**`Object getAttribute(String name)`** – получает значение атрибута по имени;

**`Enumeration getAttributeNames()`** – получает список имен атрибутов;

**`void setAttribute(String name, Object object)`** – добавляет атрибут и его значение в контекст;

**`void removeAttribute(String name)`** – удаляет атрибут из контекста;

**`ServletContext getContext(String uripath)`** – позволяет получить доступ к контексту других ресурсов данного контейнера сервлетов;

**`String getServletContextName()`** – возвращает имя сервлета, которому принадлежит данный объект интерфейса **`ServletContext`**.

Используя объект **`ServletContext`**, можно регистрировать события сервлета, сессии и запроса.

## Интерфейс ServletConfig

Ранее уже упоминался метод **`getServletConfig()`**, но не было сказано об интерфейсе **`ServletConfig`**, с помощью которого контейнер сервлетов передает информацию сервлету в процессе его инициализации.

Некоторые методы класса:

**`String getServletName()`** – определение имени сервлета;

**`Enumeration getInitParameterNames()`** – определение имен параметров инициализации сервлета из дескрипторного файла **`web.xml`**;

**`String getInitParameter(String name)`** – определение значения конкретного параметра по его имени.

Чтобы задать параметры инициализации сервлета **`MyServlet`**, необходимо в тег <**`servlet`**> его описания вложить тег <**`init-param`**> с описанием имени и значения параметра в виде:

```
<servlet>
  <servlet-name>MyServletname</servlet-name>
  <servlet-class>chapt18.MyServlet</servlet-class>
            <init-param>
                <param-name>mail.smtphost</param-name>
                <param-value>mail.bsu</param-value>
            </init-param>
            <init-param>
                <param-name>mail.smtpport</param-name>
                <param-value>25</param-value>
            </init-param>
</servlet>
```

Тогда для доступа к параметрам инициализации сервлета и их дальнейшего использования можно применить следующую реализацию метода **`init()`** сервлета:

```
public void init() throws ServletException {
    ServletConfig sc = getServletConfig();

      // определение набора имен параметров инициализации
         Enumeration e = sc.getInitParameterNames();
```

```
        while(e.hasMoreElements()) {
    // определение имени параметра инициализации
            String name = (String)e.nextElement();
    // определение значения параметра инициализации
            String value = sc.getInitParameter(name);
            //...
        }
}
```

Такими же возможностями обладает и объект **ServletContext**, который содержит практически всю информацию о среде, в которой запущен и выполняется сервлет, например:

```
getServletContext().getInitParameter("mail.smtpport");
```

## Интерфейсы ServletRequest и HttpServletRequest

Информация от компьютера клиента отправляется серверу в виде объекта запроса типа **HttpServletRequest**. Данный интерфейс является производным от интерфейса **ServletRequest**. Используя методы интерфейса **ServletRequest**, можно получить много дополнительной информации, в том числе и о сервлете и деталях протокола HTTP, закодированной и упакованной в запрос:

**String getCharacterEncoding()** – определение символьной кодировки запроса;

**String getContentType()** – определение MIME-типа (Multipurpose Internet Mail Extension) пришедшего запроса;

**String getProtocol()** – определение названия и версии протокола;

**String getServerName()**, **getServerPort()** – определение имени сервера, принявшего запрос, и порта, на котором запрос был принят сервером соответственно;

**String getRemoteAddr()**, **getRemoteHost()** – определение IP-адреса клиента, от имени которого пришел запрос, и его имени соответственно;

**String getRemoteUser()** – определение имени пользователя, выполнившего запрос;

**ServletInputStream getInputStream()**, **BufferedReader getReader()** – получение ссылки на поток, ассоциированный с содержимым полученного запроса. Первый метод возвращает ссылку на байтовый поток **ServletInputStream**, а второй – на объект **BufferedReader**. В результате можно прочитать любой байт из полученного объекта-запроса. Если метод **getReader()** был вызван после вызова **getInputStream()** для этого запроса, то генерируется исключение **IllegalStateException** и наоборот.

При обращении к серверу, как правило, передаются параметры и их значения. Для разбора параметров и извлечения их значений применяются методы:

**String getParameter(String name)** – определение значения параметра по его имени или **null**, если параметр с таким именем не задан;

**String[] getParameterValues(String name)** – определение всех значений параметра по его имени;