CSP/CCP 语言规范
Language Specification

版本日期	作者	内容
2001-01-18	HD	文档初稿。
2001-11-21	HD	0.9 版本
2010-08-26	HD	1.0 版本
2011-07-18	HD	1.1 版本
2014-10-13	HD	2.0 版本
2015-08-18	HD	2.1 版本； 增加 out.print,<csp:out.print id="" scopy="" /> 输出到控制台命令；
2017-04-19	HD	2.2 版本； 增加支持多级 <csp:if:xx /> 嵌套使用功能； 增加支持 <csp:else:if:xx /> 标签功能，用于替换支持原来一级类似 else if () 语法功能； 增加支持在<?csp ?>代码段，编写类似 C/C++语言的 if (){}else if (){}else{} 语句； 增加支持 <csp:break>和<csp:continue>标签； 修改文档中几处已经笔误；

H.D akee.yang@gmail.com

目录

前言

CSP（C++ Server Pages）是一套基于 C++的服务端页面开发技术，是基于 MYCP 平台的一套 web 应用开发标签语言；MYCP 服务平台利用标准 C++语言，实现 HTTP 服务器和 C++ web 容器功能，配合 CSP+Servlet+APP+CDBC（以下简称 CSP+）。

利用 C++语言的强大功能，“JAVA/J2EE 和 PHP 等语言能做的，CSP+，做的更好！”

CCS（C++ Client Pages）是一套基于客户端页面开发技术，是基于 MYC 开发平台的一套 WEB 浏览器应用开发的标签语言；MYC 开发平台把 MYCP 的相关功能封装成客户端形式、可独立运行部署；

利用 CCP，配合 CSP，可以实现 WEB 客户端应用，同时拥有 WEB 快速开发部署，和 C++高效，稳定，安全等优点。

CSP（包含 CCP，以下同）语法是标准 XML 格式标签，CSP 帮助开发人员严格按照 MVC（模型-视图-控制器）框架进行 web 开发，通过在 HTML 文件嵌套标准 CSP 标签，调用业务功能组件；

CSP 语法简单，普通开发人员，不用半小时可以学习完成，而熟悉 JSP、PHP 等其他页面开发技术，十分钟足够；

最新 CSP/CCP 2.1 规范可以查看以下地址：

https://git.oschina.net/akee/mycp/raw/master/doc/CSP_2_1_0.pdf

第1章 CSP/CCP 入门

CSP 是标准 XML 格式标签脚本语言，通过在 HTML 增加 CSP 标签，实现页面输出、业务处理、页面流转等服务端功能；

CSP 帮助开发人员，严格执行 MVC 开发模式，从而让业务系统的 UI 和业务逻辑分开，让开发团队角色更加明确；

在大型企业 web 应用系统中，利用 MVC 框架，将极大提高系统的开发效率和和系统可维护性。

1.1 基础知识

1.1.1 MYCP

MYCP 是一套集 HTTP 服务器、C++ web（CSP+）容器功能于一身的 web 应用开发平台；MYCP 采用标准 C++语言编写，集合所有 C++语言的优势，高效率、稳定、功能强大。

CSP 文件及其他 CSP+组件部署到 MYCP 平台提供 web 服务。

1.1.2 MYC

MYC 基于 MYCP 技术而开发，开发好的 CSP 应用，利用 MYC 可以单独运行，实现本地客户端应用。

1.1.3 CSP 文件

统一使用.csp （小写）扩展名，文件内容使用标准 HTML 语言+CSP 标签语句，在某些环境中，可能使用 XML+CSP 标签，比如 ajax。

1.1.4 开发流程

- 需要先确定需求；
- 然后根据系统功能，进行划分，设计出需要多少 CSP 或者 Servlet 页

面；

- 列出对应的业务处理模型（C++ APP 应用组件）；
- 如果需要数据库功能，利用已经有的 CDBC 驱动程序操作数据库系统。
- 根据设计后，对开发人员进行分工，UI 人员设计编写 HTML 页面，其他人员开发 C++ APP，或者 C++ Servlet；
- HTML 经过简单修改，增加相应 CSP 标签，连接 APP 应用组件；
- 测试、修改、部署发布、更新完善；

