JSP简介

JSP介绍

JSP(全称Java Server Pages)Java服务端页面技术，是JavaEE平台下的技术规范。它允许使用特定的标签在HTML网页中插入Java代码，实现动态页面处理，所以JSP就是HTML与Java代码的复合体。JSP技术可以快速的实现一个页面的开发，相比在Servlet中实现页面开发将变得更加容易。

常见的视图层技术

HTML、JSP、Thymeleaf等。

前后端分离开发方式

在前后端分离的项目中真正可以做到“术业有专攻”（开发人员分离） 。前后端分离开发方式中前端页面由专业团队完成页面的开发，并通过请求调用后端的api接口进行数据交互。在开发前端页面的团队中更多关注的技术如：html、CSS、jQuery、Vue、Nodejs等前端技术。前端追求的是：页面表现，速度流畅，兼容性，用户体验等等。而后端团队则更多的是业务的具体实现。在后端开发的团队中更多关注的技术如：设计模式、分布式架构、微服务架构、数据库的操作、Java的性能优化以及数据库优化等技术。前后端分离已成为互联网项目开发的业界标准使用方式，特别是为大型分布式架构、弹性计算架构、微服务架构、多端化服务（多种客户端，例如：浏览器，车载终端，安卓，IOS等等）打下坚实的基础。

实时效果反馈

1.JSP是哪两种技术的复合?

A HTML与JS

B HTML与JAVA

C HTML与CSS

D HTML与C#

2.如下哪些技术不是视图层技术?

A HTML

B JSP

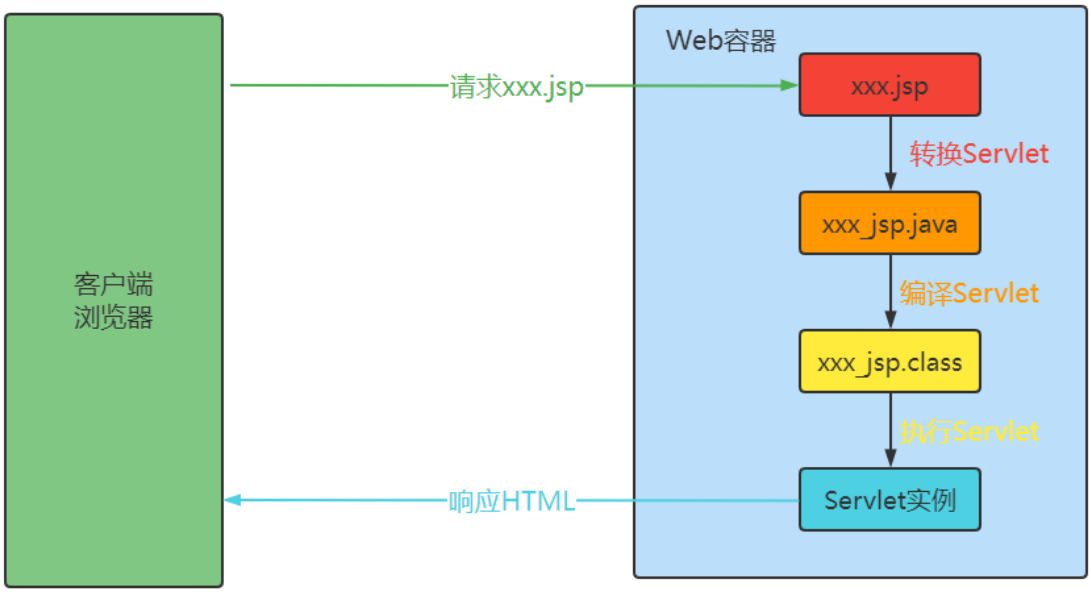
C Thymeleaf

D JAVA

答案

1=>B　　2=>D

JSP运行原理



JSP技术特点

JSP和Servlet是本质相同的技术。当一个JSP文件第一次被请求时，JSP引擎会将该JSP编译成一个Servlet，并执行这个Servlet。如果JSP文件被修改了，那么JSP引擎会重新编译这个JSP。

JSP引擎对JSP编译时会生成两个文件分别是.java的源文件以及编译后的.class文件，并放到Tomcat的work目录的Catalina对应的虚拟主机目录中的org\apache\jsp目录中。两个文件的名称会使用JSP的名称加"_jsp"表示。如：index_jsp.java、index_jsp.class

