



深圳大学
Shenzhen University

第3讲

MVC软件体系结构模式及框架

软件体系结构与设计模式

Software Architecture & Design Pattern

深圳大学计算机与软件学院



主要内容

- ◆ **3.1 MVC体系结构模式**
- ◆ **3.2 框架**
- ◆ **3.3Spring环境配置**
- ◆ **3.4Spring Hello World 实例**



3.1 MVC体系结构模式

- MVC(Model-View-Controller, 模型-视图-控制器) 是一种架构模式。
- MVC架构模式最早是small talk语言研究团提出的应用于用户交互应用程序中。
- Smalltalk语言和Java语言有很多相似性, 都是面向对象语言, 后来, MVC被推荐为用于设计J2EE应用系统的体系结构。

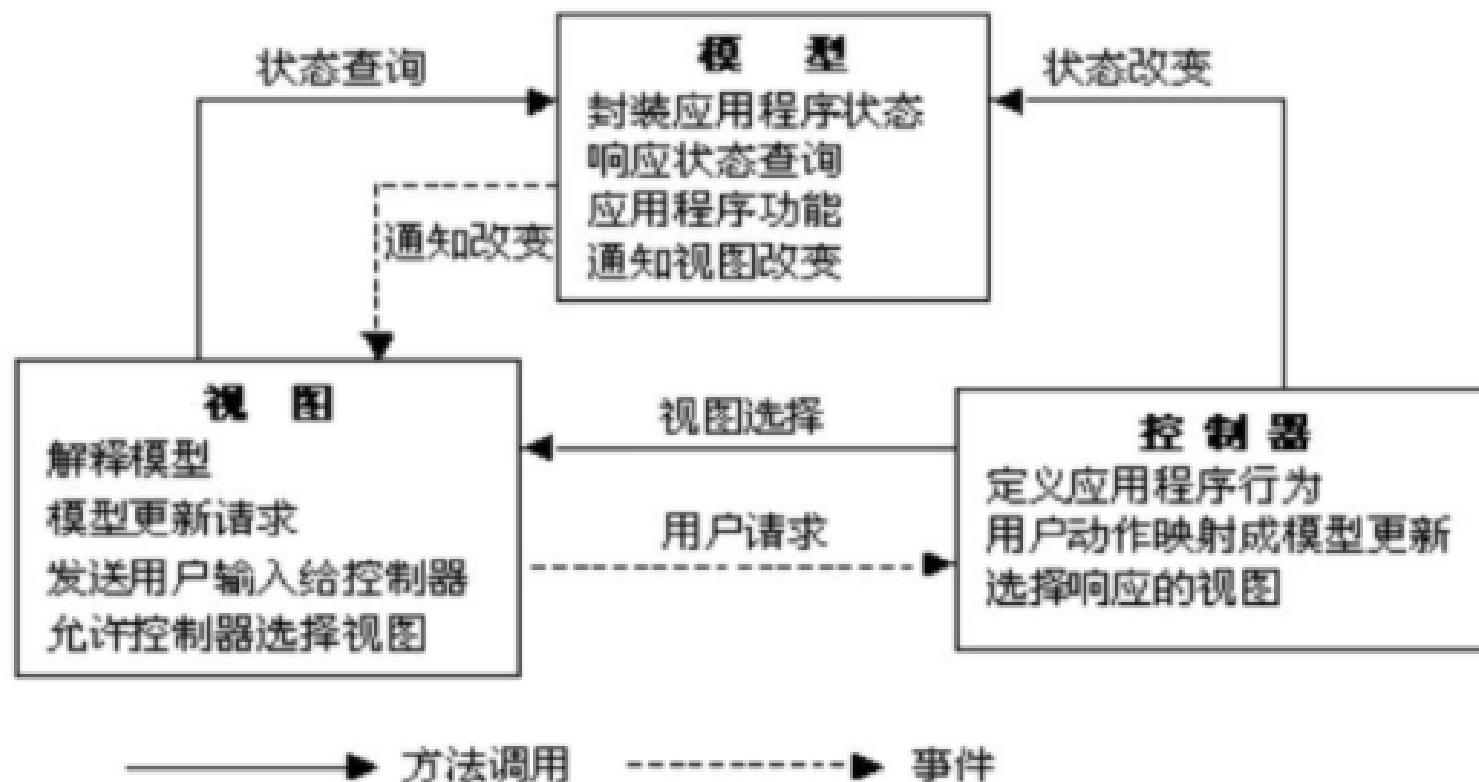




MVC设计思想

- MVC将一个交互式应用程序的输入、处理、输出流程按照**model**、**view**、**controller**的方式进行分离，这样一个应用被分成三个层组成：模型层、视图层和控制层。
- 模型：包含核心功能（业务逻辑）和数据。
- 视图：向用户显示信息。
- 控制器：处理用户输入，界面处理逻辑

MVC逻辑结构



MVC体系结构逻辑结构图

- Event事件导致controller改变model或view，或者同时改变两者。
- 只要controller改变了models的数据或者属性，所有依赖的view都会自动更新。
- 只要controller改变了view，view会从潜在的model中获取数据来刷新自己。



视图 (**view**)

- **view**代表用户交互界面，对于web应用来说可以概括为HTML界面，但有可能为XHTML、XML和applet，随着应用的复杂性和规模性，界面的处理也变得具有挑战性
- 一个应用可能有很多不同的视图，MVC对于视图的处理，仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上的业务流程的处理，业务流程的处理交给模型（model）处理



模型(model)

- 就是业务流程状态的处理以及业务规则的制定，业务流程的处理过程对其他层来说是黑箱操作，
- 模型接受视图请求的数据并返回最终的处理结果，业务模型的设计是mvc最主要的核心。
- EJB模型就是一个典型的应用例子，它从应用技术实现的角度对模型做了进一步的划分，以便充分利用现有的组件，但它不能作为应用设计模型的框架，它仅仅指出，按这种模型设计就可以利用某些技术组件从而减少了技术上的困难，对于一个开发者来说，就可以专注于业务模型的设计。
- Mvc要求把应用的模型按一定的规则抽取出来，抽取的层次很重要。这也是判断开发人员是否优秀的设计依据。抽象与具体不能隔得太远，也不能太近。
- Mvc并没有提供模型的设计方法，而只要求应该组织管理这些模型，以便于模型的重构和提高重用性。
- 就像：mvc定义了一个顶级类，告诉它的子类，你只能做这些，但没法限制你能做这些，这点对编程的开发人员非常重要



控制器(controller)

- 可以理解为从用户接收请求，将模型与视图匹配在一起，共同完成用户的请求。
- 划分控制层的作用是指明:它就是一个分发器，选择什么样的模型，选择什么样的视图，可以完成什么样的用户请求。控制层并不做任何的数据处理。
- 例如，用户点击一个链接，控制层接受请求后，并不处理业务信息，只把用户的信息传递给模型，告诉模型做什么，选择符合要求的视图返回给用户。因此一个模型可能对应多个视图，一个视图可能对应多个模型。
- 模型、视图与控制器的分离，使得一个模型可以具有多个显示视图。如果用户通过某个视图的控制器改变了模型的数据，所有其它依赖于这些数据的视图都应反映到这些变化。因此无论何时发生了何种数据变化，控制器都会将变化通知所有的视图，导致显示的更新。这实际上是一种模型的变化-传播机制。



优点:

- 对于同一个模型可以有不同的视图与控制器，以便提供给用户不同类型的用户图形界面
- 改变传播机制保证了模型在改变的同时自动刷新所有的视图，所有的视图都同时实时地反映了模型的现有状态
- **MVC**体系结构的设计使得改变用户图形界面变得非常容易，**MVC**非常适合业务逻辑较少改变，而用户图形界面需要经常改变的应用
- 由于全部的核心数据与核心功能在模型中，因此容易对核心的应用进行测试
- 可扩展性好



不足:

- View是可以直接访问model的
- View是依赖于model的
- 有一些业务逻辑在view里实现



三层体系结构VS MVC

■ 相同之处:

- 前者的显示层与MVC的view类似
- 前者的应用层与MVC的model类似

■ 区别:

- 各个模块之间的调用关系不同，三层体系结构中显示层不允许直接调用永久数据存储层，显示层需调用的应用层的方法访问永久数据层，而MVC体系结构的程序组件之间的交互是三角形的，两两之间可以直接交互
- 对数据库的访问方式不同，三层体系结构有永久数据存储层，负责对数据库的访问，而MVC中没有专门模块，一般情况可由Controller实现此功能
- MVC中有一个专门的controller模块，而层次体系结构中没有



MVC与J2EE的对应关系

- View处于Web Tier或者说是client tier，通常是JSP/servlet,即页面显示部分，
- Controller也处于Web Tier,通常用servlet来实现，即页面显示的逻辑部分实现，
- Model处于middle tier，通常用服务器端的JavaBean或者EJB实现及业务逻辑部分的实现



