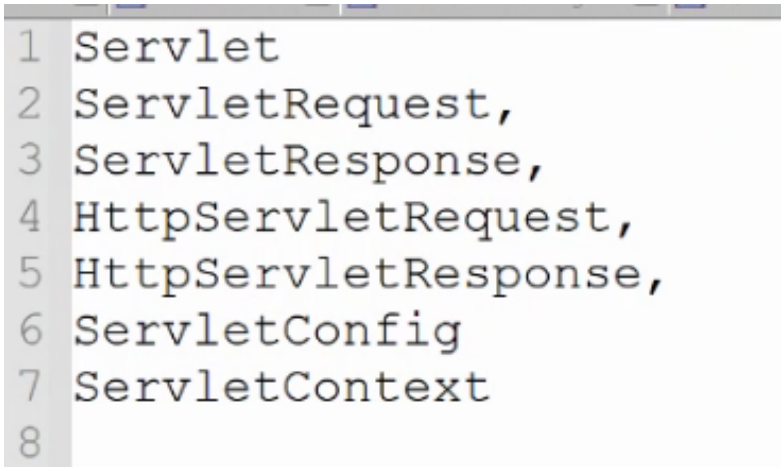
2021-2022赛季 软191级队学习资料

J2EE编程之重点API（基础包）



Servlet

英文简介

```
public interface Servlet
```

Defines methods that all servlets must implement.

A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.

To implement this interface, you can write a generic servlet that extends `javax.servlet.GenericServlet` or an HTTP servlet that extends `javax.servlet.http.HttpServlet`.

This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods and are called in the following sequence:

1. The servlet is constructed, then initialized with the `init` method.
2. Any calls from clients to the `service` method are handled.
3. The servlet is taken out of service, then destroyed with the `destroy` method, then garbage collected and finalized.

In addition to the life-cycle methods, this interface provides the `getServletConfig` method, which the servlet can use to get any startup information, and the `getServletInfo` method, which allows the servlet to return basic information about itself, such as author, version, and copyright.

中文简介

定义所有 servlet 都必须实现的方法。

servlet 是运行在 Web 服务器中的小型 Java 程序。servlet 通常通过 HTTP（超文本传输协议）接收和响应来自 Web 客户端的请求。

要实现此接口，可以编写一个扩展 `javax.servlet.GenericServlet` 的一般 servlet，或者编写一个扩展 `javax.servlet.http.HttpServlet` 的 HTTP servlet。

此接口定义了初始化 servlet 的方法、为请求提供服务的方法和从服务器移除 servlet 的方法。这些方法称为生命周期方法，它们是按以下顺序调用的：

1. 构造 servlet，然后使用 `init` 方法将其初始化。
2. 处理来自客户端的对 `service` 方法的所有调用。
3. 从服务中取出 servlet，然后使用 `destroy` 方法销毁它，最后进行垃圾回收并终止它。

除了生命周期方法之外，此接口还提供了 `getServletConfig` 方法和 `getServletInfo` 方法，servlet 可使用前一种方法获得任何启动信息，而后一种方法允许 servlet 返回有关其自身的基本信息，比如作者、版本和版权。

方法总结

Modifier and Type	Method and Description
void	**destroy**() Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
ServletConfig	**getServletConfig**() Returns a **ServletConfig** object, which contains initialization and startup parameters for this servlet.
String	**getServletInfo**() Returns information about the servlet, such as author, version, and copyright.
void	**init**(**ServletConfig** config) Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
void	**service**(**ServletRequest** req, **ServletResponse** res) Called by the servlet container to allow the servlet to respond to a request.

destroy

```
void destroy()
```

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service. This method is only called once all threads within the servlet's `service` method have exited or after a timeout period has passed. After the servlet container calls this method, it will not call the `service` method again on this servlet.

This method gives the servlet an opportunity to clean up any resources that are being held (for example, memory, file handles, threads) and make sure that any persistent state is synchronized with the servlet's current state in memory.

由 servlet 容器调用，指示将从服务中取出该 servlet。此方法仅在 servlet 的 service 方法已退出或者在过了超时期之后调用一次。在调用此方法之后，servlet 容器不会再对此 servlet 调用 service 方法。

此方法为 servlet 提供了一个清除持有的所有资源（比如内存、文件句柄和线程）的机会，并确保任何持久状态都与内存中该 servlet 的当前状态保持同步。

getServletConfig

```
ServletConfig getServletConfig()
```

Returns a `ServletConfig` object, which contains initialization and startup parameters for this servlet. The `ServletConfig` object returned is the one passed to the `init` method.