JSP与Servlet区别

- JSP以源文件形式部署到容器中。而Servlet需要编译成class文件后部署到容器中。
- JSP部署到web项目的根目录下或根目录下的其他子目录和静态同资源位于相同位置。而Servlet需要部署到WEB-INF/classes目录中。
- JSP中的HTML代码会被JSP引擎放入到Servlet的out.write()方法中。而在Servlet中我们需要自己通过对字符流输出流的操作生成响应的页面。
- JSP更擅长表现于页面显示，Servlet更擅长于逻辑控制。

实时效果反馈

1.JSP和是本质相同的技术

A Servlet

B JAVA

C HTML

D C

2.JSP引擎会在什么情况下将JSP编译成一个Servlet

A 每次都会编译

B JSP文件第一次被请求时

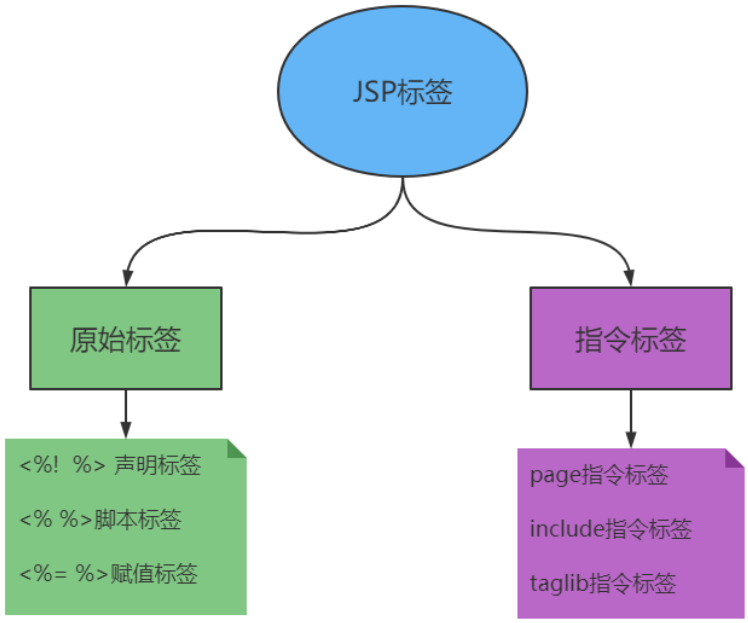
C JSP发生改变后

D JSP文件第一次被请求时或请求的JSP发生改变后

答案

1=>A　　2=>D

JSP标签的使用



JSP的三种原始标签

JSP的原始标签在JSP的任何版本中都可以使用。

<%! %> 声明标签

声明标签用于在JSP中定义成员变量与方法的定义。标签中的内容会出现在JSP被编译后的Servlet的class的{}中。

<% %>脚本标签

脚本标签用于在JSP中编写业务逻辑。标签中的内容会出现在JSP被编译后的Servlet的_jspService方法体中。

<%= %>赋值标签

赋值标签用于在JSP中做内容输出。标签中的内容会出现在_jspService方法的out.print()方法的参数中。注意我们在使用赋值标签时不需要在代码中添加”；”。

JSP原始标签的使用

需求：以20%概率显示你中奖了。

JSP的指令标签

JSP指令标签的作用是声明JSP页面的一些属性和动作。

<%@指令名称 属性="值" 属性="值1,值2...."%>

JSP指令标签分类

page指令标签	位于JSP页面的顶端，可以有多个，主要声明JSP页面的一些属性。
include指令标签	嵌入另一个JSP页面，同时解析这个JSP页面。
taglib指令标签	导入其它标签库。

Page指令标签

contentType

设置响应类型和编码。

pageEncoding

设置页面的编码。

import

导入所需要的包。

language

当前JSP页面里面可以嵌套的语言。

session

设置JSP页面是否获取session内置对象。

buffer

设置JSP页面的流的缓冲区的大小。

autoFlush

是否自动刷新。

extends

声明当前JSP的页面继承于那个类.必须继承的是httpservlet 及其子类。

isELIgnored

是否忽略el表达式。

errorPage

当前JSP页面出现异常的时候要跳转到的JSP页面。

isErrorPage

当前JSP页面是否是一个错误页面。若值为true,可以使用JSP页面的一个内置对象 exception。

Include指令标签

静态包含,可以将其他页面内容包含进来,一起进行编译运行.生成一个java文件.

```
<%@include file="被包含JSP的相对路径" %>
```