3.2 框架

- 软件体系结构确定了系统整体结构、层次划分以及不同部分之间的协作关系，是软件设计过程中的第一个层次，用于总体设计。
- **Framework（框架）** 不是现成可用的应用系统，而是一个半成品,需要后来的开发人员进行二次开发实现具有特定功能的应用系统。
- 框架构成了通用的、具有一般性的系统主体部分，二次开发人员只是像做填空一样，根据具体业务，完成特定应用系统中与众不同的特殊部分。
- 框架比体系结构更具体，更偏重于技术。
- 确定框架后，软件体系结构也随之确定，而对于同一软件体系结构，可以通过多种框架来实现。
- MVC不是框架, Spring 、 Struts ,SSM是框架



struts框架

- Struts框架只实现了MVC的View和Controller两个部分
- Model部分需要开发者自己来实现，
- Struts提供了抽象类Action，使开发者能将Model应用于Struts框架中。

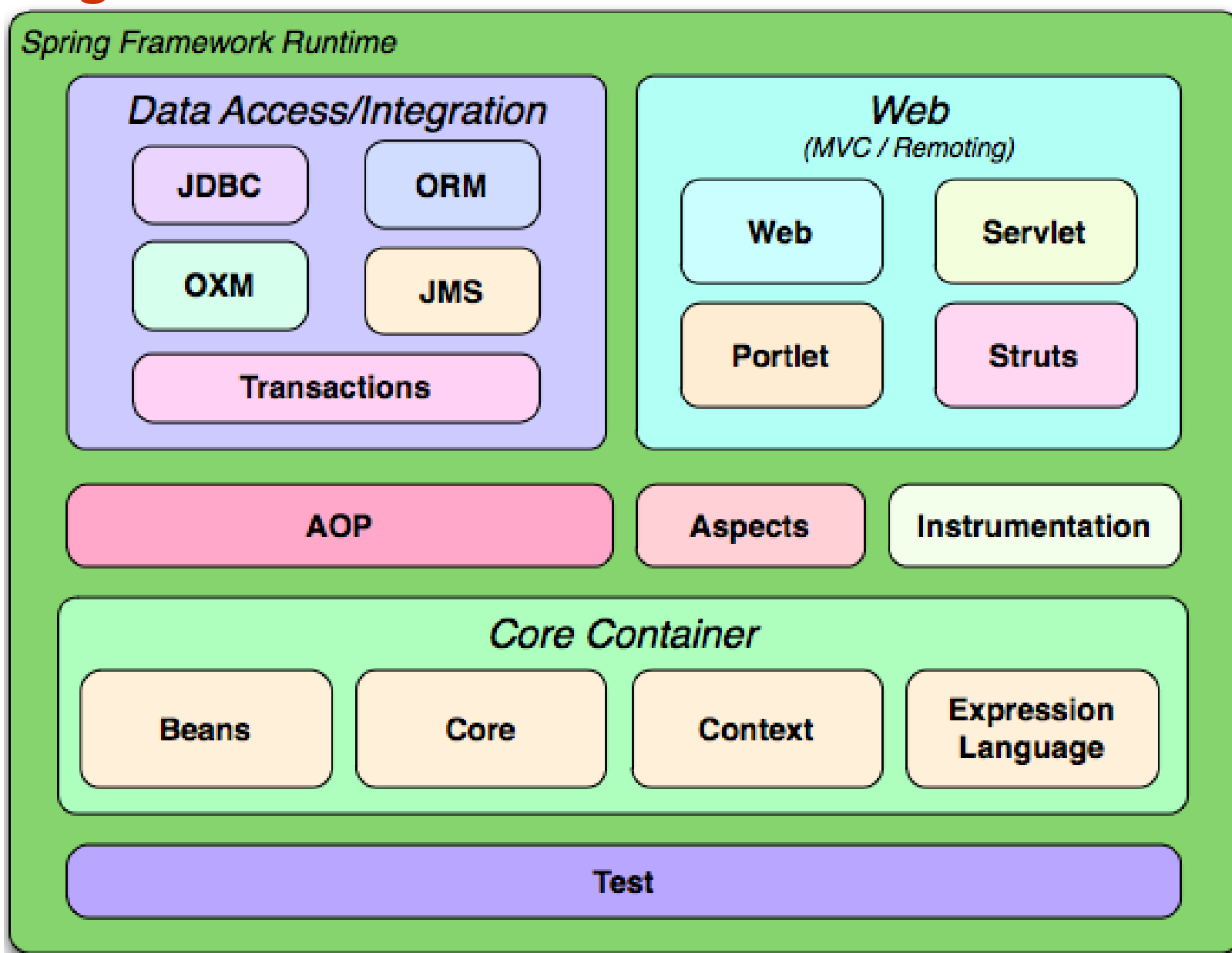


spring框架

- Spring是一个开源的轻量级的 J2EE开发框架
- Spring框架由Rod Johnson开发，2004年发布了Spring框架的第一版。
- Spring是一个从实际开发中抽取出来的框架，它完成了大量开发中的通用步骤，留给开发者的仅仅是与特定应用相关的部分，从而大大提高了企业应用的开发效率。
- 具有控制反转（IoC）和面向切面（AOP）两大核心。
- Java Spring 框架通过声明式方式灵活地进行事务的管理，提高开发效率和质量。



spring框架的组成结构



spring的核心机制

- 容器管理Bean:
- 程序主要是通过Spring容器来访问容器中的Bean, ApplicationContext是Spring容器最常用的接口, 该接口有如下两个实现类:
- ClassPathXmlApplicationContext: 从类加载路径下搜索配置文件, 并根据配置文件来创建Spring容器。
- FileSystemXmlApplicationContext: 从文件系统的相对路径或绝对路径下去搜索配置文件, 并根据配置文件来创建Spring容器。

```
public class BeanTest{  
    public static void main(String args[]) throws Exception{  
        ApplicationContext ctx = new ClassPathXmlApplicationContext("beans.xml");  
        Person p = ctx.getBean("person", Person.class);  
        p.say();  
    }  
}
```



Spring框架的核心功能

- 核心功能有两个：
- Spring容器作为超级大工厂，负责创建、管理所有的Java对象，这些Java对象被称为Bean。
- Spring容器管理容器中Bean之间的依赖关系，Spring使用一种被称为"依赖注入"的方式来管理Bean之间的依赖关系。
- 使用依赖注入，不仅可以为Bean注入普通的属性值，还可以注入其他Bean的引用。依赖注入是一种优秀的解耦方式，其可以让Bean以配置文件组织在一起，而不是以硬编码的方式耦合在一起。



AOP

- AOP（Aspect Orient Programming）面向切面编程，作为面向对象编程的一种补充，将程序运行过程分解成各个切面。
- AOP专门用于处理系统中分布于各个模块（不同方法）中的交叉关注点的问题，
- 在JavaEE应用中，常常通过AOP来处理一些具有横切性质的系统级服务，如事务管理、安全检查、缓存、对象池管理等。



AOP

- AOP（Aspect Orient Programming）面向切面编程，作为面向对象编程的一种补充，将程序运行过程分解成各个切面。
- AOP专门用于处理系统中分布于各个模块（不同方法）中的交叉关注点的问题，
- 在JavaEE应用中，常常通过AOP来处理一些具有横切性质的系统级服务，如事务管理、安全检查、缓存、对象池管理等。
- **AOP的一些术语：**
 - 切面（Aspect）：切面用于组织多个Advice，Advice放在切面中定义。
 - 连接点（Joinpoint）：程序执行过程中明确的点，如方法的调用，或者异常的抛出。在Spring AOP中，连接点总是方法的调用。
 - 增强处理（Advice）：AOP框架在特定的切入点执行的增强处理。处理有"around"、"before"和"after"等类型
 - 切入点（Pointcut）：可以插入增强处理的连接点。简而言之，当某个连接点满足指定要求时，该连接点将被添加增强处理，该连接点也就变成了切入点。



使用AspectJ实现AOP

- AspectJ是一个基于Java语言的AOP框架，提供了强大的AOP功能，其他很多AOP框架都借鉴或采纳其中的一些思想。
- 其主要包括两个部分：一个部分定义了如何表达、定义AOP编程中的语法规则，通过这套语法规则，可以方便地用AOP来解决Java语言中存在的交叉关注点的问题；另一个部分是工具部分，包括编译、调试工具等。
- **AOP实现可分为两类：**
- 静态AOP实现: AOP框架在编译阶段对程序进行修改，即实现对目标类的增强，生成静态的AOP代理类，以AspectJ为代表。
- 动态AOP实现: AOP框架在运行阶段动态生成AOP代理，以实现为目标对象的增强，以Spring AOP为代表。
- 一般来说，静态AOP实现具有较好的性能，但需要使用特殊的编译器。动态AOP实现是纯Java实现，因此无须特殊的编译器，但是通常性能略差。

Spring的AOP支持

- Spring中的AOP代理由Spring的IoC容器负责生成、管理，其依赖关系也由IoC容器负责管理。
- 为了在应用中使用@AspectJ支持，Spring需要添加三个库：
- aspectjweaver.jar
- aspectjrt.jar
- aopalliance.jar
- 并在Spring配置文件中做如下配置：

```
<!--启动@AspectJ支持-->
```

```
<aop:aspectj-autoproxy/>
```

```
<!--指定自动搜索Bean组件、自动搜索切面类-->
```

```
<context:component-scan base-package="edu.shu.sprint.service">
```

```
    <context:include-filter type="annotation" expression="org.aspectj.lang.annotation.Aspect"/>
```

```
</context:component-scan>
```



