6、该接口中的常用方法

--对请求中提交的表单数据操作的方法如下，（要注意下面几个方法是继承于ServletRequest接口中的方法）：

表单中提交的数据被封装到request对象中，该对象中的map集合存储这些数据：Map集合中的key是name，vluae是一个String类型是一维数组

--String getParameter(String name)

该方法通过key获取value一维数组中首元素（通常情况下，这个一维数组只存一个元素，所以该方法使用次数最多）

--String[] getParameterValues(String name)

该方法是通过Map集合中的key获取value（这个就可以获取到value一维数组中的其他元素，使用次数比上面少）

--Map getParameterMap()

该方法获取存储表单提交数据的那个Map集合对象

--Enumeration getParameterNames()

该方法获取Map集合对象中的key，存放到一个Enumeration类型的集合中（该集合前面见到过）

--比较简单的常用方法如下：

-String getContextPath() //获取上下文的路径（就是webapp的根路径，/部署在服务器上的项目名）

String getMethod() //获取请求方式

String getRequestURI() //获取URI（/服务器上项目名/资源路径,/资源路径就是web.xml文件中url-pattern标签

中的路径）

StringBuffer getRequestURL() //获取URL请求路径

String getServletPath() //获取Servlet PATH(就是资源路径，url-pattern标签中的路径)

String getRemoteAddr() //获取发送请求的客户端的IP地址

--与ServletContext接口中相似的方法

Object getAttribute(String name) //向request对象中读取数据

void setAttribute(String name, Object o) //向request对象中添加数据

void removeAttribute(String name) //向request对象中移除数据

--HttpServletRequest范围：首先该接口类型的变量命名为 request，代表当前的本次请求。一次请求对应一个request对象，100个对应100个。请求范围极小，request只能在同一次请求中传递数据。

ServletContext接口中的方法作用

Object getAttribute(String name) --//从ServletContext范围中读取数据

void setAttribute(String name, Object o) --//从ServletContext范围中添加数据

void removeAttribute(String name) --//从ServletContext范围中移除数据

--ServletContext范围是一个所有用户共享的范围对象，一般把ServletContext类型的变量命名为application。可知该对象代表一个应用。一个webapp只有一个该对象，范围极大。

关于范围的选择：

1、ServletContext 应用范围，可以跨用户传输

2、HttpServletRequest 请求范围，只能在一个请求中传递数据【可以跨Servlet对象传递数据，但这多个Servlet的必须在同一个请求当中。

优先选择request范围

--获取转发器对象的方法

RequestDispatcher getRequestDispatcher(String path)

--使用方法：获取转发器后使用转发器的forward方法将request和response对象传进去就可以转发了

--解决乱码的方法

void setCharacterEncoding(String env)

```
--Cookie[] getCookies()  学习Cookies
-- HttpSession  getSession()    学习Session
--HttpSession   getSession(boolean create)
```

创建web项目idea-servlet-http-9，测试上面两类方法（简单常用以及与ServletContext接口相同的）

先测试简单的几个方法：

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>测试HttpServletRequest</title>
</head>
<body>
    <a href="/idea-servlet-http-9/method/test">测试request</a>
</body>
</html>
```

web.xml

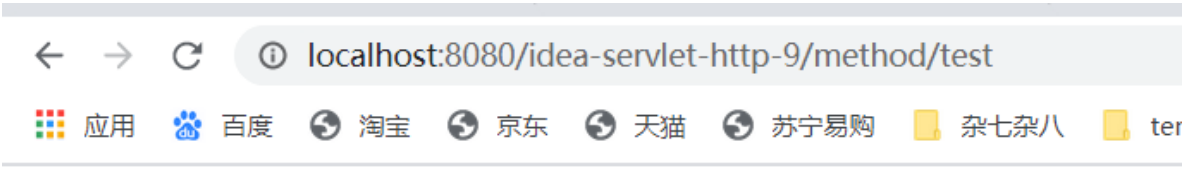
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <servlet>
        <servlet-name>request</servlet-name>
        <servlet-class>com.servlet.httpservletrequest.RequestServlet</servlet-
class>
    </servlet>
    <servlet-mapping>
        <servlet-name>request</servlet-name>
        <url-pattern>/method/test</url-pattern>
    </servlet-mapping>
</web-app>
```

RequestServlet.java

```
package com.servlet.httpservletrequest;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
```

```
public class RequestServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        //获取上下文路径（webapp根路径）
        String webPath = request.getContextPath();
        out.print(webPath);
        out.print("<br>");

        //获取浏览器的请求方式
        String method = request.getMethod();
        out.print(method);
        out.print("<br>");

        //获取URI
        String uri = request.getRequestURI();
        out.print(uri);
        out.print("<br>");

        //获取URL
        StringBuffer url = request.getRequestURL();
        out.print(url);
        out.print("<br>");

        //获取Servlet Path
        String servletPath = request.getServletPath();
        out.print(servletPath);
        out.print("<br>");

        //获取客户端IP地址
        String ip = request.getRemoteAddr();
        out.print(ip);
        out.print("<br>");
    }
}
```

测试结果如下：、



/idea-servlet-http-9
GET
/idea-servlet-http-9/method/test
http://localhost:8080/idea-servlet-http-9/method/test
/method/test
0:0:0:0:0:0:0:1