Implementations of this interface are responsible for storing the `ServletConfig` object so that this method can return it. The `GenericServlet` class, which implements this interface, already does this.

- **Returns:**

  the `ServletConfig` object that initializes this servlet

返回 ServletConfig 对象，该对象包含此 servlet 的初始化和启动参数。返回的 ServletConfig 对象是传递给 init 方法的对象。
此接口的实现负责存储 ServletConfig 对象，以便此方法可以返回该对象。实现此接口的 GenericServlet 类已经这样做了。

- return 初始化此 servlet 的 ServletConfig 对象

getServletInfo

```
String getServletInfo()
```

Returns information about the servlet, such as author, version, and copyright.

The string that this method returns should be plain text and not markup of any kind (such as HTML, XML, etc.).

- **Returns:**

  a `String` containing servlet information

返回有关 servlet 的信息，比如作者、版本和版权。
此方法返回的字符串应该是纯文本，不应该是任何种类的标记（比如 HTML、XML，等等）。

- return 包含 servlet 信息的 String

init

```
void init(ServletConfig config)
   throws ServletException
```

Called by the servlet container to indicate to a servlet that the servlet is being placed into service.

The servlet container calls the `init` method exactly once after instantiating the servlet. The `init` method must complete successfully before the servlet can receive any requests.

The servlet container cannot place the servlet into service if the `init` method

1. Throws a `ServletException`
2. Does not return within a time period defined by the Web server

- **Parameters:**

  `config` - a `ServletConfig` object containing the servlet's configuration and initialization parameters

- **Throws:**

  `ServletException` - if an exception has occurred that interferes with the servlet's normal operation

由 servlet 容器调用，指示将该 servlet 放入服务。

servlet 容器仅在实例化 servlet 之后调用 init 方法一次。在 servlet 可以接收任何请求之前，init 方法必须成功完成。

servlet 容器无法将 servlet 放入服务，如果 init 方法：

1. 抛出 ServletException
2. 未在 Web 服务器定义的时间段内返回

- **参数**

  config 包含 servlet 的配置和初始化参数的 ServletConfig 对象

- **异常**

  Throws ServletException: 如果发生妨碍 servlet 正常操作的异常

service

```
void service(ServletRequest req,
             ServletResponse res)
      throws ServletException,
             IOException
```

Called by the servlet container to allow the servlet to respond to a request.

This method is only called after the servlet's `init()` method has completed successfully.

The status code of the response always should be set for a servlet that throws or sends an error.

Servlets typically run inside multithreaded servlet containers that can handle multiple requests concurrently. Developers must be aware to synchronize access to any shared resources such as files, network connections, and as well as the servlet's class and instance variables. More information on multithreaded programming in Java is available in the Java tutorial on multi-threaded programming.

- **Parameters:**

  `req` - the `ServletRequest` object that contains the client's request

  `res` - the `ServletResponse` object that contains the servlet's response

- **Throws:**

  `ServletException` - if an exception occurs that interferes with the servlet's normal operation

  `IOException` - if an input or output exception occurs

由 servlet 容器调用，以允许 servlet 响应某个请求。

此方法仅在 servlet 的 init() 方法成功完成之后调用。

应该为抛出或发送错误的 servlet 设置响应的状态代码。

servlet 通常运行在可同时处理多个请求的多线程 servlet 容器中。开发人员必须知道要同步对所有共享资源（比如文件、网络连接以及 servlet 的类和实例变量）的访问。有关 Java 中多线程编程的更多信息，可从 the Java tutorial on multi-threaded programming 中获得。

- **参数**

  req 包含客户端请求的 ServletRequest 对象
  res 包含 servlet 的响应的 ServletResponse 对象

- **异常**

  Throws ServletException: 如果发生妨碍 servlet 正常操作的异常

  Throws java.io.IOException: 如果发生输入或输出异常

ServletRequest

英文简介

Defines an object to provide client request information to a servlet. The servlet container creates a `ServletRequest` object and passes it as an argument to the servlet's `service` method.

