

	<div><div><div><div><div><div><<ServletContextListener>> ContextLoaderListener</div><div>소유 (포함) ◇ ..</div></div><div><div>Spring IoC 컨테이너</div><div>✓ 공유됨!</div></div></div><div><div>이제 프로젝트 컨테이너에서 공유해서 사용</div><div>TxManagerDataSource Dao Service</div></div></div></div></div>
	<div><div><div><div><div><div>*084 Root IoC 컨테이너 사용</div><div>AppWebApplicationInitializer가 준비</div><div>여기 프로젝트 컨테이너가 공유해서 사용할 객체를 준비한다.(스프링에서 제공)</div><div><div><<ServletContextListener>> ContextLoaderListener</div><div>소유 ◇ ..</div></div><div>모든 프로젝트를 컨테이너가 공유함.</div><div>사용</div><div>Spring IoC 컨테이너</div><div>↑ 생성? 누가?</div><div>참고 →</div><div>RootConfig</div><div>- DAO - Service</div><div>객체를 준비한다.(공통으로 사용함)</div><div>의미 생성하면 ↑ 실질안하면 Spring IoCContextLoaderListener도 생성 안됨 컨테이너가 넘어감 IoC 컨테이너를 못만들고 넘어 가니깐 안됨.</div><div>DatabaseConfig</div><div>- DataSource - TransactionManager</div><div>AppWebApplicationInitializer가 준비</div><div>app</div><div><<Front Controller>> DispatcherServlet</div><div>소유 ◇ ..</div><div>Spring IoC 컨테이너</div><div>참고 →</div><div>AppWebConfig</div><div>- Controller - 기타 - 웹 환경 필요성 (admin* 제외)</div><div>AppWebApplicationInitializer가 준비</div><div>admin</div><div><<Front Controller>> DispatcherServlet</div><div>소유 ◇ ..</div><div>Spring IoC 컨테이너</div><div>참고 →</div><div>AdminWebConfig</div><div>- Controller - 웹 환경 필요성 (admin*)</div></div></div></div></div></div>
	<div><div><div><div><div><div><<interface>> WebApplicationInitializer</div><div>△ ↓</div><div><<abstract>> AbstractContextLoaderInitializer</div><div>△ ↓</div><div><<abstract>> AbstractDispatcherServletInitializer</div><div>△ ↓</div><div><<abstract>> AbstractAnnotationConfigDispatcherServletInitializer</div></div><div>2. onStartUp()</div><div>3. registerContextLoaderListener</div><div>1. onStartUp()</div><div>4. registerDispatcherServlet()</div><div>10. createDispatcherServlet()</div><div>12. registerServletFilter()</div><div>4. createRootApplicationContext</div><div>8. createServletApplicationContext</div><div>★추상메서드는 호출하지 못함. super클래스의 메서드를 호출</div><div>★JangD4: Template Method 패턴</div><div>★추상클래스에서 호출되는 메서드를 만들고 호출되는 구현을 서브클래스에서 함</div><div>★서브클래스에게 할당</div></div><div><div>1. Servlet 컨테이너 시작</div><div>2. 웹 애플리케이션을 시작</div><div>3. SpringServletContainerInitializer.onStartUp() 호출</div><div>4. WebApplicationInitializer의 Interface 규칙에 따라 onStartUp() 호출</div></div><div><div><<concrete>> AppWebApplicationInitializer</div><div>★추상메서드다 여기서 못해서 출구.</div><div>#5 getRootConfigClasses() {} (-)</div><div>#7 getServletName() {} (-)</div><div>#9 getServletConfigClasses() {} (-)</div><div>#11 getServletMappings() {} (-)</div><div>#12 getServletFilters() {} (-)</div><div>#14 customizeRegistration() {} (-)</div></div><div><div><<abstract>> AbstractDispatcherServletInitializer</div><div>△ ↓</div><div><<abstract>> AbstractContextLoaderInitializer</div></div><div><div><<abstract>> AbstractDispatcherServletInitializer</div><div>△ ↓</div><div><<abstract>> AbstractContextLoaderInitializer</div></div></div></div></div>
	<div><div><div><div><div><div><<abstract>> AbstractDispatcherServletInitializer</div><div>@Override</div><div>public void onStartUp(ServletContext servletContext) throws ServletException { ★super★.onStartUp(servletContext); registerDispatcherServlet(servletContext); }</div></div><div><<abstract>> AbstractContextLoaderInitializer</div></div><div><div>selected WebApplicationContext createWebApplicationContext() { Class<?>[] configClasses = getRootConfigClasses(); # (Object)Utils.isEmpty(configClasses) { AnnotationConfigWebApplicationContext context = new AnnotationConfigWebApplicationContext(); context.register(configClasses); return context; } 반환값이 넘어감 IoC 컨테이너를 못만들고 넘어 가니깐 안됨.</div><div># (rootAppContext != null) { ContextLoaderListener listener = new ContextLoaderListener(rootAppContext); listener.setContextInitializers(getRootApplicationContextInitializers()); servletContext.addListener(listener); }</div></div></div></div></div>
	<div><div><div><div><div><div>*085. Mybatis SQL Mapper 도입</div><div>[도입 전]</div><div>BoardDao</div><div>JDBC Code + SQL</div><div>*****</div><div>*반복적이고 단순한 JDBC 로직코딩을 피하게 된다</div><div>*** * * * * *</div><div>*SQL을 별도 파일로 분리</div><div>SQL을 다루기 더 쉽다</div><div>[도입 후]</div><div>BoardDao</div><div>↓ Call</div><div>Board</div><div><<Mybatis>> JDBC</div><div>점드</div><div>Use</div><div>SQL</div><div>점드</div><div>-- 자바 소스 파일에서 SQL 코드를 분리</div><div>-- JDBC 코드를 호출함</div><div>메서드를 호출스름 하는다.</div></div></div></div></div></div>
	<div><div><div><div><div><div>*Mybatis를 도입한 후의 DAO</div><div>[도입 전]</div><div>BoardDao</div><div>Call</div><div>JDBC API</div><div>SQL 실행</div><div><<Table>> app-board</div><div>[도입 후]</div><div>BoardDao</div><div>Call</div><div>Mybatis API</div><div>Call</div><div>JDBC API</div><div>SQL 실행</div><div><<Table>> app-board</div><div>Mapping해주는 것 간편적으로 흐름</div></div></div></div></div></div>

*Mybatis 사용법





















