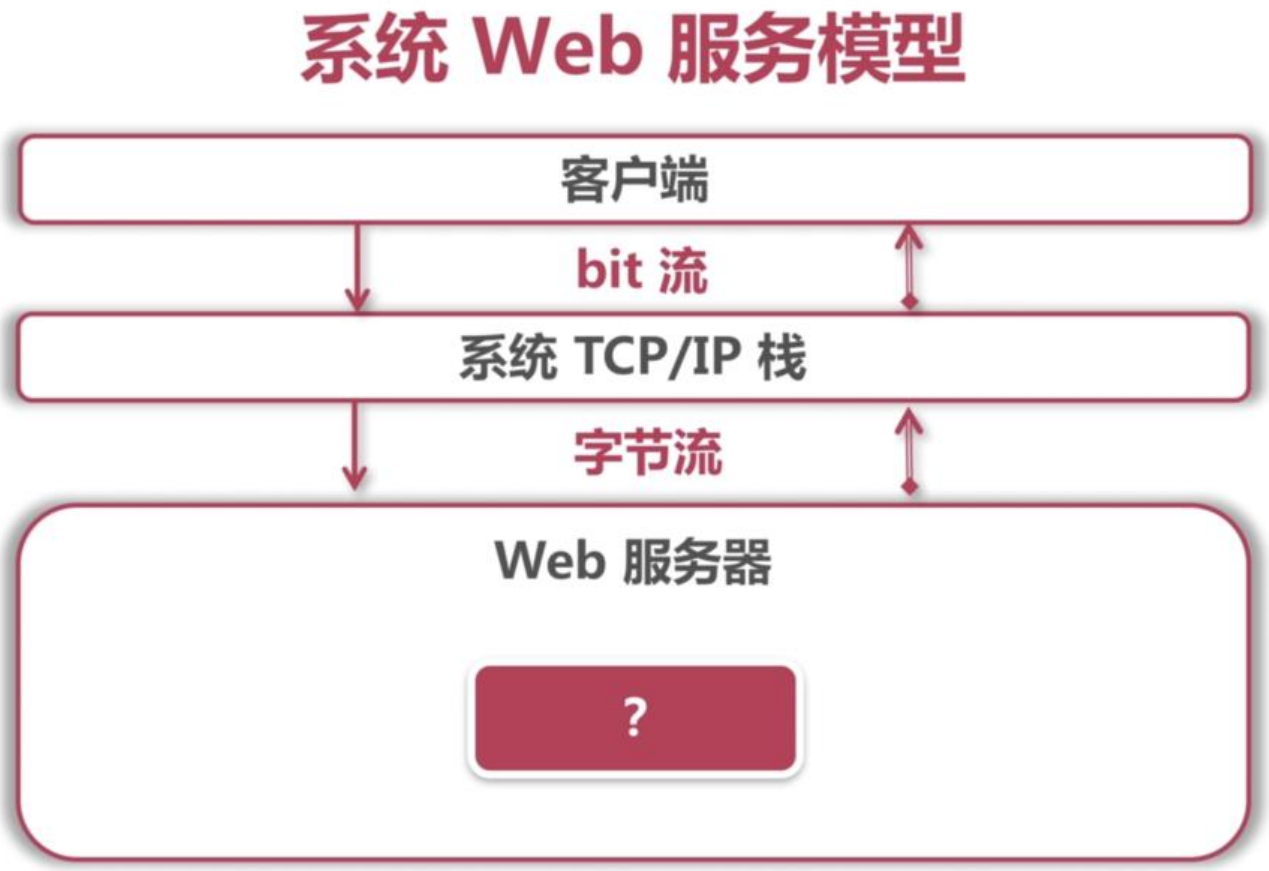
链滴

mini-spring 第二期 --web 服务与 servlet，集成 tomcat 服务器

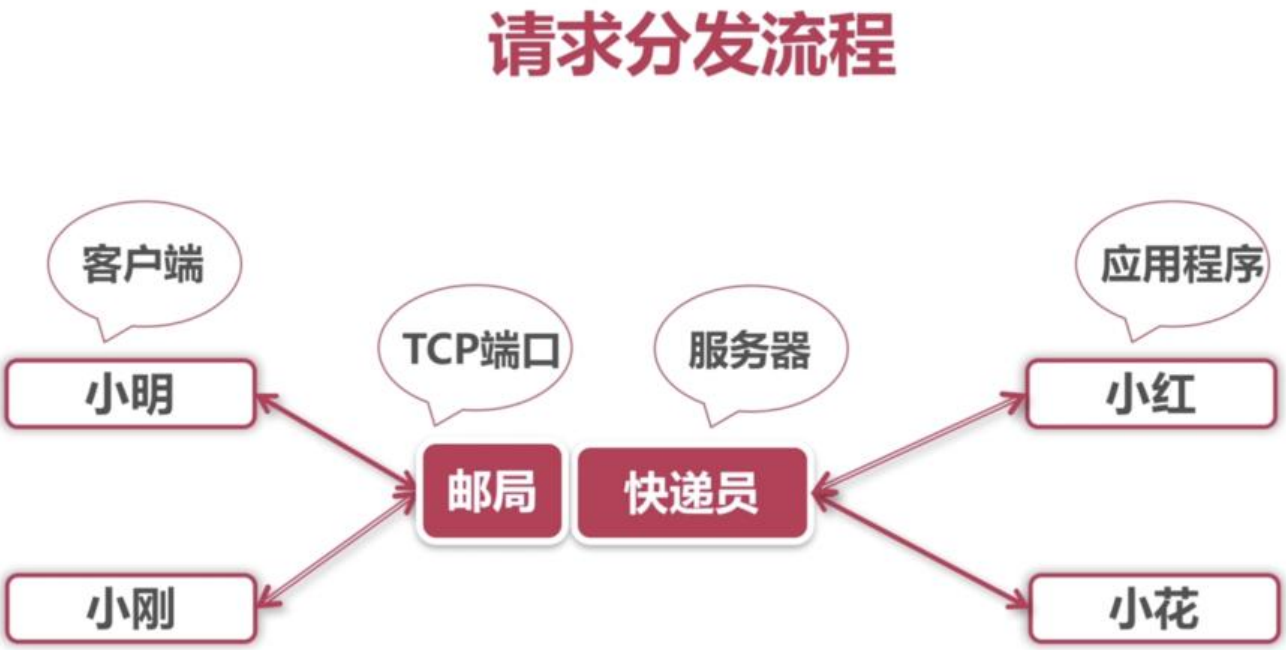
作者：ChenforCode
原文链接：https://ld246.com/article/1572880567253
来源网站：链滴

1.web服务器只只专注于接受请求和返回相应，并不涉及具体的业务逻辑。



2.请求转发模型

tcp端口不管有没有请求，都会一直等着，并一直监听着，等待着请求的到来



3.sevlet

sevlet是一种规范，约束了Java服务器和业务类的通信方式

是一种接口，javax.servlet.Servlet

是一个实现类，实现了sevlet接口的实现类

web服务器通过sevletRequest和sevletResponse类和sevlet进行交互

4.在framework模块中加入tomcat embed的依赖

// https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-core

compile group: 'org.apache.tomcat.embed', name: 'tomcat-embed-core', version: '8.5.23'

5.web包中开始写sever

```
package cn.chenforcode.web.server;

import org.apache.catalina.LifecycleException;
import org.apache.catalina.startup.Tomcat;

public class TomcatServer {
    private Tomcat tomcat;
    private String[] args;

    TomcatServer(String[] args) {
        this.args = args;
    }

    public void startServer() throws LifecycleException {
        tomcat = new Tomcat();
        tomcat.setPort(6699);
        tomcat.start();

        Thread awaitThread = new Thread("tomcat_await_thread") {
            @Override
            public void run() {
                TomcatServer.this.tomcat.getServer().await();
            }
        };
        awaitThread.setDaemon(false);
        awaitThread.start();
    }
}
```

其中startServer方法是启动一个tomcat服务器

6.在miniapplication中启动一个服务器

```
public class MiniApplication {
    public static void run(Class<?> cls, String[] args) {