A `ServletRequest` object provides data including parameter name and values, attributes, and an input stream. Interfaces that extend `ServletRequest` can provide additional protocol-specific data (for example, HTTP data is provided by `HttpServletRequest`.

中文简介

定义将客户端请求信息提供给某个 servlet 的对象。servlet 容器创建 `ServletRequest` 对象，并将该对象作为参数传递给该 servlet 的 `service` 方法。

`ServletRequest` 对象提供包括参数名称、参数值、属性和输入流的数据。扩展 `ServletRequest` 的接口可提供其他特定于协议的数据，例如 `javax.servlet.http.HttpServletRequest` 提供的 HTTP 数据。

方法总结

Modifier and Type	Method and Description
`AsyncContext`	`getAsyncContext()` Gets the AsyncContext that was created or reinitialized by the most recent invocation of `startAsync()` or `startAsync(ServletRequest,ServletResponse)` on this request.
`Object`	`getAttribute(String name)` Returns the value of the named attribute as an `Object`, or `null` if no attribute of the given name exists.
`Enumeration<String>`	`getAttributeNames()` Returns an `Enumeration` containing the names of the attributes available to this request.
`String`	`getCharacterEncoding()` Returns the name of the character encoding used in the body of this request.
`int`	`getContentLength()` Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known ir is greater than Integer.MAX_VALUE.
`long`	`getContentLengthLong()` Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known.
`String`	`getContentType()` Returns the MIME type of the body of the request, or `null` if the type is not known.
`DispatcherType`	`getDispatcherType()` Gets the dispatcher type of this request.
`ServletInputStream`	`getInputStream()` Retrieves the body of the request as binary data using a `ServletInputStream`.
`String`	`getLocalAddr()` Returns the Internet Protocol (IP) address of the interface on which the request was received.
`Locale`	`getLocale()` Returns the preferred `Locale` that the client will accept content in, based on the Accept-Language header.
`Enumeration<Locale>`	`getLocales()` Returns an `Enumeration` of `Locale` objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header.
`String`	`getLocalName()` Returns the host name of the Internet Protocol (IP) interface on which the request was received.
`int`	`getLocalPort()` Returns the Internet Protocol (IP) port number of the interface on which the request was received.
`String`	`getParameter(String name)` Returns the value of a request parameter as a `String`, or `null` if the parameter does not exist.
`Map<String,String[]>`	`getParameterMap()` Returns a java.util.Map of the parameters of this request.
`Enumeration<String>`	`getParameterNames()` Returns an `Enumeration` of `String` objects containing the names of the parameters contained in this request.
`String[]`	`getParameterValues(String name)` Returns an array of `String` objects containing all of the values the given request parameter has, or `null` if the parameter does not exist.
`String`	`getProtocol()` Returns the name and version of the protocol the request uses in the form *protocol/majorVersion.minorVersion*, for example, HTTP/1.1.
`BufferedReader`	`getReader()` Retrieves the body of the request as character data using a `BufferedReader`.
`String`	`getRealPath(String path)` **Deprecated.** As of Version 2.1 of the Java Servlet API, use `ServletContext.getRealPath(java.lang.String)` instead.

String	**getRemoteAddr()** Returns the Internet Protocol (IP) address of the client or last proxy that sent the request.
String	**getRemoteHost()** Returns the fully qualified name of the client or the last proxy that sent the request.
int	**getRemotePort()** Returns the Internet Protocol (IP) source port of the client or last proxy that sent the request.
RequestDispatcher	**getRequestDispatcher(String path)** Returns a **RequestDispatcher** object that acts as a wrapper for the resource located at the given path.
String	**getScheme()** Returns the name of the scheme used to make this request, for example, `http`, `https`, or `ftp`.
String	**getServerName()** Returns the host name of the server to which the request was sent.
int	**getServerPort()** Returns the port number to which the request was sent.
ServletContext	**getServletContext()** Gets the servlet context to which this ServletRequest was last dispatched.
boolean	**isAsyncStarted()** Checks if this request has been put into asynchronous mode.
boolean	**isAsyncSupported()** Checks if this request supports asynchronous operation.
boolean	**isSecure()** Returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.
void	**removeAttribute(String name)** Removes an attribute from this request.
void	**setAttribute(String name, Object o)** Stores an attribute in this request.
void	**setCharacterEncoding(String env)** Overrides the name of the character encoding used in the body of this request.
AsyncContext	**startAsync()** Puts this request into asynchronous mode, and initializes its **AsyncContext** with the original (unwrapped) ServletRequest and ServletResponse objects.
AsyncContext	**startAsync(ServletRequest servletRequest, ServletResponse servletResponse)** Puts this request into asynchronous mode, and initializes its **AsyncContext** with the given request and response objects.

