《Spring源码学习四：BeanDefinition装载前奏曲》中提到，对于非延迟单例bean的初始化在finishBeanFactoryInitialization(beanFactory)中完成。进入这个方法，代码如下。

```
protected void finishBeanFactoryInitialization(ConfigurableListableBeanFactory beanFactory) {
  // Initialize conversion service for this context.
  if (beanFactory.containsBean(CONVERSION_SERVICE_BEAN_NAME) &&
      beanFactory.isTypeMatch(CONVERSION_SERVICE_BEAN_NAME,
ConversionService.class)) {
    beanFactory.setConversionService(
        beanFactory.getBean(CONVERSION_SERVICE_BEAN_NAME,
ConversionService.class));
  }

  // Initialize LoadTimeWeaverAware beans early to allow for registering their transformers
early.
  String[] weaverAwareNames =
beanFactory.getBeanNamesForType(LoadTimeWeaverAware.class, false, false);
  for (String weaverAwareName : weaverAwareNames) {
    getBean(weaverAwareName);
  }

  // Stop using the temporary ClassLoader for type matching.
  beanFactory.setTempClassLoader(null);

  // Allow for caching all bean definition metadata, not expecting further changes.
  beanFactory.freezeConfiguration();

  // Instantiate all remaining (non-lazy-init) singletons.
  beanFactory.preInstantiateSingletons();
}
```

关注最后一行代码，beanFactory.preInstantiateSingletons()完成初始化所有非延迟的单例bean，进入这个方法的具体实现，代码如下。

```
public void preInstantiateSingletons() throws BeansException {
  if (this.logger.isDebugEnabled()) {
    this.logger.debug("Pre-instantiating singletons in " + this);
  }

  // Iterate over a copy to allow for init methods which in turn register new bean definitions.
  // While this may not be part of the regular factory bootstrap, it does otherwise work fine.
  List<String> beanNames = new ArrayList<String>(this.beanDefinitionNames);

  // Trigger initialization of all non-lazy singleton beans...
  for (String beanName : beanNames) {
    RootBeanDefinition bd = getMergedLocalBeanDefinition(beanName);
    if (!bd.isAbstract() && bd.isSingleton() && !bd.isLazyInit()) {
      if (isFactoryBean(beanName)) {
        final FactoryBean<?> factory = (FactoryBean<?>) getBean(FACTORY_BEAN_PREFIX +
beanName);
        boolean isEagerInit;
        if (System.getSecurityManager() != null && factory instanceof SmartFactoryBean) {
          isEagerInit = AccessController.doPrivileged(new PrivilegedAction<Boolean>() {
            @Override
            public Boolean run() {
              return ((SmartFactoryBean<?>) factory).isEagerInit();
            }
          }, getAccessControlContext());
        }else {
          isEagerInit = (factory instanceof SmartFactoryBean &&
              ((SmartFactoryBean<?>) factory).isEagerInit());
        }
        if (isEagerInit) {
          getBean(beanName);
        }
      }else {
        getBean(beanName);
      }
    }
  }

  // Trigger post-initialization callback for all applicable beans...
  for (String beanName : beanNames) {
    Object singletonInstance = getSingleton(beanName);
    if (singletonInstance instanceof SmartInitializingSingleton) {
      final SmartInitializingSingleton smartSingleton = (SmartInitializingSingleton)
singletonInstance;
      if (System.getSecurityManager() != null) {
        AccessController.doPrivileged(new PrivilegedAction<Object>() {
          @Override
          public Object run() {
            smartSingleton.afterSingletonsInstantiated();
            return null;
          }
        }, getAccessControlContext());
      }else {
        smartSingleton.afterSingletonsInstantiated();
      }
    }
  }
}
```

从上面的代码中看到，只会对非延迟单例bean进行初始化，scope为其它值的bean会在使用到的时候进行初始化，如prototype。这里关注getBean方法，这个方法看着很眼熟，其实就是《Spring源码学习一：源码分析概述》示例代码中用到的getBean，Spring对这个方法做了重复使用。getBean方法的具体实现在doGetBean方法中，这个方法的代码很长就不贴代码了。在doGetBean中，首先会初始化其依赖的bean，然后进行自身的初始化，这个方法里关注如下的代码段。

```
    // Create bean instance.
    if (mbd.isSingleton()) {
      sharedInstance = getSingleton(beanName, new ObjectFactory<Object>() {
        @Override
        public Object getObject() throws BeansException {
          try {
            return createBean(beanName, mbd, args);
          }
          catch (BeansException ex) {
            // Explicitly remove instance from singleton cache: It might have been put there
            // eagerly by the creation process, to allow for circular reference resolution.
            // Also remove any beans that received a temporary reference to the bean.
            destroySingleton(beanName);
            throw ex;
          }
        }
      });
      bean = getObjectForBeanInstance(sharedInstance, name, beanName, mbd);
    }
```

这段代码完成了单例bean的初始化，追踪代码进入doCreateBean方法中，在这个方法中进行bean实例的创建、属性填充、将bean实例加入单例bean实例的缓存中。doCreateBean方法中有如下代码段。

```
if (instanceWrapper == null) {
  instanceWrapper = createBeanInstance(beanName, mbd, args);
}
```

createBeanInstance方法里完成bean实例的创建，具体过程可继续追踪代码查看，其实就是使用反射进行实例对象的创建。