Eclipse使用Spring

- 在Eclipse等IDE工具中，用户可以自建User Library，然后把Spring的Jar包都放入其中，
- 也可以将Jar包直接放在项目的/WEB-INF/lib目录下，
- 如果使用User Library，在项目发布时，需要将用户库所引用的Jar文件随应用一起发布，就是将User Library所使用的Jar复制到/WEB-INF/lib目录下，这是因为对于一个Web应用，Eclipse部署Web应用时不会将用户库的Jar文件复制到/WEB-INF/lib下，需要手动复制。



Spring的优点

- 低侵入式设计，代码的污染极低。
- 独立于各种应用服务器，基于Spring框架的应用，可以真正实现Write Once, Run Anywhere的承诺。
- Spring的IoC容器降低了业务对象替换的复杂性，提高了组件之间的解耦。
- Spring的AOP支持允许将一些通用任务如安全、事务、日志等进行集中式管理，从而提供了更好的复用。
- Spring的ORM和DAO提供了与第三方持久层框架的良好整合，并简化了底层的数据库访问。
- Spring的高度开放性，并不强制应用完全依赖于Spring，开发者可自由选用Spring框架的部分或全部。



3.3Spring环境配置

- 指导你如何准备开发环境来使用 **Spring** 框架开始你的工作。
- 教你在安装 **Spring** 框架之前如何在你的机器上安装 **JDK**，**Tomcat** 和 **Eclipse**。

第 1 步：安装 Java 开发工具包（JDK）

- 你可以从 Oracle 的 Java 网站 Java SE Downloads 下载 JDK 的最新版本。
- 你会在下载的文件中找到教你如何安装 JDK 的说明，按照给出的说明安装和配置 JDK 的设置。最后，设置 PATH 和 JAVAHOME 环境变量，引入包含 java 和 javac 的目录，通常分别为 java install dir/bin 和 java install _ dir。
- 如果你运行的是 Windows，并在 C:\jdk1.6.0_15 上安装了 JDK，你就可以把下面这行写入 C:\autoexec.bat 文件中。

```
set PATH=C:\jdk1.6.0_15\bin;%PATH%  
set JAVA_HOME=C:\jdk1.6.0_15
```

- 或者，在 Windows 中，你也可以右键单击“我的电脑”，选择“属性”，然后是“高级”，然后是“环境变量”，进行设置。

第 2 步：安装 Apache Commons Logging API

- Apache创建的第三方日志库
- 你可以从 <http://commons.apache.org/logging/> 下载 Apache Commons Logging API 的最新版本。一旦你下载完安装包，并且解压二进制的发行版本到一个方便的位置。例如在 windows 上的 C:\commons-logging-1.1.1 中。该目录将有如下的 jar 文件和其他支持的文件等。
- 确保你在这个目录上正确的设置 CLASSPATH 变量，否则你将会在运行应用程序时遇到问题。

Name	Date modified	Type	Size
site	11/22/2007 12:28 ...	File folder	
commons-logging-1.1.1	11/22/2007 12:28 ...	WinRAR archive	60 KB
commons-logging-1.1.1-javadoc	11/22/2007 12:28 ...	WinRAR archive	139 KB
commons-logging-1.1.1-sources	11/22/2007 12:28 ...	WinRAR archive	74 KB
commons-logging-adapters-1.1.1	11/22/2007 12:28 ...	WinRAR archive	26 KB
commons-logging-api-1.1.1	11/22/2007 12:28 ...	WinRAR archive	52 KB
commons-logging-tests	11/22/2007 12:28 ...	WinRAR archive	109 KB
LICENSE	11/22/2007 12:27 ...	Text Document	12 KB
NOTICE	11/22/2007 12:27 ...	Text Document	1 KB
RELEASE-NOTES	11/22/2007 12:27 ...	Text Document	8 KB

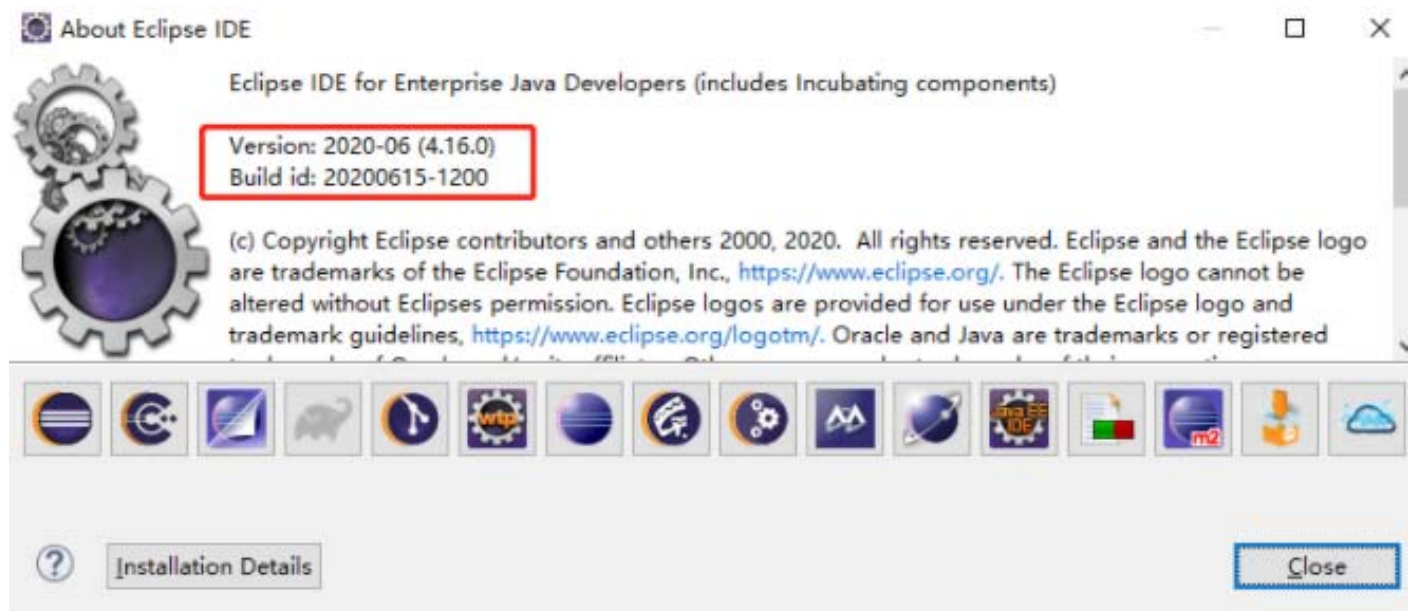
第 3 步：安装 Eclipse IDE

- 你可以在你的机器上安装 Eclipse 的最新版本。
- 为了安装 Eclipse IDE，点开网址 <http://www.eclipse.org/downloads/>，根据操作系统位数（32位或64位）下载相应的版本。



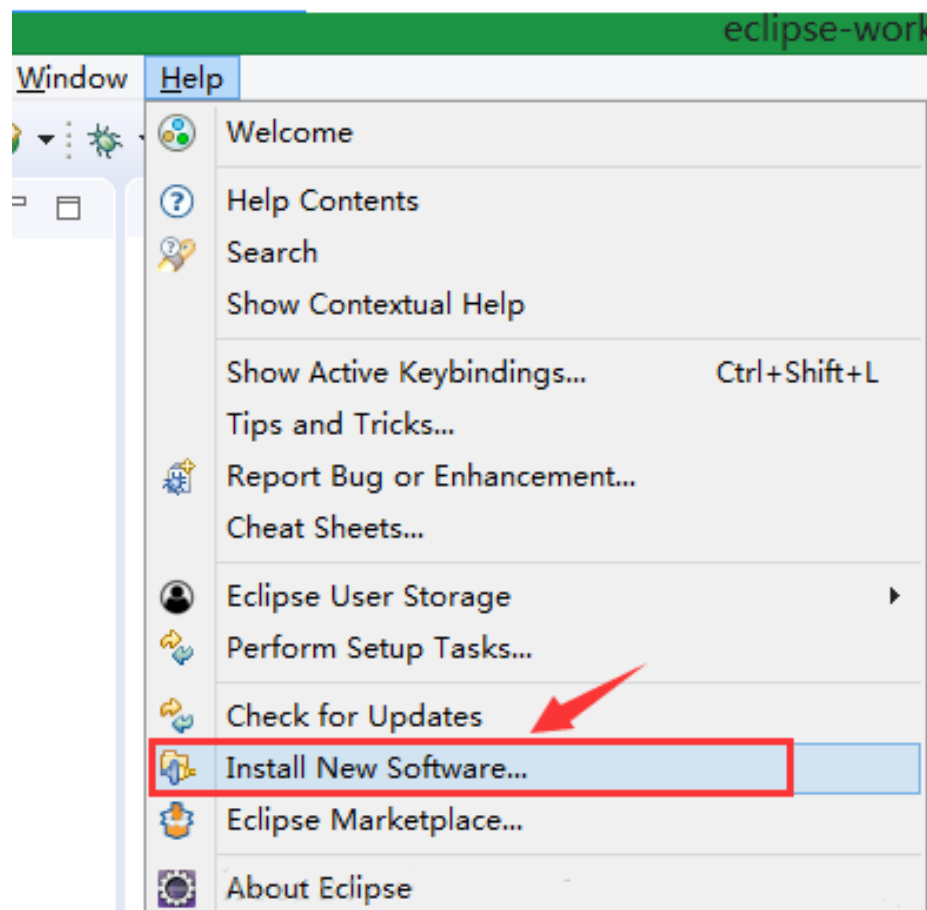
第 4 步：安装 Spring 框架库

- 如果前三个步骤一切正常，你就可以继续设置你的 Spring 框架。下面是在电脑上下载并安装框架的简单步骤。
- 选择是要在 Windows 还是在 UNIX 上安装 Spring，然后继续进行下一个步骤，在 Windows 上下载 .zip 文件,在 Unix 上安装则下载 .tz 文件。
- 从 Spring 官方下载对应 Eclipse 版本的 springsource-tool-suite。