getContentLength

```
int getContentLength()
```

Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known ir is greater than Integer.MAX_VALUE. For HTTP servlets, same as the value of the CGI variable CONTENT_LENGTH.

- **Returns:**

  an integer containing the length of the request body or -1 if the length is not known or is greater than Integer.MAX_VALUE.

返回请求正文的长度（以字节为单位），并使输入流可以使用它，如果长度未知，则返回 -1。对于 HTTP servlet，返回的值与 CGI 变量 CONTENT_LENGTH 的值相同。

- return

  包含请求正文长度的整数，如果长度未知，则返回 -1

getContentType

```
String getContentType()
```

Returns the MIME type of the body of the request, or `null` if the type is not known. For HTTP servlets, same as the value of the CGI variable CONTENT_TYPE.

- **Returns:**

  a `String` containing the name of the MIME type of the request, or null if the type is not known

返回请求正文的 MIME 类型，如果该类型未知，则返回 null。对于 HTTP servlet，返回的值与 CGI 变量 CONTENT_TYPE 的值相同。

- return

  包含请求的 MIME 类型名称的 String，如果该类型未知，则返回 null

> **Q** 什么是MIME？什么是MIME邮件？
>
> **A** MIME, 全称为“Multipurpose Internet Mail Extensions”, 比较确切的中文名称为“多用途互联网邮件扩展”。它是当前广泛应用的一种电子邮件技术规范，基本内容定义于RFC 2045-2049。
>
> 自然，MIME邮件就是符合MIME规范的电子邮件，或者说根据MIME规范编码而成的电子邮件。
>
> 在MIME出台之前，使用RFC 822只能发送基本的ASCII码文本信息，邮件内容如果要包括二进制文件、声音和动画等，实现起来非常困难。MIME提供了一种可以在邮件中附加多种不同编码文件的方法，弥补了原来的信息格式的不足。实际上不仅仅是邮件编码，现在MIME经成为HTTP协议标准的一个部分。

getInputStream

```
ServletInputStream getInputStream()
                            throws IOException
```

Retrieves the body of the request as binary data using a `ServletInputStream`. Either this method or `getReader()` may be called to read the body, not both.

- **Returns:**

  a `ServletInputStream` object containing the body of the request

- **Throws:**

  `IllegalStateException` - if the `getReader()` method has already been called for this request

  `IOException` - if an input or output exception occurred

使用 ServletInputStream 以二进制数据形式获取请求正文。可调用此方法或 getReader 读取正文，而不是两种方法都调用。

- return

  包含请求正文的 ServletInputStream 对象

- Throws

  IllegalStateException: 如果已经为此请求调用 getReader 方法

- Throws

  java.io.IOException: 如果发生输入或输出异常

getParameter

```
String getParameter(String name)
```

Returns the value of a request parameter as a `String`, or `null` if the parameter does not exist. Request parameters are extra information sent with the request. For HTTP servlets, parameters are contained in the query string or posted form data.

You should only use this method when you are sure the parameter has only one value. If the parameter might have more than one value, use `getParameterValues(java.lang.String)`.

If you use this method with a multivalued parameter, the value returned is equal to the first value in the array returned by `getParameterValues`.

If the parameter data was sent in the request body, such as occurs with an HTTP POST request, then reading the body directly via `getInputStream()` or `getReader()` can interfere with the execution of this method.

