|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 加载器目前有两种选择：ContextLoaderListener和ContextLoaderServlet。     这两者在功能上完全等同，只是一个是基于Servlet2.3版本中新引入的Listener接口实现，而另一个基于Servlet接口实现。开发中可根据目标Web容器的实际情况进行选择。   |  |  | | --- | --- | | 配置非常简单，在web.xml中增加：  <listener>         <listener-class>            org.springframework.web.context.ContextLoaderListener        </listener-class>  </listener>  或：  <servlet>           <servlet-name>context</servlet-name>           <servlet-class>             org.springframework.web.context.ContextLoaderServlet           </servlet-class>           <load-on-startup>1</load-on-startup>  </servlet> |  |   通过以上配置，Web容器会自动加载/WEB-INF/applicationContext.xml初始化  ApplicationContext实例，如果需要指定**配置文件位置**，可通过context-param加以指定：  <context-param><param-name>contextConfigLocation</param-name> <param-value>classpath:applicationContext-bean.xml,classpath:spring-jamon.xml</param-value> </context-param> 配置完成之后，即可通过  WebApplicationContextUtils.getWebApplicationContext方法在Web应用中获取ApplicationContext引用。  如：ApplicationContext ctx=WebApplicationContextUtils.getWebApplicationContext();           LoginAction action=(LoginAction)ctx.getBean("action");  -------------------------------------------------------------------------------------------  **spring为ApplicationContext提供有多种实现（举例）**     spring为ApplicationContext提供的3种实现分别为：ClassPathXmlApplicationContext，FileSystemXmlApplicationContext和XmlWebApplicationContext，其中XmlWebApplicationContext是专为Web工程定制的。使用举例如下：    **1. FileSystemXmlApplicationContext**      **eg1.**ApplicationContext ctx = new FileSystemXmlApplicationContext("bean.xml"); //加载单个配置文件      **eg2.**String[] locations = {"bean1.xml", "bean2.xml", "bean3.xml"};           ApplicationContext ctx = new FileSystemXmlApplicationContext(locations ); //加载多  个配置文件  **eg3.**ApplicationContext ctx =new FileSystemXmlApplicationContext("D:/project/bean.xml");//根据具体路径加载文件  注：  （1）这种方式适用于采用Spring框架的独立应用程序，需要程序通过配置文件手工初始化Spring的情况。  （2）获取失败时抛出异常。   **2. ClassPathXmlApplicationContext**       **eg1.**ApplicationContext ctx = new ClassPathXmlApplicationContext("bean.xml");       **eg2.**String[] locations = {"bean1.xml", "bean2.xml", "bean3.xml"};            ApplicationContext ctx = new ClassPathXmlApplication(locations); 注：  （1）.其中FileSystemXmlApplicationContext和ClassPathXmlApplicationContext与BeanFactory的xml文件定位方式一 样是基于路径的。  **3. XmlWebApplicationContext**     **eg1.**ServletContext servletContext = request.getSession().getServletContext();                 ApplicationContext ctx = WebApplicationContextUtils.getWebApplicationContext(servletContext);  注 :  （1）一般是 ApplicationContext ctx =  WebApplicationContextUtils.getWebApplicationContext(this.getServletContext());  （2）获取失败时返回null。 |   （3）这种方式适合于采用Spring框架的B/S系统，通过ServletContext对象获取ApplicationContext对象，然后在通过它获取需要的类实例。  **4.继承自抽象类ApplicationObjectSupport** 说明：抽象类ApplicationObjectSupport提供getApplicationContext()方法，可以方便的获取到ApplicationContext。 Spring初始化时，会通过该抽象类的setApplicationContext(ApplicationContext context)方法将ApplicationContext 对象注入。  **5．继承自抽象类WebApplicationObjectSupport** 说明：类似上面方法，调用getWebApplicationContext()获取WebApplicationContext  **6.实现接口ApplicationContextAware** 说明：实现该接口的setApplicationContext(ApplicationContext context)方法，并保存ApplicationContext 对象。 Spring初始化时，会通过该方法将ApplicationContext对象注入。  *实现方法：*  **public** **void** setApplicationContext(ApplicationContext arg0) **throws** BeansException {  applicationContext = arg0;  }  *获取bean:*  ITaskService bean = (ITaskService)applicationContext.getBean(taskServiceName);  **参考：**  1. http://hi.baidu.com/fyears/blog/item/e2f689afd05216ce7dd92a37.html  2. http://www.blogjava.net/Todd/archive/2010/04/22/295112.html |