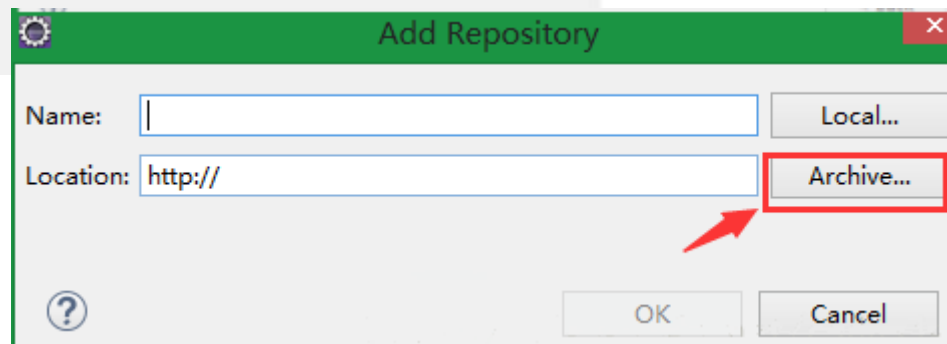
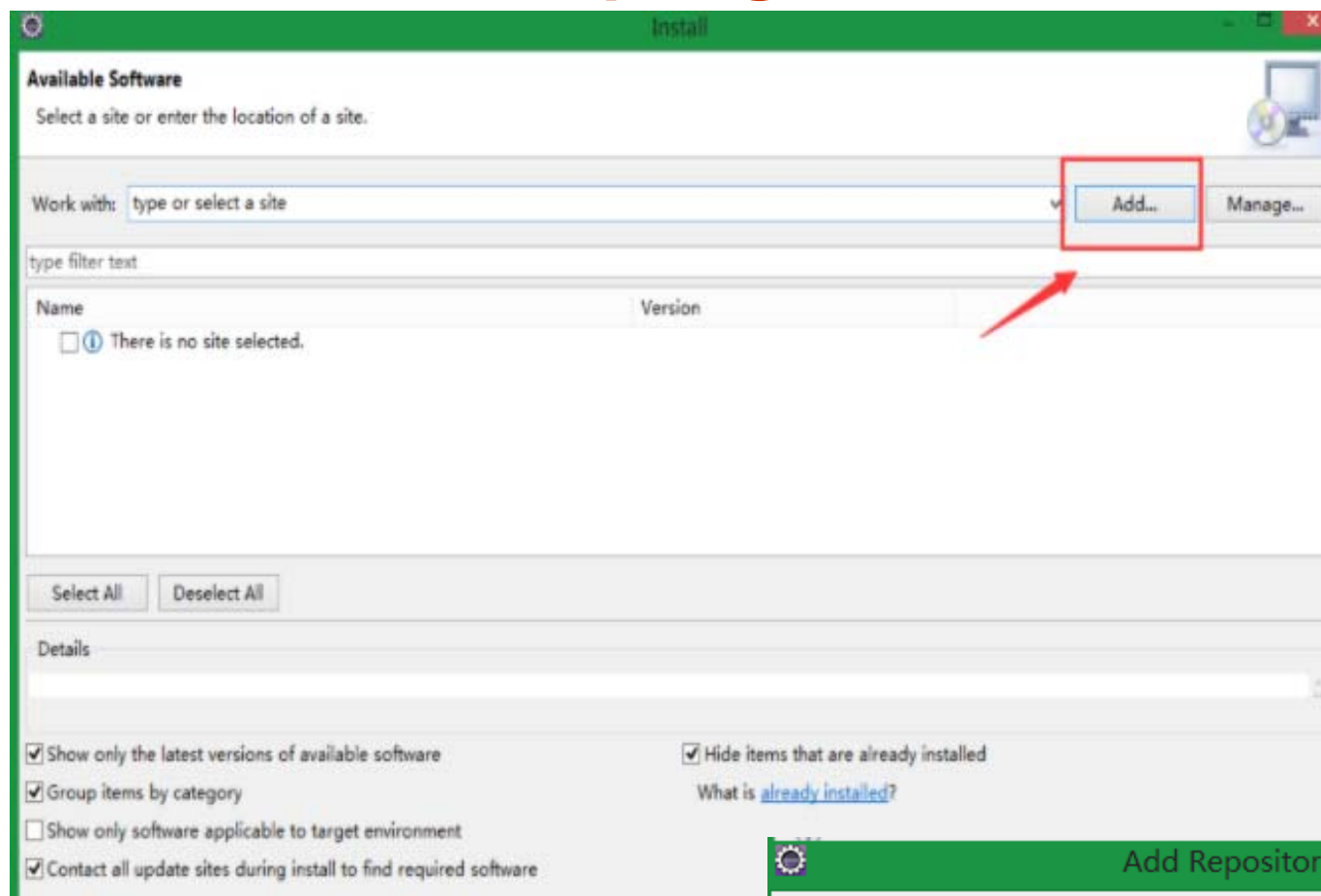


第 4 步：安装 Spring 框架库

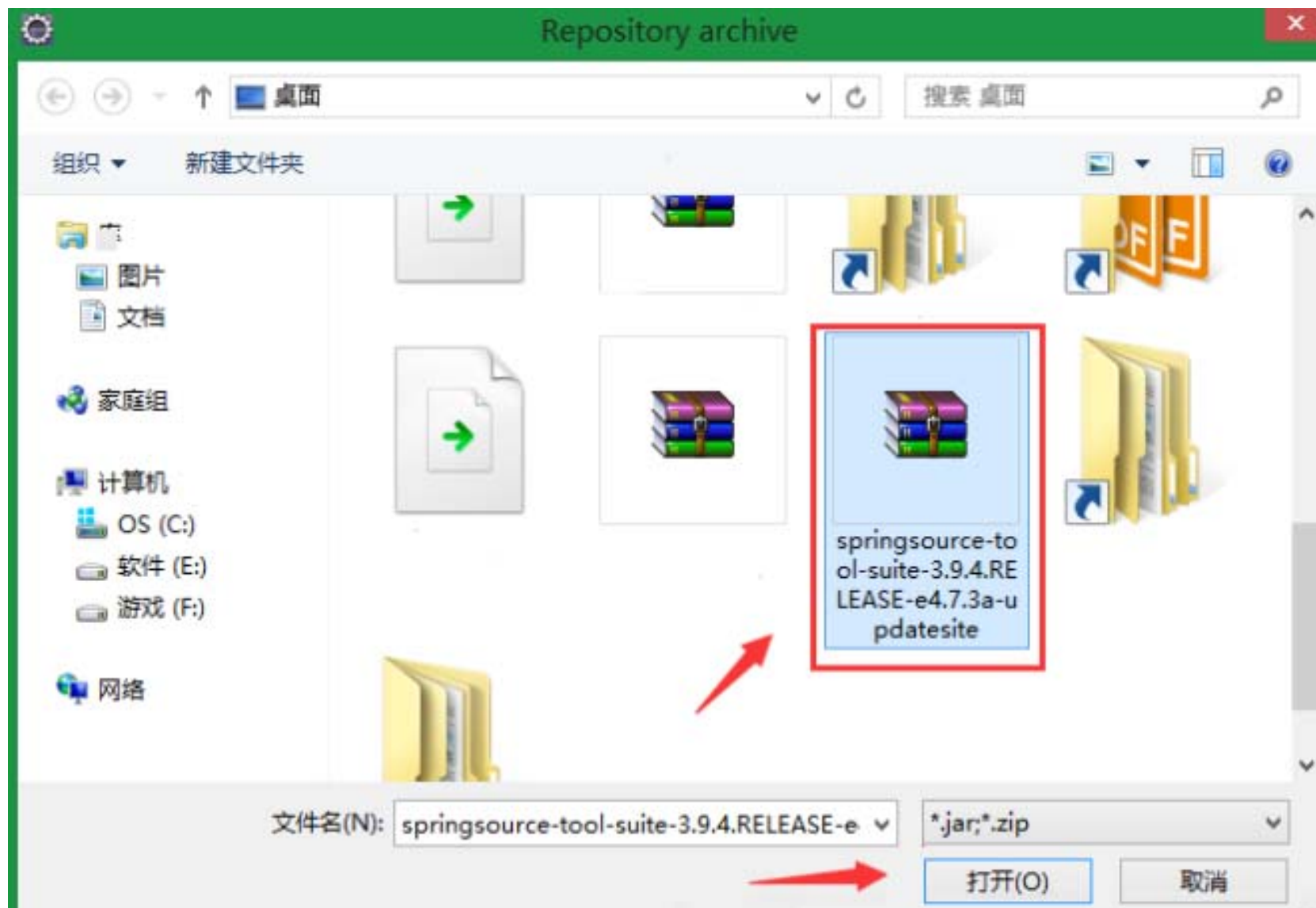
- 安装 springsource-tool-suite
- Eclipse–Help–Install New Software–Add–Archive-- 选择刚才下载好的 springsource-tool-suite 压缩包 --打开 –OK