- **Parameters:**

  `name` - a `String` specifying the name of the parameter

- **Returns:**

  a `String` representing the single value of the parameter

以 String 形式返回请求参数的值，如果该参数不存在，则返回 null。请求参数是与请求一起发送的额外信息。对于 HTTP servlet，参数包含在查询字符串或发送的表单数据中。

只有在确定参数只有一个值时，才应该使用此方法。如果参数可能拥有一个以上的值，则使用 #getParameterValues。

如果将此方法用于一个有多个值的参数，则返回的值等于 getParameterValues 返回的数组中的第一个值。

如果参数数据是在请求正文中发送的，比如发生在 HTTP POST 请求中，则通过 #getInputStream 或 #getReader 直接读取正文可能妨碍此方法的执行。

- 参数

  name 指定参数名称的 String

- 返回值

  return 表示单个参数值的 String

getParameterValues

```
String[] getParameterValues(String name)
```

Returns an array of `String` objects containing all of the values the given request parameter has, or `null` if the parameter does not exist.

If the parameter has a single value, the array has a length of 1.

- **Parameters:**

  `name` - a `String` containing the name of the parameter whose value is requested

- **Returns:**

  an array of `String` objects containing the parameter's values

返回包含给定请求参数拥有的所有值的 String 对象数组，如果该参数不存在，则返回 null。

如果该参数只有一个值，则数组的长度为 1。

- 参数

  name 包含请求其值的参数的名称的 String

- 返回值

  return 包含参数值的 String 对象数组

getReader

```
BufferedReader getReader()
                  throws IOException
```

Retrieves the body of the request as character data using a `BufferedReader`. The reader translates the character data according to the character encoding used on the body. Either this method or `getInputStream()` may be called to read the body, not both.

- **Returns:**

  a `BufferedReader` containing the body of the request

- **Throws:**

  `UnsupportedEncodingException` - if the character set encoding used is not supported and the text cannot be decoded

  `IllegalStateException` - if `getInputStream()` method has been called on this request

  `IOException` - if an input or output exception occurred

使用 BufferedReader 以字符数据形式获取请求正文。读取器根据正文上使用的字符编码转换字符数据。可调用此方法或 #getInputStream 读取正文，而不是两种方法都调用。

- 返回值

  return 包含请求正文的 BufferedReader

- 异常

  Throws UnsupportedEncodingException: 如果使用的字符集编码不受支持，并且无法对文本进行解码
  Throws IllegalStateException: 如果已对此请求调用 #getInputStream 方法
  Throws java.io.IOException: 如果发生输入或输出异常

getLocalPort

```
String getLocalAddr()
```

Returns the Internet Protocol (IP) address of the interface on which the request was received.

- **Returns:**

  a `String` containing the IP address on which the request was received.

返回接收请求的接口的 Internet Protocol (IP) 端口号。

- 返回值

  return 指定端口号的整数

ServletResponse

英文简介

Defines an object to assist a servlet in sending a response to the client. The servlet container creates a `ServletResponse` object and passes it as an argument to the servlet's `service` method.

To send binary data in a MIME body response, use the `ServletOutputStream` returned by `getOutputStream()`. To send character data, use the `PrintWriter` object returned by `getWriter()`. To mix binary and text data, for example, to create a multipart response, use a `ServletOutputStream` and manage the character sections manually.

The charset for the MIME body response can be specified explicitly using any of the following techniques: per request, per web-app (using `ServletContext.setRequestCharacterEncoding(java.lang.String)`, deployment descriptor), and per container (for all web applications deployed in that container, using vendor specific configuration). If multiple of the preceding techniques have been employed, the priority is the order listed. For per request, the charset for the response can be specified explicitly using the `setCharacterEncoding(java.lang.String)` and `setContentType(java.lang.String)` methods, or implicitly using the `setLocale(java.util.Locale)` method. Explicit specifications take precedence over implicit specifications. If no charset is explicitly specified, ISO-8859-1 will be used. The `setCharacterEncoding`, `setContentType`, or `setLocale` method must be called before `getWriter` and before committing the response for the character encoding to be used.

See the Internet RFCs such as RFC 2045 for more information on MIME. Protocols such as SMTP and HTTP define profiles of MIME, and those standards are still evolving.

