Building Web Applications with the SpringSource Application Platform

Sam Brannen
Senior Software Engineer
SpringSource

Learn how to ...

- Build web applications with the SpringSource Application Platform: from WAR to Web Module

Speaker’s qualifications

- Sam is a member of the SpringSource Application Platform development team and is responsible for OSGi-enabled web deployment models and Tomcat integration in the Platform.
- Sam is a veteran web developer with 10 years' experience and is also a member of the core development team for the Spring Framework.
- He is a regular speaker at conferences covering topics such as Core Spring, Spring MVC, and the Spring TestContext Framework.
- In his spare time, Sam is a technical reviewer for Spring related books.

Monolithic WAR deployments hinder both development and runtime modularity and promote *library bloat.*

The SpringSource Application Platform harnesses the power of OSGi, Spring, and Spring-DM to solve these issues and more.

Agenda

- OSGi
- Spring-DM
- SpringSource Application Platform
- Deployment Options on the S2AP
- Web Application Demos

OSGi

- OSGi: Open Services Gateway initiative
- Dynamic module system with a central notion of a *bundle*
- Bundle: a JAR file with special manifest headers and explicit versioning
- Specify dependencies on other bundles
- Dynamically install, start, stop, refresh, and uninstall
- Enables hot redeployment at runtime!
- Publish and consume services via the OSGi Service Registry

Spring Dynamic Modules for OSGi™ Service Platforms

- Spring in an OSGi environment
- Simplicity and power of Spring … with the dynamics of OSGi
- Spring's fine-grained component model and middleware services
- Spring-DM manages the Spring **`ApplicationContext`** life cycle
- One **`ApplicationContext`** per bundle
- OSGi manages bundles and services at runtime

Spring-DM + OSGi *vs.* Spring + Java EE

- Similar programming models
- Services implemented as Spring managed components
- No direct dependencies on runtime environment: just Spring + JDK
- Different deployment environments
- OSGi: bundles, service registry
- Java EE: WAR / EAR files, JNDI environment
- OSGi and Java EE are alternative runtimes
- Spring provides the common ground!

SpringSource Application Platform

- Server platform: Dynamic Module Kernel™ (dm-Kernel)
- Modular profiles, bundle repository, library provisioning
- Serviceability: FFDC, logging, tracing
- Personality Deployers: Web (future: Batch, Web Services, ...)
- OSGi Container: modular architecture, small footprint, bundles, service registry
- Spring-DM powered bundles: Spring application contexts, service publication & consumption

Deployment Options on the S2AP

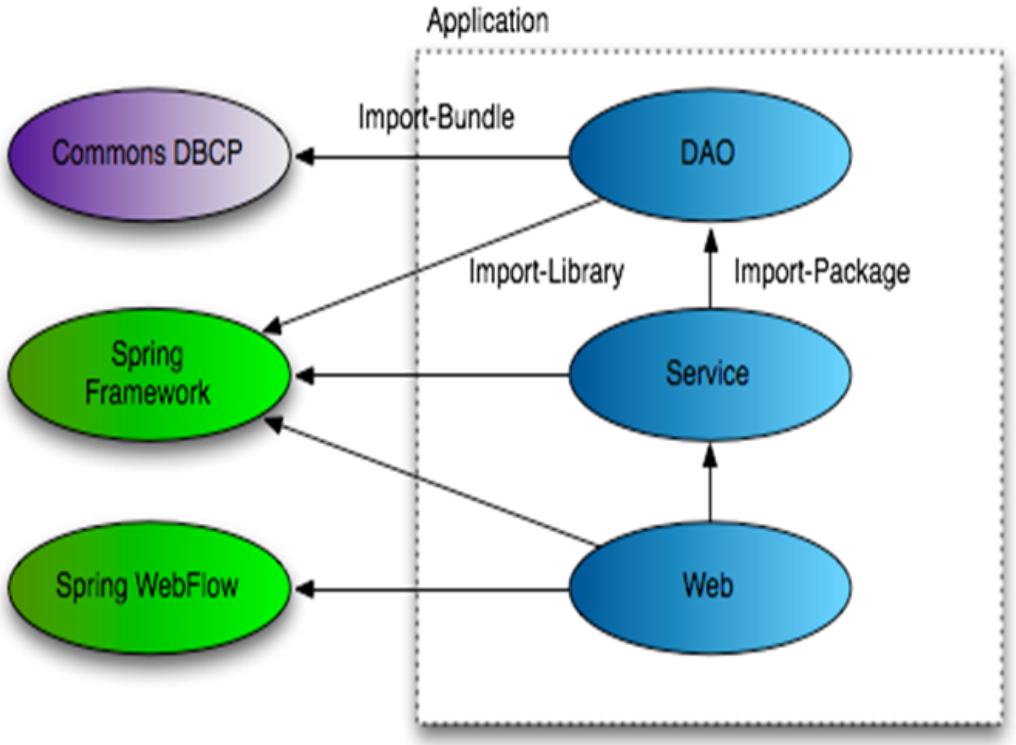
- Raw OSGi Bundle
- Platform Archive (PAR)
- Web Applications: WAR and Web Module