第 4 步：安装 Spring 框架库

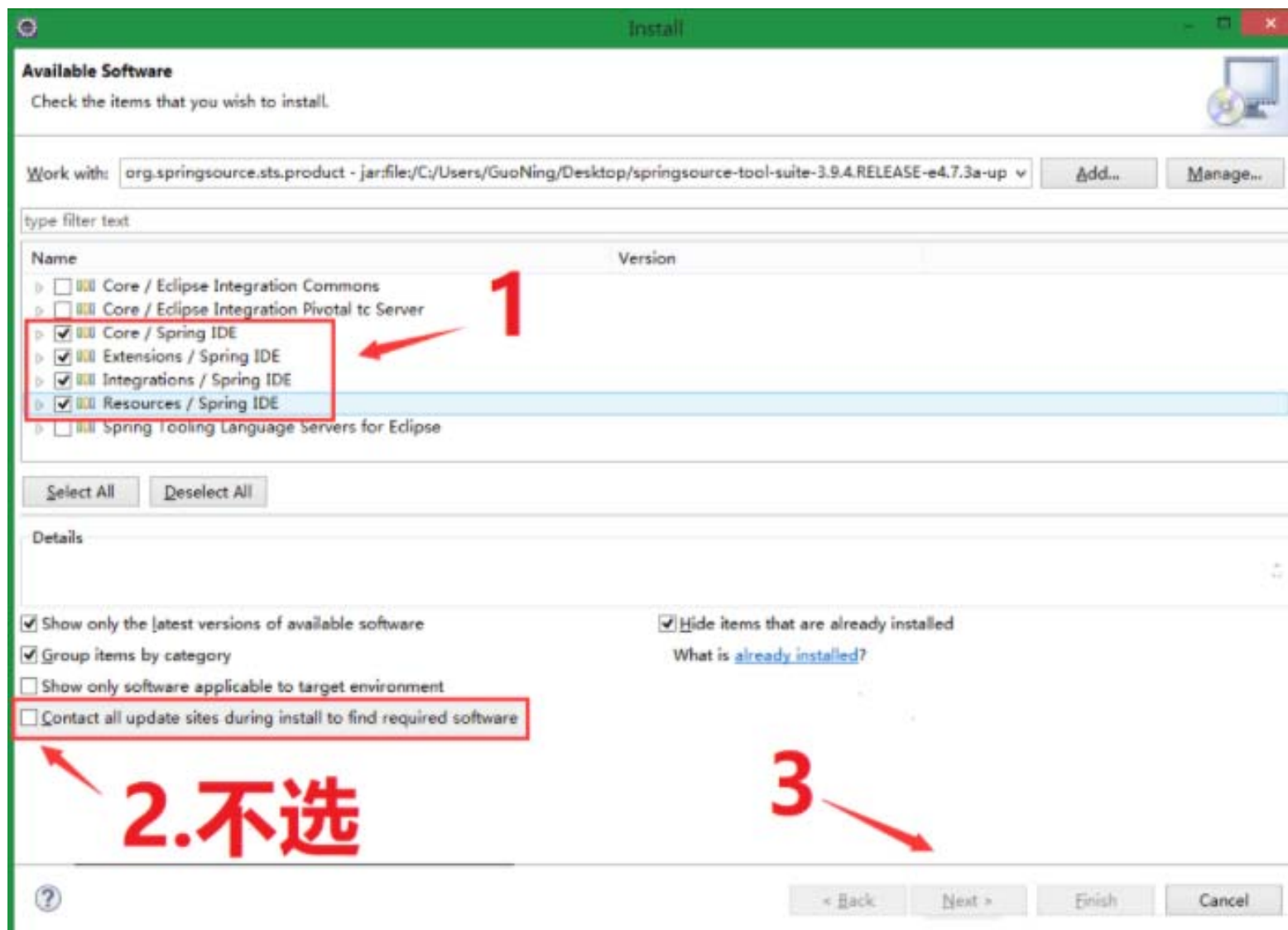


第 4 步：安装 Spring 框架库



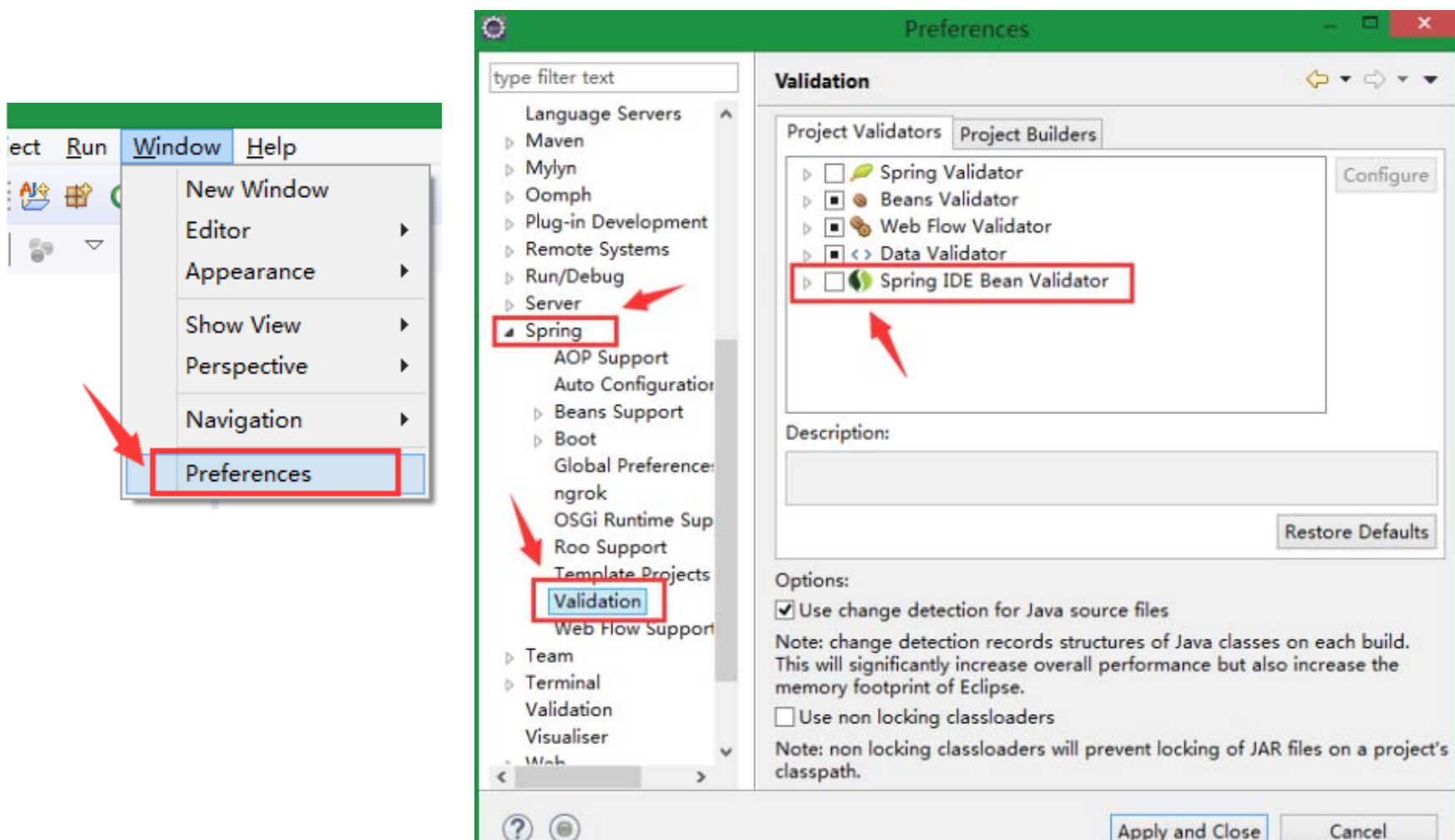
第 4 步：安装 Spring 框架库

- 选择要安装的组件：选定 4 个标有“IDE”的组件：



第 4 步：安装 Spring 框架库

- 一路点 “Next”，最后 “Finish”。等待安装。安装完成后在弹出对话框中点 “yes” 重启 eclipse。顺利的话这时你已经安装好 Spring 插件，点击 “Window”--> “Preferences”查看是否安装成功，看到下图两片绿叶的图标，说明已经安装成功：



