design pattern
- Has its own WebApplicationContext that’s scoped to the servlet and inherits from the root WebApplicationContext


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)

WebApplicationContext

- Extends ApplicationContext for the web!
- Tied to servlet via servlet context
- Uses a series of beans by default to process requests and render views:
    - These are, of course, configurable.
    - For example it’s quite common to configure an InternalResourceViewResolver by setting its prefix property to the parent location of view files.
- Has more bean scopes than ApplicationContext:
    - Request, session, global


DispatcherServlet
Servlet WebApplicationContext
(containing controllers, view resolvers, and other web-related beans)
Controllers
ViewResolver
HandlerMapping
Delegates if no bean found
Root WebApplicationContext
(containing middle-tier services, datasources, etc.)
Services
Repositories


DispatcherServlet
(with empty contextConfigLocation)
Servlet WebApplicationContext
Delegates
Root WebApplicationContext
(containing all beans)
Controllers
HandlerMapping
ViewResolver
Services
Repositories

WAC Default Beans

- **HandlerMapping** : Maps incoming requests to handlers and a list of pre- and post-processors (handler interceptors) based on some criteria. Most often supports annotated controllers.
- HandlerAdapter : Helps the DispatcherServlet to invoke a handler mapped to a request regardless of the handler is actually invoked. For example, invoking an annotated controller requires resolving various annotations.
- HandlerExceptionResolver : Maps exceptions to views and allows for more complex exception handling code.
- **ViewResolver** : Resolves logical String-based view names to actual View types.
- … and more! Bold beans are more important

WAC Beans

- Any bean we define in our ApplicationContext we can define in our WebApplicationContext
- Default beans just defined
- Controllers
- Services
- Repositories

Spring MVC Request-Response Flow

- Request sent to DispatcherServlet
- DispatcherServlet calls HandlerMapping for help with request URI
- HandlerMapping looks up the handler for the URI, a registered Controller which is returned to the DispatcherServlet and called
- Controller is the entry-point for an event in and out of the rest of the program
- Controller returns a View name & Model to the DispatcherServlet
- DispatcherServlet consults ViewResolver to interpret View name as a template file and weave the Model into the response body
- Response sent to client

Note: Controllers and Flow

- Spring MVC is highly configurable
- Building your views from a template with something like a JSP is less popular lately
- We can use the Controller to short-circuit the Request Flow to skip views
- This happens normally in a RESTful Spring Webapp that produces JSON

Bean Scopes, Expanded

- In a web-aware applicationcontext, we have our two basic scopes:
  - Singleton
  - Prototype
- We also have 3 scopes specifically for servlets and requests:
  - Request : scoped to a single http request
  - Session : scoped to a single session, server-side
  - Global : scoped to the servlet container