Root ApplicationContext (1)

- ¿Qué es un Root ApplicationContext en Spring MVC?
  - ✓ Tiene la declaración de los componentes (archivo XML con la declaración de BEANS) que pueden ser utilizados en toda la aplicación.
    - Middle-Tier Service
      - Componentes de lógica de negocio
      - Componentes de acceso a datos (repositorios)
  - ✓ Solo hay un Root ApplicationContext por cada aplicación web.
    - A diferencia de un WebApplicationContext que pueden ser varios (uno por cada DispatcherServlet declarado).
- El Root ApplicationContext es inicializado por un Listener (ContextLoaderListener) definido en el archivo web.xml de nuestra aplicación.

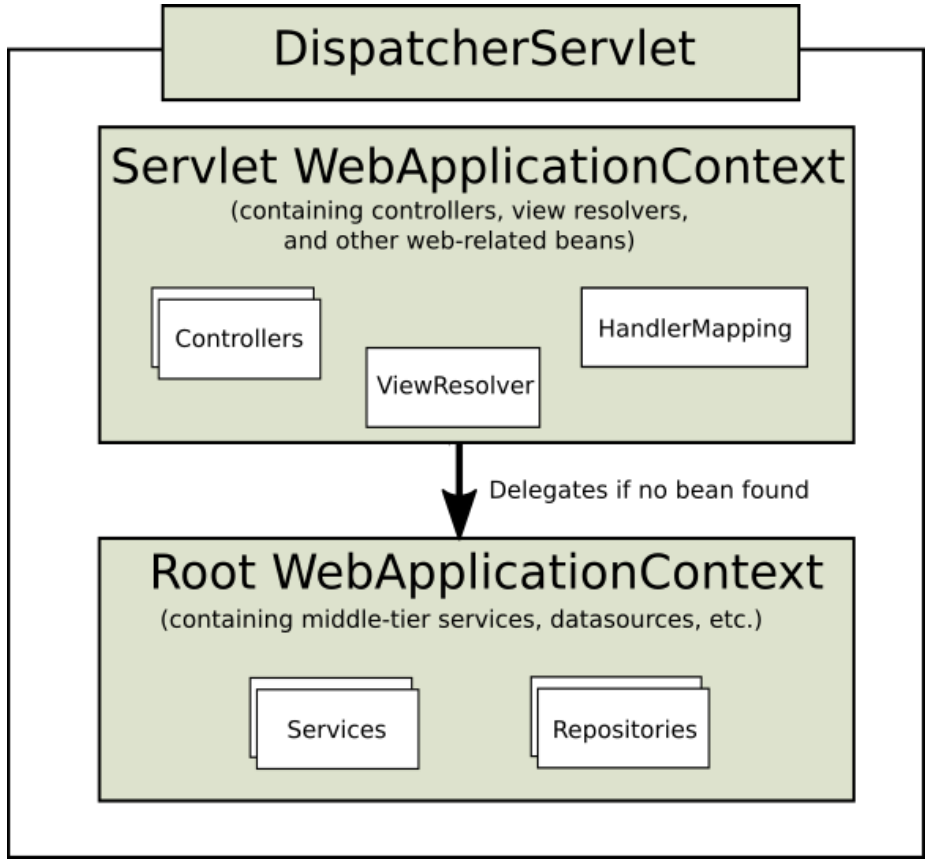
```
<listener>
   <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

- Si no es especificado el nombre del archivo de configuración del Root ApplicationContext, Spring por default buscará un archivo llamado applicationContext.xml en el directorio WEB-INF.

Root ApplicationContext (2)

- Para especificar un nombre de archivo diferente del Root ApplicationContext utilizamos un tag <context-param>

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>
```



El Root WebApplicationContext tiene componentes que pueden ser usados para toda la aplicación:

- Componentes de Servicio
  - Service
- Componentes de Datos
  - Repository
  - DAOs

Nota: La flecha indica que desde un WebApplicationContext (DispatcherServlet) se pueden mandar llamar componentes (beans) del Root ApplicacionContext. La inversa no es posible.

Anotación @Autowired – Inyectar clases de Servicio

- Procedimiento para inyectar (@Autowired) una clase de servicio en un controlador:

1. Activar el autoescanéo para detectar e instanciar beans en el Root ApplicationContext.

```
<context:component-scan base-package="net.itinajero.app.service" />
```

2. Anotar la clase de servicio (la implementación) con la anotación @Service de Spring Framework.

```
package net.itinajero.app.service;
@Service
 public class PeliculasServiceImpl implements IPeliculasService {
        // Métodos de lógica de negocio
        @Override
        public List<Pelicula> buscarTodas() {
                return lista;
        }
 }
```

3. Inyectar en nuestro controlador, una instancia de la clase de servicio.

```
@Autowired
private IPeliculasService servicePeliculas;
```

4. Utilizar los métodos de la clase de servicio en el controlador.

```
List<Pelicula> películas = servicePeliculas.buscarTodas();
```