1.1.5 虚拟主机

MYCP CSP 支持虚拟主机技术，可以实现指定 IP 地址、端口、域名、主机名，或者任何二种之间的组合来实现不同 Web 系统；

1.1.6 CSP 例子

下载最新版本的 MYCP，在 bin/web/samples 目录下有所有的 CSP 源码例子；CSP 源码例子基本展现了所有 CSP 语言规范和特性，参考学习 CSP 例子，是最好的学习 CSP 方法。

1.2 CSP+应用方向

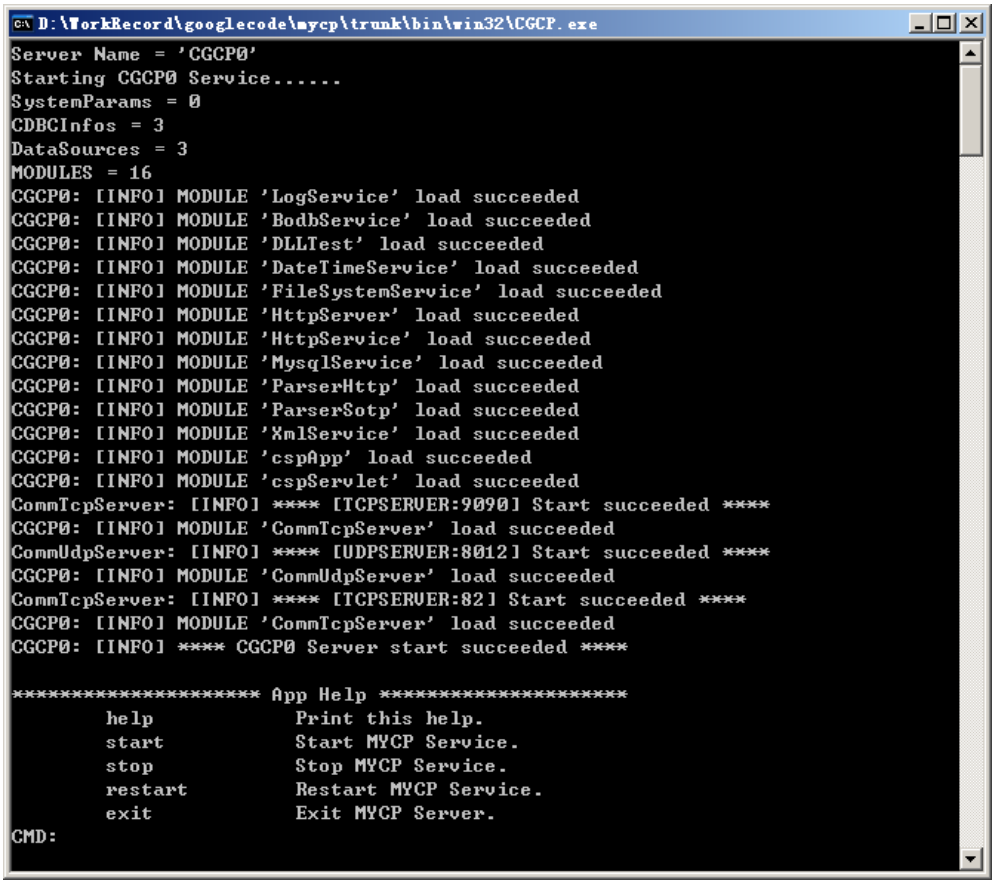
CSP+技术可以应用于以下方向：

- 普通公司网站
- 企业 Web 应用开发
- 大型分布式应用系统
- 异构系统通讯应用（例如 C/S 跟 B/S 系统，或者 web 系统调用 C++算法等）
- TV 端（类似路由器、电视盒等硬件）WEB 服务器
- ……

