```
        <plugins>
            <!-- Compile -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <source>1.6</source>
                    <target>1.6</target>
                </configuration>
            </plugin>
            <!-- Test -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.18.1</version>
                <configuration>
                    <skipTests>true</skipTests>
                </configuration>
            </plugin>
            <!-- Tomcat -->
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <version>2.2</version>
                <configuration>
                    <path>/${project.artifactId}</path>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

现在一个基于 Maven 的 Java Web 项目已搭建完毕，随后可进入具体的开发阶段。

学习任何技术，都需要从最容易入手的地方开始，下面我们利用一个简单的应用场景，一起学习 Servlet 3.0 框架的使用方法。

1.3 编写一个简单的 Web 应用

1.3.1 编写 Servlet 类

我们要做的是一件非常简单的事情：写一个 HelloServlet，接收 GET 类型的/hello 请求，转发到/WEB-INF/jsp/hello.jsp 页面，在 hello.jsp 页面上显示系统当前时间。

首先，需要在 java 目录下，创建一个名为 org.smart4j.chapter1 的包。

技巧：用鼠标选中 java 目录，按下 Alt+Insert 快捷键，选择 Package 选项，输入具体的包名，回车后即可创建包，用同样的方法也可以创建类或其他文件。

然后，在该包下创建一个 HelloServlet 的类，代码如下：

```
package org.smart4j.chapter1;

public class HelloServlet {
}
```

最后，我们需要为该类添加一些代码：

```
package org.smart4j.chapter1;

import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/hello")
public class HelloServlet extends HttpServlet {

    @Override
```

```
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String currentTime = dateFormat.format(new Date());
        req.setAttribute("currentTime", currentTime);
        req.getRequestDispatcher("/WEB-INF/jsp/hello.jsp").forward(req,
        resp);
    }
}
```

对以上代码进行一些说明：

（1）继承 HttpServlet，让它成为一个 Servlet 类。

（2）覆盖父类的 doGet 方法，用于接收 GET 请求。

（3）在 doGet 方法中获取系统当前时间，并将其放入 HttpServletRequest 对象中，最后转发到/WEB-INF/jsp/hello.jsp 页面。

（4）使用 WebServlet 注解并配置请求路径，对外发布 Servlet 服务。

不需要在 web.xml 中添加任何的 servlet 配置，因为我们使用了 Servlet 3.0 规范提供的 WebServlet 注解。除此以外，还有很多的注解或 API，会让我们拥有一个“零配置”的 web.xml 。

技巧：

（1）在 HelloServlet 类中使用 Alt+Insert 快捷键，选择“Qverrides Methods...”选项，选择 javax.servlet.http.HttpServlet 中的 doGet 方法（可输入首字母进行快速查找），最后单击“OK”按钮或回车，可快速添加 doGet 方法模版。

（2）将光标定位在某个位导入的类上（HttpServlet），使用 Alt + Insert 快捷键，可快速导入包名（javax.servlet.annotation.WebServlet）。

（3）怎样在 IDEA 中使用“代码提示”快捷键？我们需要在 Settings→Appearance & Behavior → Keymap 里设置，路径为 Main menu / Code / Completion / Basic，默认为 Ctrl + 空格键，它与切换中英文输入法是有冲突的，建议将其修改为自己最习惯的快捷键，例如，Alt + /（与 Eclipse 相同）。

1.3.2　编写 JSP 页面

HelloServlet 已经完成了，我们接着写一个 hello.jsp。在 webapp 目录下创建 jsp 目录，并在该目录下创建 hello.jsp，编写如下代码：

```
<%@ page pageEncoding="UTF-8" %>
<html>
<head>
    <title>Hello</title>
</head>
<body>