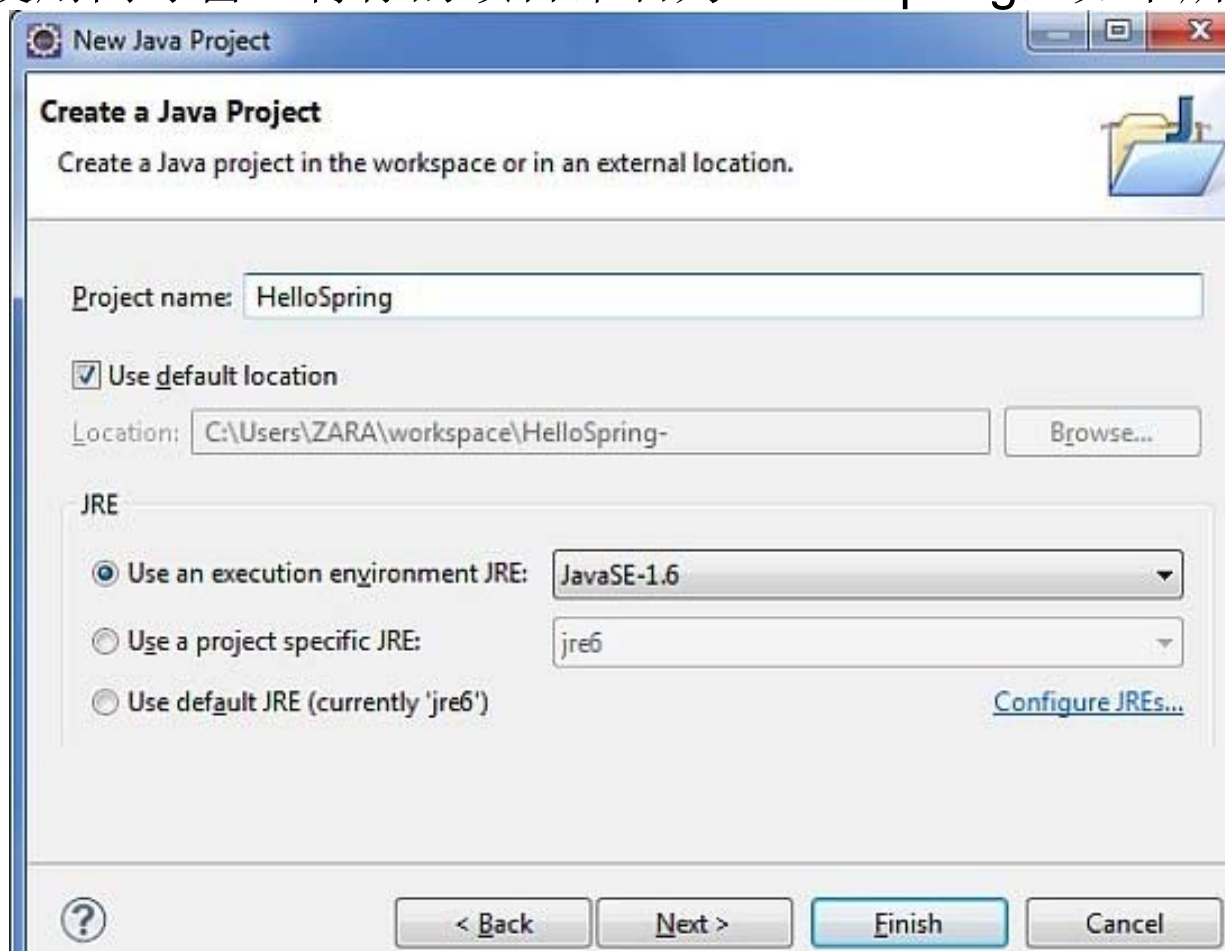


3.4Spring Hello World 实例

- 使用 Spring 框架开始实际的编程。
- 在你开始使用 Spring 框架编写第一个例子之前，你必须确保已经正确地设置了 Spring 环境
- 编写一个简单的 Spring 应用程序，它将根据在 Spring Beans 配置文件中配置的信息输出 “Hello World!” 或其他信息。

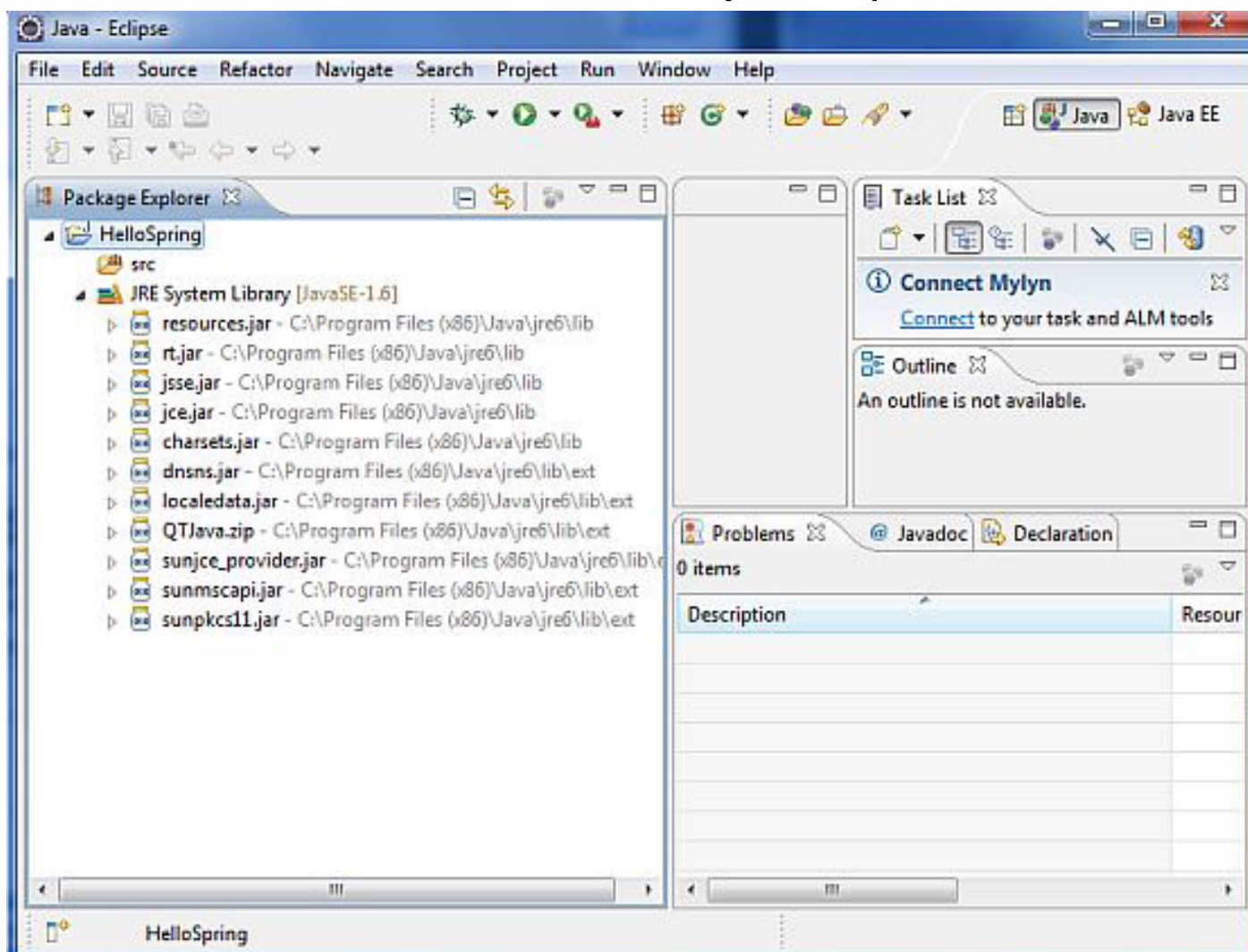
第 1 步：创建 Java 项目

- 第一步是使用 Eclipse IDE 创建一个简单的 Java 项目。按照选项 File -> New -> Project，最后从向导列表中选择 Java Project 向导。现在，使用向导窗口将你的项目命名为 HelloSpring，如下所示：