Raw OSGi Bundles

- Bundle: basic deployment module in an OSGi container
- Any OSGi-compliant bundle can be deployed as-is on the S2AP
- Typical usage scenarios:
- Stand-alone libraries
- Global services
- Small, stand-alone applications

Platform Archives (1/3)

- What is a PAR?
- Packaging format for all modules in an application
- Standard JAR with **`Application-*`** manifest headers
- Single application-level unit: deploy, refresh, undeploy
- Replacement for an EAR within the context of an OSGi container



Platform Archives (2/3)

- Refresh individual modules within a PAR
- S2AP Tools for Eclipse (uses the **`Deployer`** MBean)
- JConsole (via the **`Deployer`** MBean)
- JIRA opened for similar support via the web admin console

Platform Archives (3/3)

- Logical and physical application boundaries
- Scoping of types and services
- Synthetic context bundle
- Component scanning
- Load-time weaving
- Context class loading
- Per-application trace (across the PAR)

Web Application Deployment Options

- Standard Java EE WAR
- Shared Libraries WAR
- Shared Services WAR
- Web Module

Recipe for a Web Application

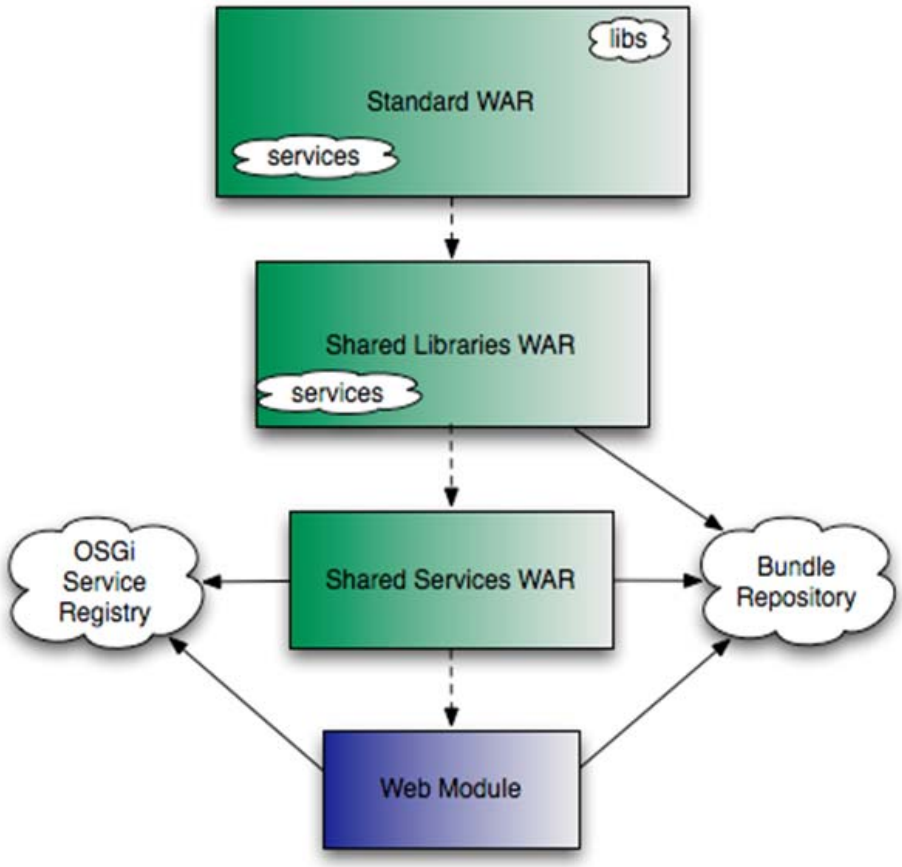
- **Static content**: HTML, CSS, JavaScript, images, ...
- **Dynamic content**: Servlet, JSP, JSF, ...
- **Dependencies**: third-party libraries, application JARs, ...
- **Infrastructure**: services, data sources, JNDI, ...
- **Configuration**: `web.xml` deployment descriptor, Spring `ApplicationContext`, ...