<h1>Hello!</h1>

<h2>当前时间：${currentTime}</h2>

</body>
</html>
```

可见，我们使用了 JSTL 表达式来获取从 HelloServlet 里传递过来的 currentTime 请求属性。

至此，所有的代码已编写完毕，我们即将让这个 Web 应用跑起来！

1.4 让 Web 应用跑起来

1.4.1 在 IDEA 中配置 Tomcat

首先，我们需要在 IDEA 里配置一个 Tomcat，详细过程如下：

（1）单击 IDEA 的工具栏上的“Edit Configurations...”（在一个下拉框里，一眼就能看到它），弹出 Run/Debug Configurations 对话框。

（2）单击左上角的“+”按钮（或使用 Alt+Insert 快捷键），选择 Tomcat Server→Local 选项。

（3）输入 Tomcat 的 Name（例如：tomcat），取消勾选“After launch”选项。

（4）单击 Application server 下拉框右侧的“Configure...”按钮，配置一个 Tomcat，配置完毕后，在下拉框中选择该 Tomcat。

（5）切换到 Deployment 选项卡，单击右边的“+”按钮（或使用 Alt+Insert 快捷键），选择“Artifact...”选项，弹出 Select Artifact to Deploy 对话框。

（6）选择 chapter1:war exploded，单击“OK”按钮或回车，关闭 Select Artifact to Deploy 对话框。

（7）回到 Run/Debug Configurations 对话框，在 Application context 中输入/chapter1。

（8）切换回 Server 选项卡，在 On frame deactivation 下拉框中选择“Update resources”选项，单击“OK”按钮，完成所有配置，关闭 Run/Debug Configurations 对话框。

然后，单击 IDEA 工具栏上的“Run”或“Debug”按钮，启动 Tomcat 并部署 Web 应用。

技巧：在开发阶段建议使用 Debug 方式运行 Tomcat，这样在代码中所做的修改可进行自动部署（热部署），只需使用 Ctrl + F9 键手工编译即可。不过有些情况下是无法进行热部署的，例如，修改了类名、方法名、成员变量名等。

1.4.2　使用 Tomcat 的 Maven 插件

如果我们当前使用的是 IDEA 的社区版，该版本并没有提供集成 Tomcat 的功能，那么我们怎么在 IDEA 中启动 Tomcat 并实现 Debug 功能呢？

我们可以使用 Tomcat 的 Maven 插件，并在 IDEA 中以插件的方式启动 Tomcat。只需在 pom.xml 中添加如下配置：

```
<!-- Tomcat -->
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/${project.artifactId}</path>
    </configuration>
</plugin>
```

打开 IDEA 的 Maven 面板，双击 tomcat7:run 命令，即可运行 Maven 命令（mvn tomcat7:run），如图 1-4 所示。

此时，若我们尝试在 hello.jsp 中修改部分代码，刷新浏览器，将看到相应的调整，但修改了 HelloServlet 并编译后，并不能实现热部署，此外，也不能设置断点进行调试。

此方法仅用于运行 Maven 命令，并不能与 IDEA 整合，难道就没有更好的解决方案了吗？

一定要相信强大的 IDEA，它是绝对不会让我们失望的！

1.4.3　以 Debug 方式运行程序

我们在 IDEA 社区版中添加一个 Maven 的 Configuration（具体操作与添加 Tomcat 类似），

在 Name 中输入 tomcat，在 Command line 中输入 tomcat7:run，回车后关闭对话框后，我们就可以单击“Run”或“Debug”按钮来启动 Tomcat 并部署应用了。

使用这种方法，不仅修改的 JSP 可立即生效，也可以实现类的热部署，而且还可以使用断点调试。

最后，打开我们最喜欢的浏览器，输入以下地址：http://localhost:8080/chapter1/hello。

此时，我们就会看到想要的效果，如图 1-5 所示。

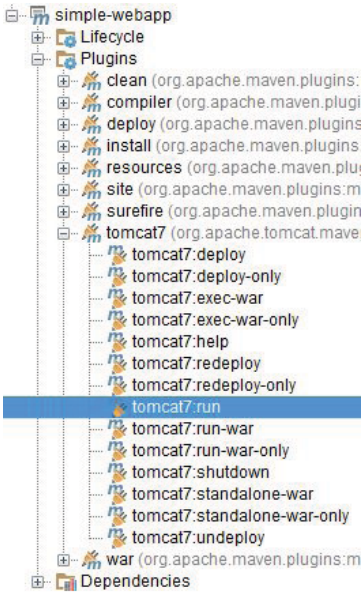

图 1-4　在 IDEA 中执行 Maven 命令

图 1-5　Hello 应用运行效果

1.5　将代码放入 Git 仓库中

1.5.1　编写.gitignore 文件

现在项目框架已经搭建完毕，我们有必要将代码放入 Git 仓库中进行版本控制管理。

有些文件是不需要放入 Git 中的，比如 Maven 生成的 target 目录、IDEA/Eclipse 的工程文件。

在项目的根目录（C:）下添加一个名为.gitignore 的文件，内容如下：