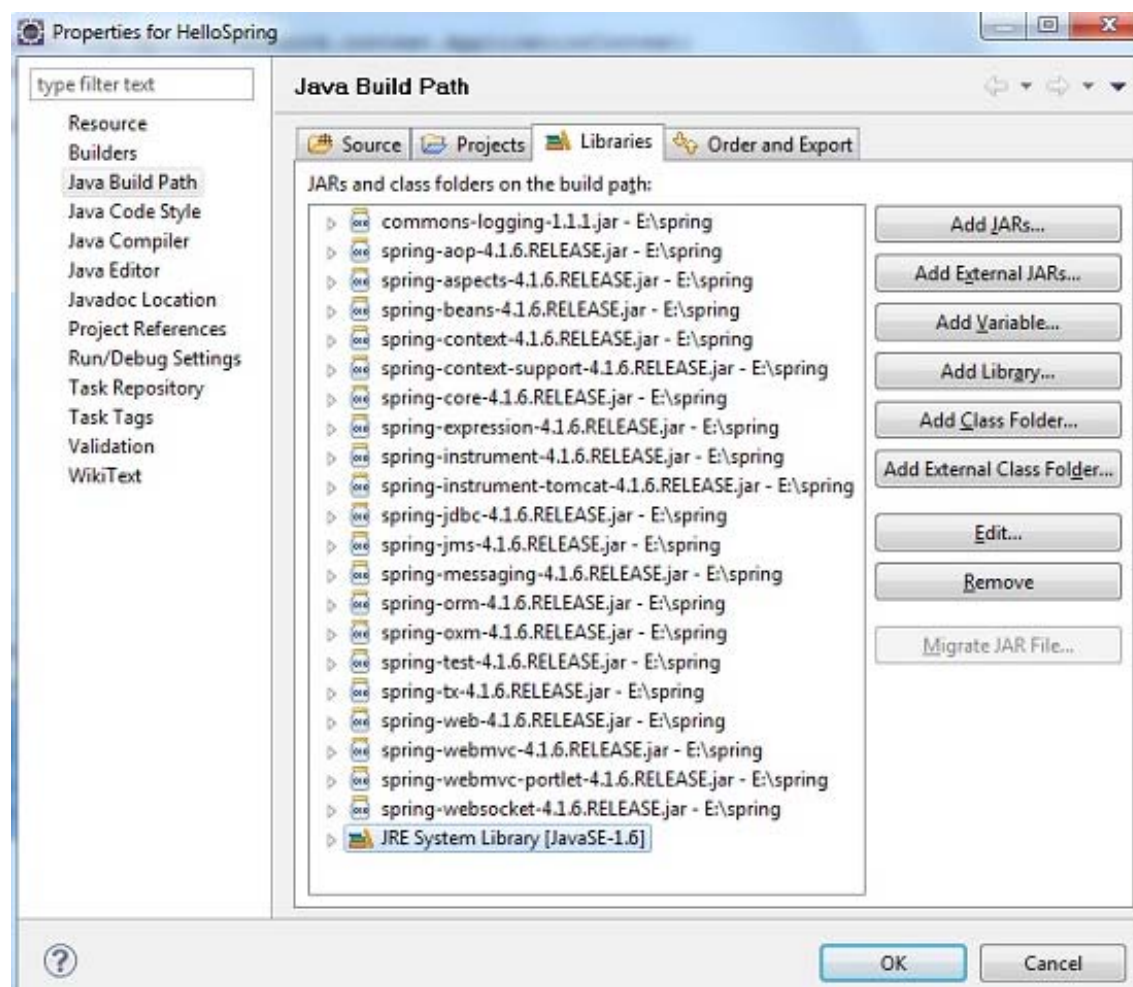
第 1 步：创建 Java 项目

- 一旦你的项目创建成功后，将在 Project Explorer 看到下面的内容：



第 2 步：添加必需的库

- 第二步添加 Spring 框架和通用的日志 API 库到我们的项目中。在你的项目名称 HelloSpring 上单击右键，然后在快捷菜单上按照下面可用的选项：Build Path -> Configure Build Path 显示 Java 构建路径窗口，如下所示：



第 2 步：添加必需的库

- 在 Libraries 标签中使用可用的 Add External JARs 按钮，添加从 Spring 框架和通用日志安装目录下面的核心 JAR 文件：

commons-logging-1.1.1

spring-aop-4.1.6.RELEASE

spring-aspects-4.1.6.RELEASE

spring-beans-4.1.6.RELEASE

spring-context-4.1.6.RELEASE

spring-context-support-4.1.6.RELEASE

spring-core-4.1.6.RELEASE

spring-expression-4.1.6.RELEASE

spring-instrument-4.1.6.RELEASE

spring-instrument-tomcat-4.1.6.RELEASE

spring-jdbc-4.1.6.RELEASE

spring-jms-4.1.6.RELEASE

spring-messaging-4.1.6.RELEASE

spring-orm-4.1.6.RELEASE

spring-oxm-4.1.6.RELEASE

spring-test-4.1.6.RELEASE

spring-tx-4.1.6.RELEASE

spring-web-4.1.6.RELEASE

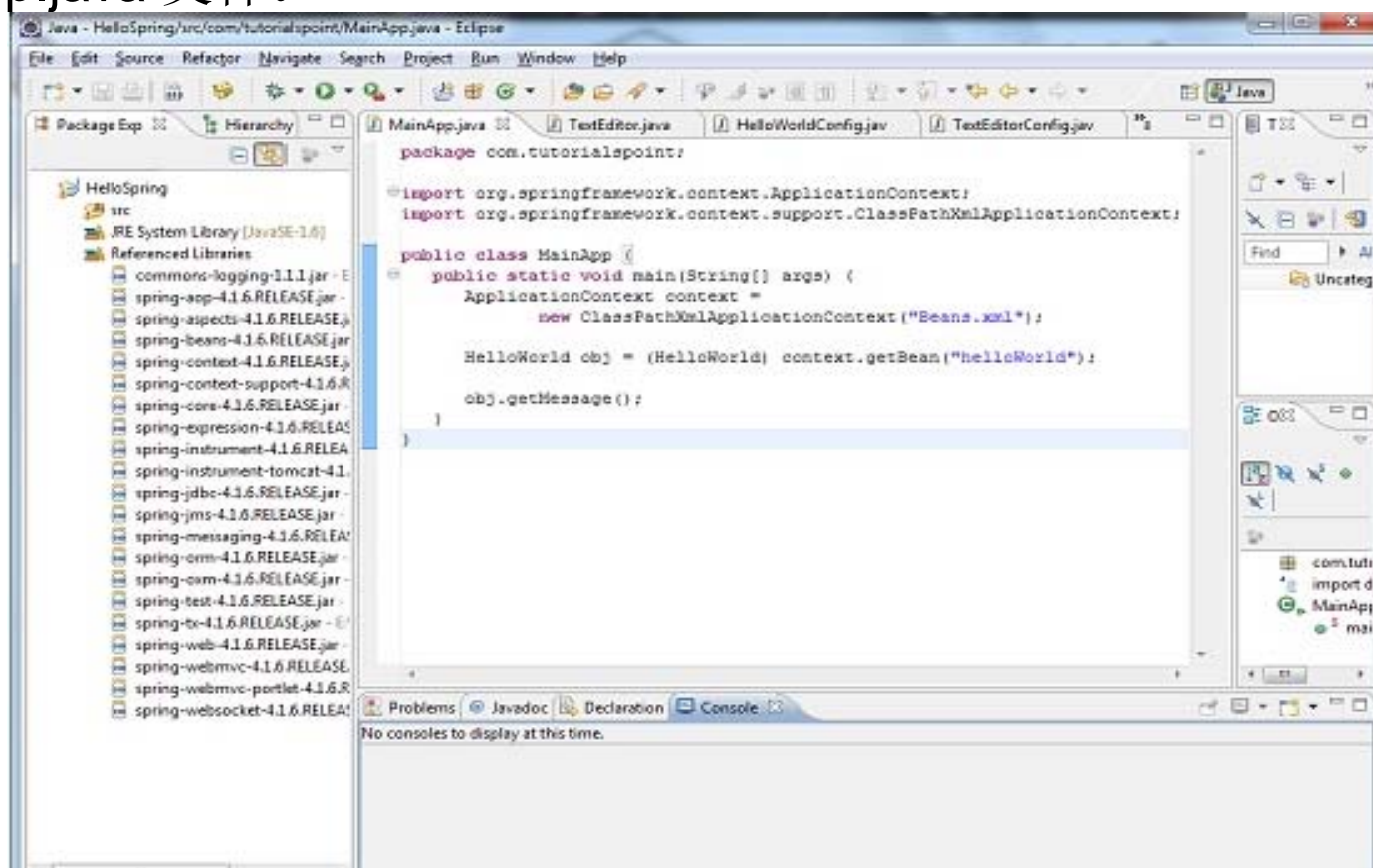
spring-webmvc-4.1.6.RELEASE

spring-webmvc-portlet-4.1.6.RELEASE

spring-websocket-4.1.6.RELEASE

第 3 步：创建源文件

- 在 HelloSpring 项目下创建实际的源文件。首先，我们需要创建一个名为 `com.tutorialspoint` 的包。在 `package explore` 区域中的 `src` 上点击右键，并按照选项：New -> Package。
- 接下来，在包 `com.tutorialspoint` 下创建 `HelloWorld.java` 和 `MainApp.java` 文件。



第 3 步：创建源文件

- 这里是 HelloWorld.java 文件的内容：

```
package com.tutorialspoint;
public class HelloWorld {
    private String message;
    public void setMessage(String message){
        this.message = message;
    }
    public void getMessage(){
        System.out.println("Your Message : " + message);
    }
}
```

第 3 步：创建源文件

- MainApp.java 的内容：

```
package com.tutorialspoint;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context =
            new ClassPathXmlApplicationContext("Beans.xml");
        HelloWorld obj = (HelloWorld) context.getBean("helloWorld");
        obj.getMessage();
    }
}
```