Taglib指令标签

导入标签库。

```
<%@taglib prefix="前缀名" uri="名称空间" %>
```

JSP的内置对象

JSP中一共预先定义了9个这样的对象，分别为：request、response、session、application、out、pagecontext、config、page、exception。

request对象

request 对象是 HttpServletRequest类型的对象。

response对象

response 对象是 HttpServletResponse类型的对象。

session对象

session 对象是HttpSession类型的对象。只有在包含 session=“true” 的页面中才可以被使用。

application对象

application 对象是ServletContext类型的对象。

out 对象

out 对象是JspWriter类型的对象。

config 对象

config 对象是ServletConfig类型的对象。

pageContext 对象

pageContext 对象是PageContext类型的对象。作用是取得任何范围的参数，通过它可以获取 JSP页面的out、request、reponse、session、application 等对象。pageContext对象的创建和初始化都是由容器来完成的，在JSP页面中可以直接使用 pageContext对象。

page 对象

page 对象代表JSP本身。

exception 对象

exception 对象的作用是显示异常信息，只有在包含 isErrorPage="true" 的页面中才可以被使用。

请求转发

什么是请求转发

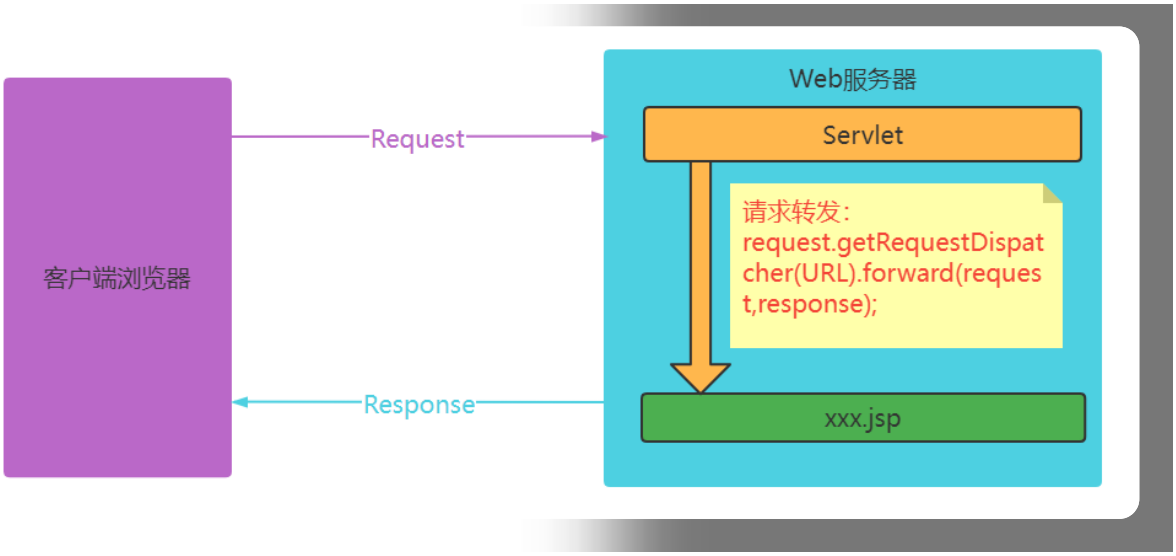
请求转发是服务端的一种请求方式，相当于在服务端中直接请求某个资源。

RequestDispatcher dispatcher = request.getRequestDispatcher("/test.jsp");

dispatcher.forward(request,response);

简写方式：

request.getRequestDispatcher("/test.jsp").forword(request,response);



请求转发与重定向的区别

- 请求转发对于客户端浏览器而言是在一次请求与响应中完成，而重定向是在两次请求两次响应中完成。
- 请求转发并不会改变客户端浏览器的地址栏中的内容。而重定向会改变客户端浏览器地址栏中的内容。
- 请求转发可以使用request对象传递数据，而重定向不能使用request对象传递数据。
- 如果是处理的DML操作，建议使用重定向方式为客户端浏览器产生响应，可以解决表单重复提交现象。