1.3 测试 CSP 例子

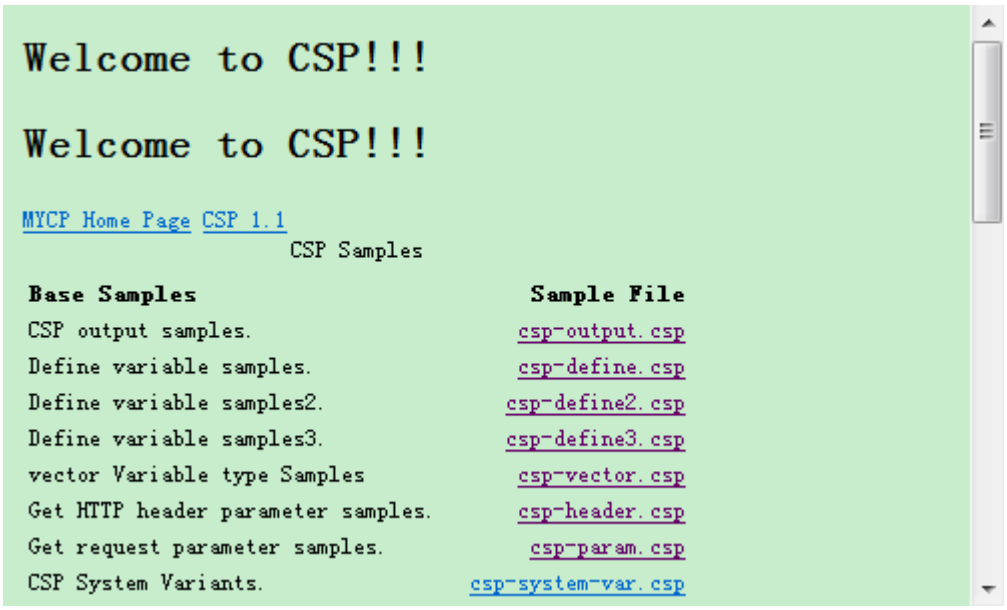
MYCP 服务方式：

直接启动 CGCP 应用服务，然后打开浏览器，输入访问 CSP 地址 http://127.0.0.1:82，如下图：


D:\WorkRecord\googlecode\mycp\trunk\bin\win32\CGCP.exe
Server Name = 'CGCP0'
Starting CGCP0 Service......
SystemParams = 0
CDBCInfos = 3
DataSources = 3
MODULES = 16
CGCP0: [INFO] MODULE 'LogService' load succeeded
CGCP0: [INFO] MODULE 'BodbService' load succeeded
CGCP0: [INFO] MODULE 'DLLTest' load succeeded
CGCP0: [INFO] MODULE 'DateTimeService' load succeeded
CGCP0: [INFO] MODULE 'FileSystemService' load succeeded
CGCP0: [INFO] MODULE 'HttpServer' load succeeded
CGCP0: [INFO] MODULE 'HttpService' load succeeded
CGCP0: [INFO] MODULE 'MysqlService' load succeeded
CGCP0: [INFO] MODULE 'ParserHttp' load succeeded
CGCP0: [INFO] MODULE 'ParserSotp' load succeeded
CGCP0: [INFO] MODULE 'XmlService' load succeeded
CGCP0: [INFO] MODULE 'cspApp' load succeeded
CGCP0: [INFO] MODULE 'cspServlet' load succeeded
CommTcpServer: [INFO] **** [TCPSERVER:9090] Start succeeded ****
CGCP0: [INFO] MODULE 'CommTcpServer' load succeeded
CommUdpServer: [INFO] **** [UDPSERVER:8012] Start succeeded ****
CGCP0: [INFO] MODULE 'CommUdpServer' load succeeded
CommTcpServer: [INFO] **** [TCPSERVER:82] Start succeeded ****
CGCP0: [INFO] MODULE 'CommTcpServer' load succeeded
CGCP0: [INFO] **** CGCP0 Server start succeeded ****
********** App Help **********
help Print this help.
start Start MYCP Service.
stop Stop MYCP Service.
restart Restart MYCP Service.
exit Exit MYCP Server.
CMD:

MYC 应用方式：

直接启动 MYC 应用服务，如下图：



第2章 基本 CSP 语句

2.1 二种 CSP 代码类型

CSP 支持下列二种不同代码类型：

编程式代码类型：

```
<?csp
编程类型代码段
?>
```

编程式代码类型支持单行（<?csp include(“”); ?>）和多行（类似上面代码段）二种编写原则；

编程式代码结尾以半角分号“;”结束；

标签式代码类型：

```
<%-- … --%>
<csp … />
```

标签式代码类型必须独行编写原则，即一条指令一行编写完成；

在一个 CSP（HTML）页面代码中，支持同时混合二种不同的编写类型代码。

2.2 注释语句 CSP Comment

```
<?csp
// 单行注释
/* 单行注释 */
/*
多行注释
*/
?>

<%-- Comment 单行注释 --%>
<%--
  Comment 多行注释
--%>
```

2.3 输出语句 Output

2.3.1 echo $variable;

```
<?csp
echo ‘hello world.’;
?>
```

输出’hello world.’字符串到当前网页上。

2.3.2 <%=variable%>

```
<%=variable%>
```

variable	可以是用户变量$variable，临时变量_$variable，也可以是普通字符串"variable"；（以下同）

例子 Sample: <%=$var1%>

例子 Sample: <%=P$UserName%>

2.3.3 <csp:write id="" scopy="" />

```
<csp:write id="" scopy="" />
```

打印 id 的值;

id	可以是用户变量$variable，临时变量_$variable，也可以是普通字符串"variable"；（以下同）
scopy	如果是用户变量，指定变量存放区域；默认不填为 page 区域；

2.3.4 <csp:write name="" property="" />

```
<csp:write name="" property="" />
```

打印 C++ APP 应用组件，指定属性的参数值。

name	APP 应用组件名称
property	属性名称

2.3.5 out.print $variable;

```
<?csp
out.print  'hello world.' ;
?>
```

输出’hello world.’字符串到控制台窗口上。

2.3.6 <csp:out.print id="" scopy="" />

```
<csp:out.print id="" scopy="" />
```

输入 id 的值到控制台窗口上;

id	可以是用户变量$variable，临时变量_$variable，也可以是普

	通字符串"variable"；（以下同）
scopy	如果是用户变量，指定变量存放区域；默认不填为 page 区域；

2.3.7 <csp:out.print name="" property="" />

```
<csp:out.print name="" property="" />
```

打印 C++ APP 应用组件，指定属性的参数值到控制台窗口上。

name	APP 应用组件名称
property	属性名称

2.3.8 提示

- <%=%>可以存在于 CSP 文件内任何位置，用于打印指定信息；不包含在<%-- --%>注释语句内；
- <csp:write />语句必须占满一行位置；
- 可以组合使用，如：<%=$var1%><csp:write id="$var2" />，中间有允许有其他字符；

2.4 例子 Example

```
<?csp
echo 1000;
echo 'Hi, how are you?';
echo $var0;
?>
```

以上例子，分别输出 1000 的值，'Hi, how are you?'字符串，和$var0 变量的值到网页上；

```
<%=$var0%>
```

输出$var0 变量的值到网页上；

```
<csp:write id="$var0" />
```

输出$var0 变量的值到网页上；

```
<csp:write name="A$xml" property="item" />
```

输出 A$xml 应用组件的 item 参数的值到网页上；

```
<?csp
```

```
out.print ‘Hi, how are you?’;
our.ptin $var0;
?>
```

以上例子，分别输出 ‘Hi, how are you?’字符串，和$var0 变量的值到控制台窗口上；

```
<csp:out.print id=”$var0” />
```

输出$var0 变量的值到控制台窗口上；

第3章 变量 CSP Variable

3.1 CSP 变量类型 Type

CSP 包括有以下变量类型：

变量类型	定义头	描述
用户变量	$	例如$var_user
系统变量	S$	用于获取系统信息，请看《附录系统变量表》 例如 S$ ParamNames、S$RequestURL
PARAMETER 变量	P$	用于获取用户表单信息； 例如 P$UserName、P$UserAccount
HEADER 变量	H$	用于获取 HTTP 头信息；后面由用户自己确定； 例如 H$Content-Type
临时变量	_$	请看《附录临时变量表》，详细看各 CSP 标签使用； 例如_$index、_$value、_$result
应用组件变量	A$	用于访问 C++ APP 应用组件，通过 apps.xml 配置；
数据源变量	D$	用于访问数据源接口，通过 datasources.xml 配置；