        System.out.println("Hello Mini-spring!");
        TomcatServer tomcatServer = new TomcatServer(args);
        try {
            tomcatServer.startServer();
        } catch (LifecycleException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

7.建立一个servlet，并实现其servlet方法

```
    @Override
    public void service(ServletRequest req, ServletResponse res) throws ServletException, IOExc
ption {
        res.getWriter().println("test");
    }
```

8.在服务器中注册servlet

```
public void startServer() throws LifecycleException {
        tomcat = new Tomcat();
        tomcat.setPort(6699);
        tomcat.start();

        //建立一个容器
        Context context = new StandardContext();
        context.setPath("");
        //增加生命周期监听器
        context.addLifecycleListener(new Tomcat.FixContextListener());

        //创建servlet
        TestServlet testServlet = new TestServlet();
        //把servlet加入到tomcat里，并支持异步
        Tomcat.addServlet(context, "testServlet",testServlet).setAsyncSupported(true);

        //增加servlet的映射
        context.addServletMappingDecoded("/test.json", "testServlet");

        //context容器必须依附在一个host容器中
        tomcat.getHost().addChild(context);


        Thread awaitThread = new Thread("tomcat_await_thread") {
            @Override
            public void run() {
                TomcatServer.this.tomcat.getServer().await();
            }
        };
        awaitThread.setDaemon(false);
        awaitThread.start();
    }
```

9.重新打包运行，在浏览器中访问/test.json就能够得到相应的结果。