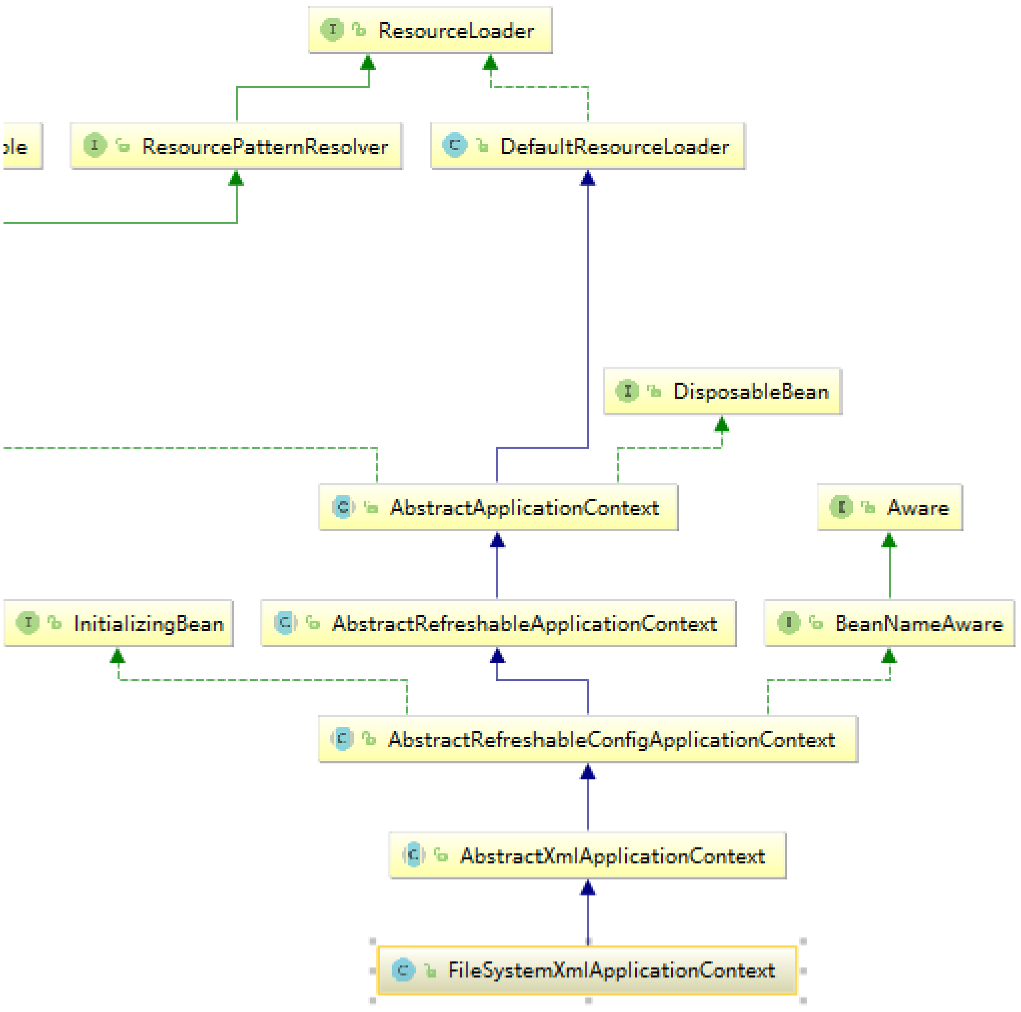
```
ApplicationContext applicationContext = new FileSystemXmlApplicationContext( configLocation: "xx.xml");
```

初始化开始，进入FileSystemXmlApplicationContext 构造函数

FileSystemXmlApplicationContext 的继承体系



ResourceLoader
ResourcePatternResolver
DefaultResourceLoader
DisposableBean
AbstractApplicationContext
Aware
InitializingBean
AbstractRefreshableApplicationContext
BeanNameAware
AbstractRefreshableConfigApplicationContext
AbstractXmlApplicationContext
FileSystemXmlApplicationContext

构造函数1：

```
public FileSystemXmlApplicationContext(String configLocation) throws BeansException {
    this(new String[]{configLocation},  refresh: true, (ApplicationContext)null);
}
```

转到构造函数2：

```
    public FileSystemXmlApplicationContext(String[] configLocations, boolean refresh, ApplicationContext parent)
throws BeansException {
        super(parent);
        this.setConfigLocations(configLocations);
        if (refresh) {
            this.refresh();
        }

    }
```

- 设置ApplicationContext 容器（调用的AbstractApplicationContext 里面的）

```
public AbstractApplicationContext(ApplicationContext parent) {
    this();
    this.setParent(parent);
}
```