请求转发案例

需求：在Servlet中获取客户端浏览器所支持的语言，并通过JSP页面将客户端浏览器所支持的语言响应给客户端浏览器。

JSP中的四大作用域对象

作用域：“数据共享的范围”，也就是说数据能够在多大的范围内有效。

对象名称	作用范围
application	整个应用都有效
session	在当前会话中有效
request	在当前请求中有效
page	在当前页面有效

JSTL标签库

什么是JSTL标签库

JSTL（Java server pages standard tag library，即JSP标准标签库）JSTL标签是基于JSP页面的。这些标签可以插入在JSP代码中，本质上JSTL也是提前定义好的一组标签，这些标签封装了不同的功能，在页面上调用标签时，就等于调用了封装起来的功能。JSTL的目标是使JSP页面的可读性更强、简化JSP页面的设计、实现了代码复用、提高效率。

在JSP2.0版本后开始支持JSTL标签库。在使用JSTL标签库时需要在JSP中添加对应的taglib指令标签。

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
```

JSTL标签分类

根据JSTL标签所提供的功能，可以将其分为5个类别。

核心标签

最常用、最重要，也是最基本的标签

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
```

标签	描述
<c:out>	用于在JSP中显示数据，就像<%= ... >
<c:set>	用于保存数据
<c:remove>	用于删除数据
<c:catch>	用来处理产生错误的异常状况，并且将错误信息储存起来
<c:if>	与我们在一般程序中用的if一样
<c:choose>	本身只当做<c:when>和<c:otherwise>的父标签
<c:when>	<c:choose>的子标签，用来判断条件是否成立
<c:otherwise>	<c:choose>的子标签，接在<c:when>标签后，当<c:when>标签判断为false时被执行
<c:import>	检索一个绝对或相对 URL，然后将其内容暴露给页面
<c:forEach>	基础迭代标签，接受多种集合类型
<c:forTokens>	根据指定的分隔符来分隔内容并迭代输出
<c:param>	用来给包含或重定向的页面传递参数
<c:redirect>	重定向至一个新的URL.
<c:url>	使用可选的查询参数来创造一个URL

格式化标签

JSTL格式化标签用来格式化并输出文本、日期、时间、数字。

```
<%@ taglib prefix="fmt"
uri="http://java.sun.com/jsp/jstl/fmt" %>
```

标签	描述
<fmt:formatNumber>	使用指定的格式或精度格式化数字
<fmt:parseNumber>	解析一个代表着数字，货币或百分比的字符串
<fmt:formatDate>	使用指定的风格或模式格式化日期和时间
<fmt:parseDate>	解析一个代表着日期或时间的字符串
<fmt:bundle>	绑定资源
<fmt:setLocale>	指定地区
<fmt:setBundle>	绑定资源
<fmt:timeZone>	指定时区
<fmt:setTimeZone>	指定时区
<fmt:message>	显示资源配置文件信息
<fmt:requestEncoding>	设置request的字符编码

SQL标签

JSTL SQL标签库提供了与关系型数据库（Oracle，MySQL，SQL Server等等）进行交互的标签。

```
<%@ taglib prefix="sql"
uri="http://java.sun.com/jsp/jstl/sql" %>
```

标签	描述
<sql:setDataSource>	指定数据源
<sql:query>	运行SQL查询语句
<sql:update>	运行SQL更新语句
<sql:param>	将SQL语句中的参数设为指定值
<sql:dateParam>	将SQL语句中的日期参数设为指定的java.util.Date 对象值
<sql:transaction>	在共享数据库连接中提供嵌套的数据库行为元素，将所有语句以一个事务的形式来运行

XML标签

JSTL XML标签库提供了创建和操作XML文档的标签。

```
<%@ taglib prefix="x"
uri="http://java.sun.com/jsp/jstl/xml" %>
```