3.2 变量存放区域 scopy

CSP 变量可以存放于以下不同区域：

page	页面区域，对当前页面有效；默认存放区域
request	请求区域，当前 request 有效，可以用于 forward
session	会话区域，当前整个 SESION 周期有效
application	应用区域，当前应用（虚拟主机内）有效
server	WEB SERVER 区域，当前 HTTP SERVER 下所有应用有效

3.3 变量数据类型 DataType

CSP 支持以下数据类型：

int	C++ int 数据类型
bitint	C++ __int64/long long 数据类型
time	C++ time_t 数据类型
float	C++ double 数据类型
boolean	C++ bool 数据类型
string	C++ std::string 数据类型；**默认数据类型**
vector	C++ std::vector 数据类型，可以包含以上基础数据类型
map	C++ std::map 数据类型，可以包含以上基础数据类型

3.4 定义 Define

3.4.1 定义变量 Define Variable

```
<?csp
$var1 =’ string’ ;
$var2 = 12;
?>
```

或者；

```
<csp:define id="var_name" value="var_value" />
```

定义变量 var_name，初始化"var_value"，默认使用字符串类型；

```
<csp:define id="var_name" type="var_type" value="var_value" />
```

定义变量 var_name，初始化"var_value"，指定 type 类型；

```
<csp:define id="var_name" name="app_name" property="app_property" />
```

定义变量 var_name，初始化值，为 app_name 应用的 app_property 参数值；

3.4.2 定义只读变量(常量) Define Constant

```
<csp:define id="cons_name" value="var_value" property="const" />
```

定义常量 cons_name，该常量值初始化后，不可更改；

3.4.3 提示

- 用户定义变量必须以$开头，如$var1，依此类推；
- CSP 支持首先使用定义变量功能，即不用通过定义，直接使用指定变量名称保存；
- vector 变量只能使用 0 1 2 以此类推整数序号，如%var[0] = 32;
- map 变量可以使用任意字符串序号，如 echo $var4['age'];

3.5 例子 Example

```
<?csp
$var1 = 'China';
$var2 = 2011;
$varbig = 20110102030405;
$varfloat = 123.45;

$var3 = vector();
$var3[0] = 'Computer';
$var3[1] = 'Telephone';

$var4 = map();
$var4['age'] = 32;
```

```
$var4[‘name’] = ‘akee’;

?>
```

以上例子，分别定义一个字符串变量$var1，一个整数变量$var2，一个 64 位大整数$varbig，一个浮点数$varfloat，一个 vector 变量$var3 和一个 map 变量$var4。

```
<csp:define id="$var1" value="100" />
```

定义字符串类型变量$var1，值为"100"；

```
<csp:define id="$var1" type="int" value="100" />
```

定义整数类型变量$var1，值为 100；

```
<csp:define id="$max_second" value="1000" type="int" property="const" />
```

定义一个常量$max_secnd，常量值为 1000，该值定义后不可更改。

第4章 比较控制 if

CSP 的 if 类似 C++的 if 语句，支持类似 if elseif else 多级控制。

4.1 格式 Format

```
<csp:if:== id="var_name" value="compare_value" />
<csp:else:if:== name="app_name" property="app_property" value="compare_value" />
<csp:else:if:== name="app_name" property="app_property" id="compare_id" />
<csp:else:if:== id="compare_id" name="app_name" property="app_property" />
```