- 设置路径（AbstractRefreshableConfigApplicationContext）

```
public void setConfigLocations(String[] locations) {
    if (locations != null) {
        Assert.noNullElements(locations, message: "Config locations must not be null");
        this.configLocations = new String[locations.length];

        for(int i = 0; i < locations.length; ++i) {
            this.configLocations[i] = this.resolvePath(locations[i]).trim();
        }
    } else {
        this.configLocations = null;
    }

}
```

这里代码的意思是分解路径，因为它可以传一个字符串“x.xml,c.xml1,a.xml”这种

```
this.configLocations[i] = this.resolvePath(locations[i]).trim();
```

二、 refresh() 方法

该方法在AbstractApplicationContext（）下

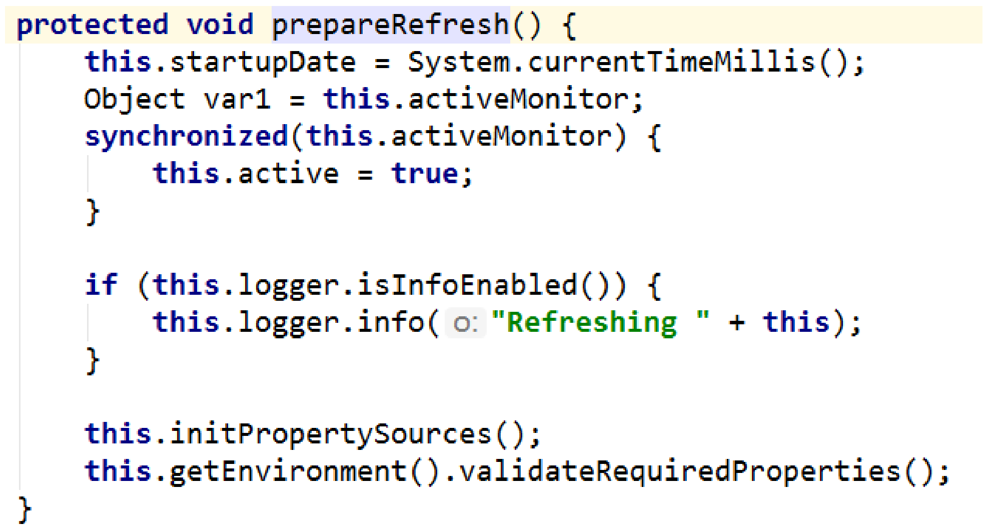
```
public void refresh() throws BeansException, IllegalStateException {
    Object var1 = this.startupShutdownMonitor;
    synchronized(this.startupShutdownMonitor) {
        this.prepareRefresh();
        ConfigurableListableBeanFactory beanFactory = this.obtainFreshBeanFactory();
        this.prepareBeanFactory(beanFactory);

        try {
            this.postProcessBeanFactory(beanFactory);
            this.invokeBeanFactoryPostProcessors(beanFactory);
            this.registerBeanPostProcessors(beanFactory);
            this.initMessageSource();
            this.initApplicationEventMulticaster();
            this.onRefresh();
            this.registerListeners();
            this.finishBeanFactoryInitialization(beanFactory);
            this.finishRefresh();
        } catch (BeansException var5) {
            this.destroyBeans();
            this.cancelRefresh(var5);
            throw var5;
        }

    }
}
```

2.1 prepareRefresh ( ) 方法

该方法主要是spring 初始化的前期准备

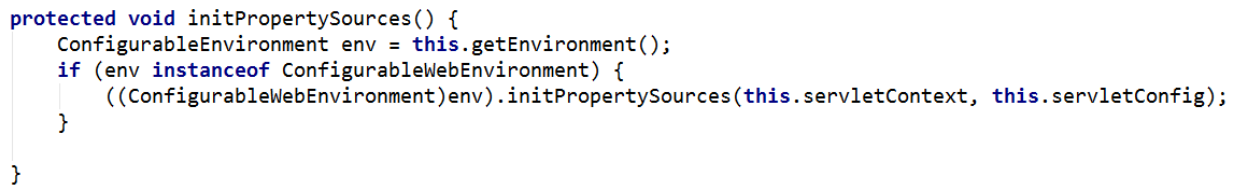


```
protected void prepareRefresh() {
    this.startupDate = System.currentTimeMillis();
    Object var1 = this.activeMonitor;
    synchronized(this.activeMonitor) {
        this.active = true;
    }

    if (this.logger.isInfoEnabled()) {
        this.logger.info( o: "Refreshing " + this);
    }

    this.initPropertySources();
    this.getEnvironment().validateRequiredProperties();
}
```

initPropertySources（） //AbstractRefreshableWebApplicationContext下设置servlet上下文



```
protected void initPropertySources() {
    ConfigurableEnvironment env = this.getEnvironment();
    if (env instanceof ConfigurableWebEnvironment) {
        ((ConfigurableWebEnvironment)env).initPropertySources(this.servletContext, this.servletConfig);
    }

}
```

getEnvironment().validateRequiredProperties()//确定必要的配置

2.2 obtainFreshBeanFactory()——获取ConfigurableListableBeanFactory