标签	描述
<x:out>	与<%= ... >,类似，不过只用于XPath表达式
<x:parse>	解析 XML 数据
<x:set>	设置XPath表达式
<x:if>	判断XPath表达式，若为真，则执行本体中的内容，否则跳过本体
<x:forEach>	迭代XML文档中的节点
<x:choose>	<x:when>和<x:otherwise>的父标签
<x:when>	<x:choose>的子标签，用来进行条件判断
<x:otherwise>	<x:choose>的子标签，当<x:when>判断为false时被执行
<x:transform>	将XSL转换应用在XML文档中
<x:param>	与<x:transform>共同使用，用于设置XSL样式表

JSTL函数

JSTL包含一系列标准函数，大部分是通用的字符串处理函数。

```
<%@ taglib prefix="fn"
uri="http://java.sun.com/jsp/jstl/functions"
%>
```

函数	描述
fn:contains()	测试输入的字符串是否包含指定的子串
fn:containsIgnoreCase()	测试输入的字符串是否包含指定的子串，大小写不敏感
fn:endsWith()	测试输入的字符串是否以指定的后缀结尾
fn:escapeXml()	跳过可以作为XML标记的字符
fn:indexOf()	返回指定字符串在输入字符串中出现的位置
fn:join()	将数组中的元素合成一个字符串然后输出
fn:length()	返回字符串长度
fn:replace()	将输入字符串中指定的位置替换为指定的字符串然后返回
fn:split()	将字符串用指定的分隔符分隔然后组成一个子字符串数组并返回
fn:startsWith()	测试输入字符串是否以指定的前缀开始
fn:substring()	返回字符串的子集
fn:substringAfter()	返回字符串在指定子串之后的子集
fn:substringBefore()	返回字符串在指定子串之前的子集
fn:toLowerCase()	将字符串中的字符转为小写
fn:toUpperCase()	将字符串中的字符转为大写
fn:trim()	移除首尾的空白符

EL表达式

Expression Language

什么是EL表达式

EL（Expression Language）是一种表达式语言。是为了使JSP写起来更加简单，减少java代码，可以使得获取存储在Java对象中的数据变得非常简单。在JSP2.0版本后开始支持EL表达式。

语法结构

${表达式}

${对象.属性名}

EL表达式中的操作符

操作符	描述
( )	优先级
+	加
-	减或负
*	乘
/ 或div	除
% 或 mod	取模
== 或 eq	测试是否相等
!= 或 ne	测试是否不等
< 或 lt	测试是否小于
> 或gt	测试是否大于
<= or le	测试是否小于等于
>= 或 ge	测试是否大于等于
&& 或 and	测试逻辑与
|| 或 or	测试逻辑或
! 或not	测试取反
empty	测试是否空值

EL表达式的隐含对象

隐含对象	描述
pageScope	page 作用域
requestScope	request 作用域
sessionScope	session 作用域
applicationScope	application 作用域
param	Request 对象的参数，字符串
paramValues	Request对象的参数，字符串集合
header	HTTP 信息头，字符串
headerValues	HTTP 信息头，字符串集合
initParam	上下文初始化参数
cookie	Cookie值
pageContext	当前页面的pageContext

使用EL表达式取出作用域中的值

${pageScope.name}

${requestScope.name}

${sessionScope.name}

${applicationScope.name}

获取作用域属性中的数据时，也可以只写属性名，EL表达式会按照pageScope、requestScope、sessionScope、applicationScope的顺序查找该属性的值。

${name}

JSTL标签库与EL表达式的使用

JSTL标签库的使用步骤

添加jstl.jar

在JSP页面中添加taglib指令标签。

JSTL核心标签的使用

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" %>
```

< c:if >

标签判断表达式的值，如果表达式的值为 true 则执行其主体内容。

< c:choose >, < c:when >, < c:otherwise >

< c:choose >标签与Java switch语句的功能一样，用于在众多选项中做出选择。

switch语句中有case，而< c:choose >标签中对应有< c:when >，switch语句中有default，而< c:choose >标签中有< c:otherwise >。

< c:forEach >

迭代器，用于迭代集合。

属性	描述
items	被迭代的集合
begin	迭代器的起始因子
end	迭代器的结束因子
step	迭代因子的增长数
var	代表当前迭代元素的变量名称
varStatus	代表循环状态的变量名称

varStatus 属性

current: 当前这次迭代的（集合中的）项

index: 当前这次迭代从 0 开始的迭代索引

count: 当前这次迭代从 1 开始的迭代计数

first: 用来表明当前这轮迭代是否为第一次迭代的标志

last: 用来表明当前这轮迭代是否为最后一次迭代的标志

begin: 属性值

end: 属性值

step: 属性值

使用ForEach迭代List

需求：

创建Users对象，含有userid，username属性。

创建一个Servlet，在Servlet中创建多个Users对象并放到List集合中，在showUsers.jsp的页面中显示所有的Users对象的信息。

使用ForEach迭代Map

需求：

创建Users对象，含有userid，username属性。

创建一个Servlet，在Servlet中创建多个Users对象并放到Map集合中，在showUsers2.jsp的页面中显示所有的Users对象的信息。

JSTL格式化标签的使用

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>

对日期的格式化处理