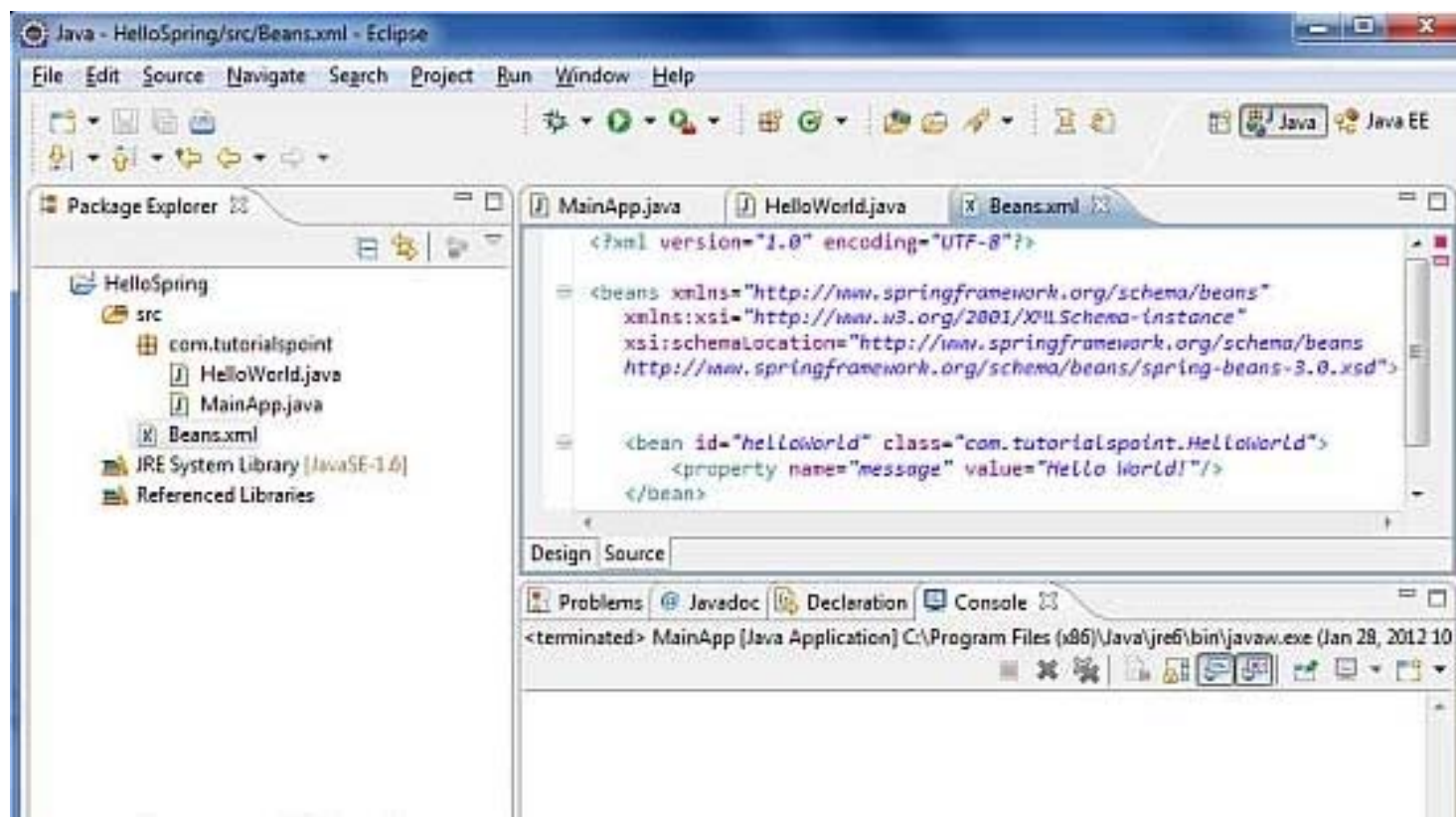


第 3 步：创建源文件

- 关于主要程序有以下两个要点需要注意：
- 第一步是我们使用框架 API `ClassPathXmlApplicationContext()` 来创建应用程序的上下文。这个 API 加载 beans 的配置文件并最终基于所提供的 API，它处理创建并初始化所有的对象，即在配置文件中提到的 beans。
- 第二步是使用已创建的上下文的 `getBean()` 方法来获得所需的 bean。这个方法使用 bean 的 ID 返回一个最终可以转换为实际对象的通用对象。一旦有了对象，你就可以使用这个对象调用任何类的方法。

第 4 步：创建 bean 的配置文件

- 你需要创建一个 Bean 的配置文件，该文件是一个 XML 文件，并且作为粘合 bean 的粘合剂即类。这个文件需要在 src 目录下创建，如下图所示：





第 4 步：创建 **bean** 的配置文件

- 通常开发人员保存该文件的名称为 **Beans.xml** 文件，也可以设置成任何你喜欢的名称。但是必须确保这个文件在 **CLASSPATH** 中是可用的，并在主应用程序中使用相同的名称，而在 **MainApp.java** 文件中创建应用程序的上下文。
- **Beans.xml** 用于给不同的 **bean** 分配唯一的 **ID**，并且控制不同值的对象的创建，而不会影响 **Spring** 的任何源文件。
- 例如，使用下面的文件，你可以为 “**message**” 变量传递任何值，因此你就可以输出信息的不同值，而不会影响的 **HelloWorld.java**和**MainApp.java** 文件。

第 4 步：创建 bean 的配置文件

- 例如，使用下面的文件，你可以为 “message” 变量传递任何值，因此你就可以输出信息的不同值，而不会影响的 HelloWorld.java 和 MainApp.java 文件。让我们来看看它是如何工作的：
- 当 Spring 应用程序被加载到内存中时，框架利用了上面的配置文件来创建所有已经定义的 beans，并且按照标签的定义为它们分配一个唯一的 ID。你可以使用标签来传递在创建对象时使用不同变量的值。

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id="helloWorld" class="com.tutorialspoint.HelloWorld">
        <property name="message" value="Hello World!"/>
    </bean>

</beans>
```

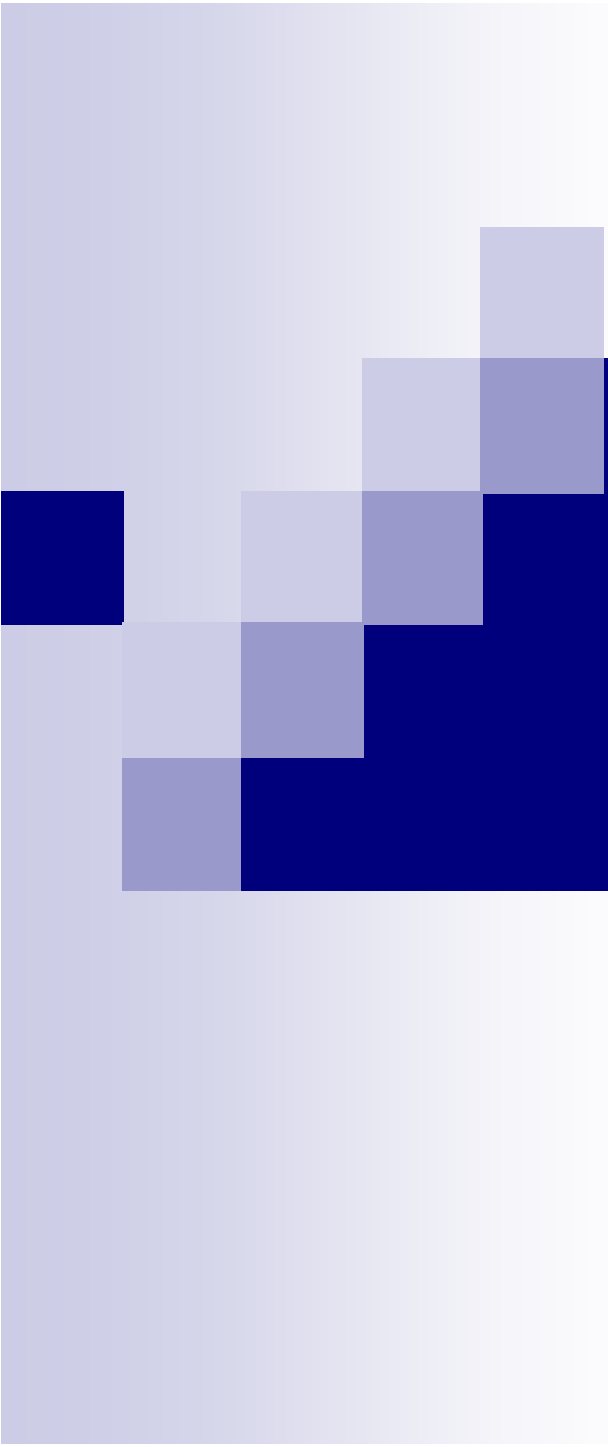


第 5 步：运行程序

- 一旦你完成了创建源代码和 **bean** 的配置文件后，就可以准备编译和运行你的程序了。
- 请保持 **MainApp.Java** 文件标签是有效的，并且在 **Eclipse IDE** 中使用可用的 **Run** 选项，或使用 **Ctrl + F11** 编译并运行你的应用程序 **MainApp**。
- 如果你的应用程序一切都正常，将在 **Eclipse IDE** 控制台打印以下信息：

```
Your Message : Hello World!
```

- 恭喜你，你已经成功地创建了你的第一个 **Spring** 应用程序。通过更改“**message**”属性的值并且保持两个源文件不变，你可以看到上述 **Spring** 应用程序的灵活性。



谢谢