```
# Maven #
target/
```

```
# IDEA #
.idea/
*.iml

# Eclipse #
.settings/
.metadata/
.classpath
.project
Servers/
```

1.5.2　提交本地 Git 仓库

在 IDEA 中找到 VCS 菜单，单击“Import into Version Control/Create Git Repository...”菜单项，在弹出的对话框中选中项目的根目录，单击“OK”按钮关闭对话框。此时，IDEA 就为我们在本地创建了一个 Git 仓库。

选中项目根目录，在 VCS 菜单（或右键菜单）中单击“Git/Add”菜单项，可将除.gitignore 中忽略的所有文件添加到本地 Git 仓库中。当然，也可以使用 Ctrl+Alt+A 键来完成，只需选择需要提交的文件，然后使用该快捷键即可。若此时开启了 QQ，则需修改它的屏幕截图快捷键（建议修改为 Ctrl+Alt+Z），否则将引起快捷键冲突，导致 IDEA 无法使用该快捷键。

在 VCS 菜单（或右键菜单）中单击 Git/Commit Directory...菜单项，随后将弹出一个对话框，确认是否进行提交操作，输入 Commit Message 后，单击“Commit”按钮就可以提交代码到本地 Git 仓库了。当然，也可以单击工具栏中的“Commit Changes”按钮，或使用 Ctrl+K 键来做同样的事情。

提示：在提交代码时，建议勾选“Optimize imports”选项，它可优化 import 语句，去掉没有使用的包。

1.5.3　推送远程 Git 仓库

可以随时将 Git 本地仓库推送到远程仓库，只需在 IDEA 中的 VCS 菜单中单击 Git/Push 菜单项即可，或者使用 Ctrl+Shift+K 键。需要注意的是，在推送之前，必须将本地仓库与远程仓库建立一个连接。

我们可以免费使用 OSC（开源中国）提供的 Git 远程仓库，首先需要在 Git@OSC（http://git.oschina.net/）中创建一个项目，执行如下命令来完成推送：

```
git remote add origin <您的 Git 仓库地址>
git push -u origin master
```

现在就可以在 Git@OSC 中看到已推送的代码了。

关于“开源中国”开源中国成立于 2008 年 8 月，是目前国内最大的开源技术社区，拥有超过 200 万会员，有开源软件库、代码分享、资讯、协作翻译、讨论区和博客等几大频道内容，为 IT 开发者提供了一个发现、使用并交流开源技术的平台。2013 年，开源中国建立大型综合性的云开发平台——中国源，为中国广大开发者提供团队协作、源码托管、代码质量分析、代码评审、测试、代码演示平台等功能。

1.5.4 总结

在本章中，我们使用 IDEA 开发了一个简单的 Web 应用，学会了：

（1）如何在 IDEA 中创建 Maven 项目。

（2）如何将普通的 Maven 项目转为 Java Web 项目。

（3）如何在 IDEA 中集成 Tomcat（提供了三种方法）。

（4）如何将代码提交到本地 Git 仓库并推送到远程 Git 仓库。

我们现在已经熟悉了 Java Web 开发的常用工具与技巧，现在仅仅是一个 Servlet+JSP 而已，并没有与数据库打交道，我们在下一章里会编写更多的 Servlet 与 JSP，此外还会使用 JDBC 与数据库进行交互，一个真正的 Web 应用即将诞生。

第 2 章

为 Web 应用添加业务功能

经过我们上一章的努力，一个简单的 Web 应用基本完成了，但目前只能通过 Servlet 处理简单的请求，并没有实现具体的业务逻辑。

现在我们将在这个应用的基础上增加一些业务功能，您将学会更多有关项目实战的技能，具体包括：

- 如何进行需求分析；
- 如何进行系统设计；
- 如何编写应用程序。

此外，我们将使用一些代码重构的技巧，不断优化现有代码。好的程序不是一次性写出来的，而是不断地“改”出来的，这里的“改”就是重构。

我们现在就从需求分析开始。