```
<fmt:formatDate value="${data}"
pattern="yyyy-MM-dd"/>
```

对数字格的式化处理

```
<fmt:formatNumber value="${balance}"
type="currency"/>
```

MVC模式

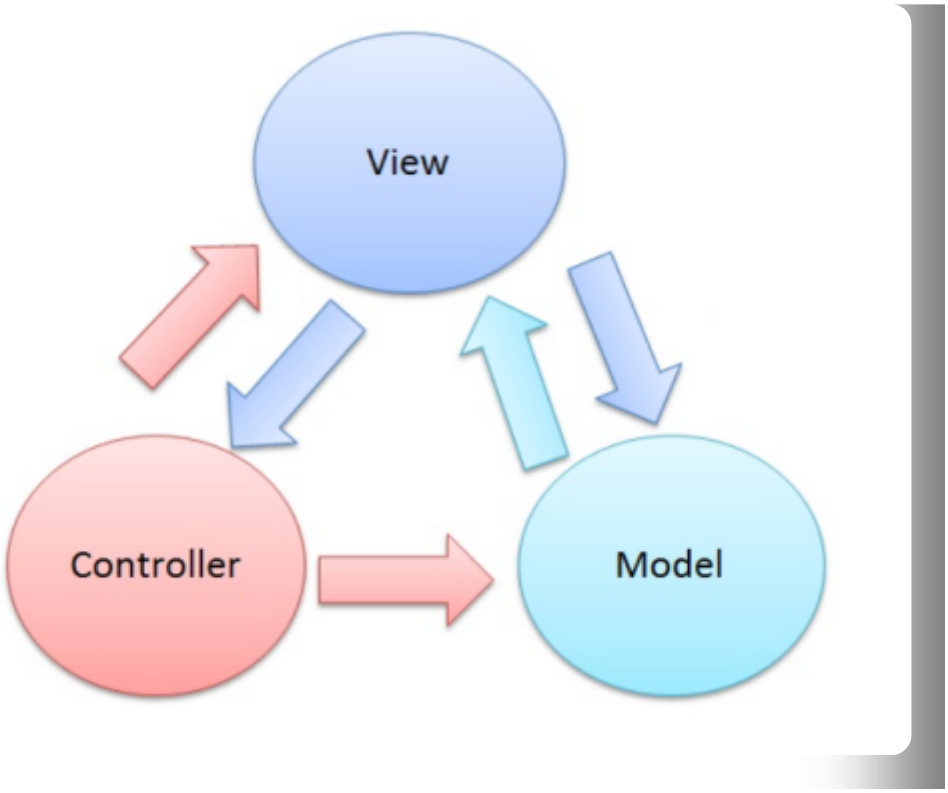
什么是MVC模式

MVC模式：Model、View、Controller即模型、视图、控制器。是软件的一种架构模式（Architecture pattern）。MVC要实现的目标是将软件的用户界面和业务逻辑分离，可提高代码可扩展性、可复用性、可维护性、以及灵活性。

View(视图)：用户的操作界面。如：html、jsp。

Model(模型)：具体的业务模型与数据模型。如：service、dao、pojo。

Controller(控制)：处理从视图层发送的请求，并选取模型层的业务模型完成响应的业务实现，并产生响应。如：Servlet。



MVC模式与应用程序分层的区别

MVC模式是一种软件的架构方式，而应用程序分层这是一种代码的组织方式。MVC模式与应用程序分层的目标都是一致的：为了解耦和、提高代码复用性。