Introduction to the Hello * Demos

- Spring MVC 2.5 based "Hello World" web application
- Domain model: **`HelloMessage`**
- Service layer: **`HelloService`**, **`HelloServiceImpl`**
- Presentation layer:
- Controllers: **`HelloController`**
- Views: JSP + JSTL

WAR to Web Module Migration Path

- **Status quo**: everything in a monolithic WAR
- **Goals**:
- Libraries: eradicate "library bloat"
- Services: OSGi Service Registry
- **Results**: decrease overall footprint



Standard Java EE WAR

- Supported directly on the Platform *as is*
- The WAR file is transformed into an OSGi bundle and installed into Tomcat
- All standard WAR contracts are honored
- Easy on-ramp to try out the S2AP

DEMO

Standard Java EE WAR

Shared Libraries WAR

- Eradicate the *library bloat* of monolithic Java EE WARs
- Declare dependencies via OSGi manifest headers
- **`Import-Package`** & **`Require-Bundle`**
- **`Import-Library`** & **`Import-Bundle`** → **`Import-Package`**
- Reduce application deployment footprint
- *SpringSource Enterprise Bundle Repository*: central source for OSGi bundles

DEMO

Shared Libraries WAR

Shared Services WAR

- Share services between bundles and web applications
- Spring-DM simplifies usage of the OSGi Service Registry
- Publish services via **`<osgi:service … />`**
- Consume them via **`<osgi:reference … />`**
- ***`PlatformOsgiBundleXmlWebApplicationContext`*: use as `contextClass` for `ContextLoaderListener` and `DispatcherServlet`**
- *Program to interfaces and decouple web artifacts from the domain* model, services, infrastructure, etc.

DEMO

Shared Services WAR

Web Module

- Deployment and packaging option for OSGi-compliant web applications
- Stand-alone or within a PAR
- OSGi bundle, structure similar to Shared Services WAR
- Reduced configuration for Spring MVC applications via web manifest headers:
- **`Web-DispatcherServletUrlPatterns`**, **`Web-FilterMappings`**, **`Platform-ModuleType`**, etc.
- Auto-configuration of Spring MVC's **`DispatcherServlet`**
- Single **`WebApplicationContext`** and no Root WAC
- Additional configuration via *web.xml fragments*

DEMO

Web Module

PAR with a Web Module

Logging and Tracing

- Global logging and tracing
- Per-application trace logs
- Web application access logs

DEMO

Logging and Tracing

Where to go from here

- SpringSource Application Platform Beta Program: http://www.springsource.com/beta/s2ap
- Downloads
- User Guide and Programmer Guide
- PetClinic and Form Tags sample applications
- S2AP Forum
- S2AP Tools for Eclipse
- JIRA issue tracker: http://issuetracker.springsource.com

Further Resources

- SpringSource Team Blog: http://blog.springsource.com
- OSGi: http://www.osgi.org
- Spring-DM: http://www.springframework.org/osgi
- Spring Framework: http://www.springframework.org

Summary

- S2AP = Server Platform + OSGi + Spring-DM + Spring
- Platform Archive = logical & physical application boundary
- S2AP supports multiple web deployment formats
- Java EE WAR → Shared Libraries WAR → Shared Services WAR → Web Module
- The result ...

The Healthy New Way to Run Your Apps

- Download the SpringSource Application Platform, take it for a spin, and start building your own *bloat-free* web applications!

Questions?

Sam Brannen
sam.brannen@springsource.com

http://springsource.com