下面测试后面的三个方法和一个获取请求转发器的方法

首先我们创建了两个类继承HttpServlet，分别是AServlet和BServlet；再创建一个实体类User，其中有两个属性userCode和userName。我们在AServlet类中创建一个User对象并给两个属性赋值，将该对象用

void setAttribute(String name, Object o)方法添加进request对象范围，然后在该类中用getAttribute(String name)方法获取数据，部署项目启动服务器，对AServlet对象发送一个请求，可以成功获取request对象中存进去的数据；然后在BServlet类中写一个获取数据的方法，然后部署项目启动服务器，先发送一个对AServlet对象的请求，使其运行对request对象添加数据的方法，然后发送一个对BServlet对象的请求，结果是没有从该请求的request范围对象中获取数据。因为这里是两个请求，所以是两个request对象，肯定是获取不到另一个request对象中的数据的。

那么我们将对这两个对象的请求合并成一个请求，是可以获取到的，需要用到转发技术。实现方法如下

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>测试HttpServletRequest</title>
</head>
<body>
    <a href="/idea-servlet-http-9/method/test">测试request</a><br>
    <a href="/idea-servlet-http-9/method/a">测试request</a><br>
    <a href="/idea-servlet-http-9/method/b">测试request</a>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <servlet>
        <servlet-name>request</servlet-name>
        <servlet-class>com.servlet.httpservletrequest.RequestServlet</servlet-
class>
    </servlet>
    <servlet-mapping>
        <servlet-name>request</servlet-name>
        <url-pattern>/method/test</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>AServlet</servlet-name>
        <servlet-class>com.servlet.httpservletrequest.AServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>AServlet</servlet-name>
        <url-pattern>/method/a</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>BServlet</servlet-name>
        <servlet-class>com.servlet.httpservletrequest.BServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>BServlet</servlet-name>
        <url-pattern>/method/b</url-pattern>
    </servlet-mapping>
</web-app>
```

AServlet.java

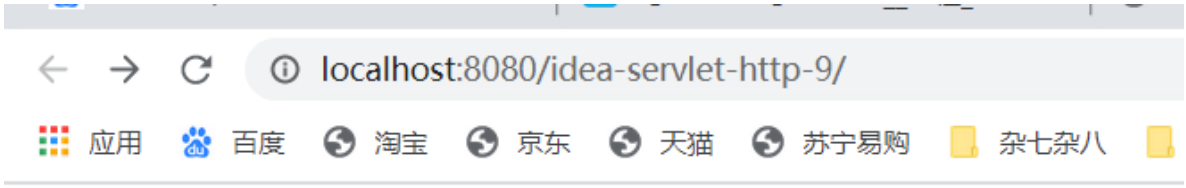
```
package com.servlet.httpservletrequest;

import com.servlet.entity.User;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class AServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        //创建一个User对象
        User user = new User();
        user.setUserCode("16020521234");
        user.setUserName("nick");
```

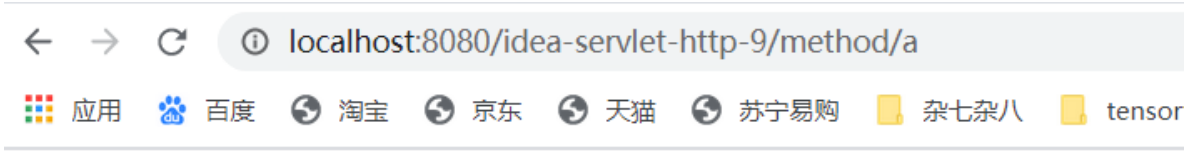
```
        //将User对象存放进request范围对象中
        request.setAttribute("user",user);

        //取出request范围中的数据
        //Object obj = request.getAttribute("user");
        //response.getWriter().print(obj);

        //希望在另一个Servlet对象中取出本request范围中的数据
        //将另一个对象的请求容纳进本对象的请求中，就可以共有本对象的request范围了
        //跳转
        //执行完AServlet后，跳转到BServlet继续执行，这样两个类对象就放到同一个请求中了
        //使用转发技术，forward（转发）方法

        //1、获取获取转发器对象(以下转发器指向了BServlet，路径中是url-pattern标签中的内容)
        RequestDispatcher dispatcher = request.getRequestDispatcher("/method/b"
);
        //2、调用转发器的forward方法将request和response传过去就可完成转发，共有一个
request和response对象
        dispatcher.forward(request,response);

    }
}
```

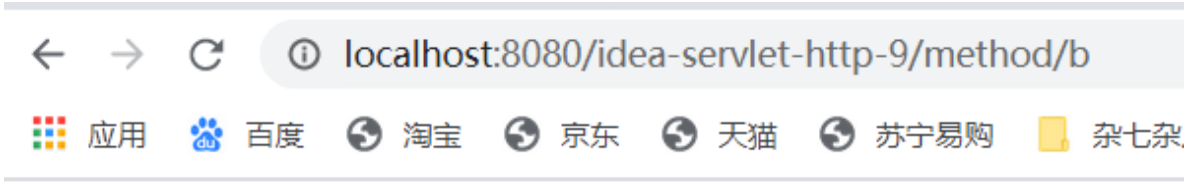
BServlet.java

```
package com.servlet.httpservletrequest;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class BServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        //取出request范围中的数据
        Object obj = request.getAttribute("user");
        response.getWriter().print(obj);

    }
}
```

测试结果如下：、







通过操作，当我们点击AServlet的时候，即使获取本请求request对象数据的方法在另一个Servlet对象中，仍然将数据获取到了；我们返回，点击BServlet时，是无法获取数据的。