中文简介

定义辅助 servlet 将响应发送到客户端的对象。servlet 容器创建 `ServletResponse` 对象，并将它作为参数传递给 servlet 的 `service` 方法。

要发送 MIME 正文响应中的二进制数据，请使用 `#getOutputStream` 返回的 `ServletOutputStream`。要发送字符数据，请使用 `#getWriter` 返回的 `PrintWriter` 对象。要混合二进制数据和文本数据，例如要创建 multipart 响应，请使用 `ServletOutputStream` 并手动管理字符部分。

可使用 `#setCharacterEncoding` 和 `#setContentType` 显式指定 MIME 正文响应的 charset，或使用 `#setLocale` 方法隐式指定它。显式指定优先于隐式指定。如果未指定 charset，则将使用 ISO-8859-1。`setCharacterEncoding`、`setContentType` 或 `setLocale` 方法必须在调用 `getWriter` 之前，并且必须在提交采用要使用的字符编码的响应之前调用。

有关 MIME 的更多信息，请参见 Internet RFC，比如 RFC 2045。诸如 SMTP 和 HTTP 之类的协议定义了 MIME 的配置文件，并且那些标准仍在不断发展。

方法总结

Modifier and Type	Method and Description
void	**flushBuffer**() Forces any content in the buffer to be written to the client.
int	**getBufferSize**() Returns the actual buffer size used for the response.
String	**getCharacterEncoding**() Returns the name of the character encoding (MIME charset) used for the body sent in this response.
String	**getContentType**() Returns the content type used for the MIME body sent in this response.
Locale	**getLocale**() Returns the locale specified for this response using the **setLocale(java.util.Locale)** method.
ServletOutputStream	**getOutputStream**() Returns a **ServletOutputStream** suitable for writing binary data in the response.
PrintWriter	**getWriter**() Returns a PrintWriter object that can send character text to the client.
boolean	**isCommitted**() Returns a boolean indicating if the response has been committed.
void	**reset**() Clears any data that exists in the buffer as well as the status code, headers.
void	**resetBuffer**() Clears the content of the underlying buffer in the response without clearing headers or status code.
void	**setBufferSize**(int size) Sets the preferred buffer size for the body of the response.
void	**setCharacterEncoding**(String charset) Sets the character encoding (MIME charset) of the response being sent to the client, for example, to UTF-8.
void	**setContentLength**(int len) Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.
void	**setContentLengthLong**(long len) Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.
void	**setContentType**(String type) Sets the content type of the response being sent to the client, if the response has not been committed yet.
void	**setLocale**(Locale loc) Sets the locale of the response, if the response has not been committed yet.

getContentType

```
String getContentType()
```

Returns the content type used for the MIME body sent in this response. The content type proper must have been specified using `setContentType(java.lang.String)` before the response is committed. If no content type has been specified, this method returns null. If a content type has been specified, and a character encoding has been explicitly or implicitly specified as described in `getCharacterEncoding()` or `getWriter()` has been called, the charset parameter is included in the string returned. If no character encoding has been specified, the charset parameter is omitted.

- **Returns:**

  a `String` specifying the content type, for example, `text/html; charset=UTF-8`, or null

返回用于此响应中发送的 MIME 正文的内容类型。必须在提交响应之前已使用 #setContentType 指定适当的内容类型。如果未指定内容类型，则此方法返回 null。如果已指定内容类型，并且已经如 #getCharacterEncoding 中所述显式或隐式指定了字符编码或者已调用 #getWriter，则返回的字符串中将包含 charset 参数。如果未指定字符编码，则省略 charset 参数。

- 返回值

  return 指定内容类型的 String，例如 text/html; charset=UTF-8，或者返回 null

getOutputStream

```
ServletOutputStream getOutputStream()
                            throws IOException
```

Returns a `ServletOutputStream` suitable for writing binary data in the response. The servlet container does not encode the binary data.

Calling flush() on the ServletOutputStream commits the response. Either this method or `getWriter()` may be called to write the body, not both, except when `reset()` has been called.

- **Returns:**

  a `ServletOutputStream` for writing binary data

- **Throws:**

  `IllegalStateException` - if the `getWriter` method has been called on this response

  `IOException` - if an input or output exception occurred

返回适用于在响应中编写二进制数据的 ServletOutputStream。servlet 容器不会编码二进制数据。

对 ServletOutputStream 调用 flush() 将提交响应。 可调用此方法或 #getWriter 编写正文，而不是两种方法都调用。

- 返回值

  return 用于编写二进制数据的 ServletOutputStream

- 异常

  Throws IllegalStateException: 如果已对此响应调用 getWriter 方法
  Throws java.io.IOException: 如果发生输入或输出异常

getWriter

```
PrintWriter getWriter()
               throws IOException
```

Returns a `PrintWriter` object that can send character text to the client. The `PrintWriter` uses the character encoding returned by `getCharacterEncoding()`. If the response's character encoding has not been specified as described in `getCharacterEncoding` (i.e., the method just returns the default value `ISO-8859-1`), `getWriter` updates it to `ISO-8859-1`.

Calling flush() on the `PrintWriter` commits the response.

Either this method or `getOutputStream()` may be called to write the body, not both, except when `reset()` has been called.

- **Returns:**

  a `PrintWriter` object that can return character data to the client

- **Throws:**

  `UnsupportedEncodingException` - if the character encoding returned by `getCharacterEncoding` cannot be used

  `IllegalStateException` - if the `getOutputStream` method has already been called for this response object

  `IOException` - if an input or output exception occurred

返回可将字符文本发送到客户端的 PrintWriter 对象。PrintWriter 使用 #getCharacterEncoding 返回的字符编码。如果未如 getCharacterEncoding 中所述指定响应的字符编码（即该方法只返回默认值 ISO-8859-1），则 getWriter 会将字符编码更新到 ISO-8859-1。

对 PrintWriter 调用 flush() 将提交响应。

可调用此方法或 #getOutputStream 编写正文，而不是两种方法都调用。

- 返回值

  return 可将字符数据返回到客户端的 PrintWriter 对象。

- 异常

  Throws UnsupportedEncodingException: 如果无法使用 getCharacterEncoding 返回的字符编码
  Throws IllegalStateException: 如果已对此响应对象调用 getOutputStream 方法
  Throws java.io.IOException: 如果发生输入或输出异常

setContentLength

```
void setContentLength(int len)
```

Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.

- **Parameters:**

  `len` - an integer specifying the length of the content being returned to the client; sets the Content-Length header

设置 HTTP servlet 中响应的内容正文的长度，此方法设置 HTTP Content-Length 头。

- **参数**

  len 指定将返回到客户端的内容长度的整数；设置 Content-Length 头

setContentType

```
void setContentType(String type)
```

Sets the content type of the response being sent to the client, if the response has not been committed yet. The given content type may include a character encoding specification, for example, `text/html;charset=UTF-8`. The response's character encoding is only set from the given content type if this method is called before `getWriter` is called.

This method may be called repeatedly to change content type and character encoding. This method has no effect if called after the response has been committed. It does not set the response's character encoding if it is called after `getWriter` has been called or after the response has been committed.

Containers must communicate the content type and the character encoding used for the servlet response's writer to the client if the protocol provides a way for doing so. In the case of HTTP, the `Content-Type` header is used.

- **Parameters:**

  `type` - a `String` specifying the MIME type of the content

设置将发送到客户端的响应的内容类型，如果该响应尚未提交。给定内容类型可能包含字符编码规范，例如 text/html;charset=UTF-8。如果在调用 getWriter 之前调用此方法，则只根据给定内容类型设置响应的字符编码。

可重复调用此方法来更改内容类型和字符编码。如果在已提交响应之后调用此方法，则此方法没有任何效果。如果在已调用 getWriter 之后或者在已提交响应之后调用此方法，则该方法不会设置响应的字符编码。

容器必须让客户端了解用于 servlet 响应的 writer 的内容类型和字符编码，如果协议提供了实现上述操作的方法。在使用 HTTP 的情况下，使用 Content-Type 头。

- **参数**

  type　指定内容的 MIME 类型的 String

reset

```
void reset()
```

Clears any data that exists in the buffer as well as the status code, headers. The state of calling `getWriter()` or `getOutputStream()` is also cleared. It is legal, for instance, to call `getWriter()`, `reset()` and then `getOutputStream()`. If `getWriter()` or `getOutputStream()` have been called before this method, then the corrresponding returned Writer or OutputStream will be staled and the behavior of using the stale object is undefined. If the response has been committed, this method throws an `IllegalStateException`.

- **Throws:**

  `IllegalStateException` - if the response has already been committed

清除缓冲区中存在的所有数据以及状态代码和头。如果已提交响应，则此方法将抛出 IllegalStateException。

- **异常**

  Throws　IllegalStateException: 如果已提交响应

HttpServletRequest

英文简介

中文简介

HttpServletResponse

英文简介

中文简介

ServletConfig

英文简介

中文简介

ServletContext

英文简介

中文简介