4.2 简单比较 Simple Compare

```
<csp:if:== id="var_name" value="compare_value" />
...
<csp:end>
```

true: 执行…语句，直到 end 语句结束。

4.3 带 else 语句 if else

```
<csp:if:== id="var_name" value="compare_value" />
 if true: ...
<csp:else:if:== id="var_name" value="compare_value" />
 else if true: ...
<csp:if:else>
 else: ...
<csp:end>
```

if true: 执行 true: ...语句

else if true: 执行 else if true: …语句

if false: 执行 else: ...语句

4.4 多级比较 Multi-Level Compare

```
<csp:if:== id="var_name" value="compare_value" />
...
  <csp:if:== id="var_name" value="compare_value" />
        ...
  <csp:end>
<csp:if:else>
...
<csp:end>
```

从上往下执行，比较成功，执行对应...语句后，退出<csp:end>

4.5 全部语句 All csp:if Compare

```
<csp:if:== id="var_name" value="compare_value" />
<csp:else:if:== id="var_name" value="compare_value" />
```

判断是否相等 Compare if equal.

```
<csp:if:!= id="var_name" value="compare_value" />或
<csp:if:<> id="var_name" value="compare_value" />
<csp:else:if:!= id="var_name" value="compare_value" />或
```

```
<csp:else:if:<> id="var_name" value="compare_value" />
```

判断是否不相等 Compare if not equal

```
<csp:if:> id="var_name" value="compare_value" />
<csp:else:if:> id="var_name" value="compare_value" />
```

判断是否大于 Compare if greater.

```
<csp:if:>= id="var_name" value="compare_value" />
<csp:else:if:>= id="var_name" value="compare_value" />
```

判断是否大于或者等于 Compare if greater or equal.

```
<csp:if:< id="var_name" value="compare_value" />
<csp:else:if:< id="var_name" value="compare_value" />
```

判断是否小于 Compare if less.

```
<csp:if:<= id="var_name" value="compare_value" />
<csp:else:if:<= id="var_name" value="compare_value" />
```

判断是否小于或者等于 Compare if less or equal.

```
<csp:if:&& id="var_name" value="compare_value" />
<csp:else:if:&& id="var_name" value="compare_value" />
<csp:if:|| id="var_name" value="compare_value" />
<csp:else:if:|| id="var_name" value="compare_value" />
```

```
<csp:if:empty id="var_name" />
<csp:else:if:empty id="var_name" />
```

判断是否为空 Compare variable if empty.

```
<csp:if:notEmpty id="var_name" />
<csp:else:if:notEmpty id="var_name" />
```

判断是否不为空 Compare variable if not empty.

```
<csp:if:exist id="var_name" />
<csp:else:if:exist id="var_name" />
```

判断变量是否存在 Compare variable if exist.

```
<csp:if:notExist id="var_name" />
<csp:else:if:notExist id="var_name" />
```

判断变量是否不存在 Compare variable if not exist.

```
<csp:if:else>
```

判断为 false，执行语句 Compare all else.

```
<csp:end>
```

结束语句 end.

4.6 CSP 代码 if 比较

支持以下各种复杂多级嵌套 if 代码使用功能。

```
if ($var1==1)
{
}else if ($var1==2)
{
   if ($var2==' abc' )
   {
   }else if ($var2==' 123' )
   {
   }else
   {
   }
}else
{
   if ()
   {
   }else if ()
```

```
    {
    }else
    {
    }
}
```

所有比较代码如下。

```
if ($var1==1)  // 比较相等
if ($var1!=1)  // 比较不相等
if ($var1<>1)  // 比较不相等
if ($var1>=1)  // 比较大于相等
if ($var1>1)   // 比较大于
if ($var1<=1)  // 比较小于相等
if ($var1<1)   // 比较小于
```

4.7 例子 Example

```
<csp:define id="$var1" value="value1" />
define $var1 = "<%=$var1%>".<br>

<b>Compare 1</b><br>
<csp:if:== id="$var1" value="varue2" />
    csp:if:== $var1 == "varue2" : true<br>
<csp:end>

<b>Compare 2</b><br>
<csp:if:== id="$var1" value="varue2" />
    csp:if:== $var1 == "varue2" : true<br>
<csp:if:else>
    csp:if:== $var1" == "varue2" : false<br>
<csp:end>

<b>Compare 3</b><br>
<csp:if:== id="$var1" value="value1" />
    csp:if:== $var1 == "value1" : true<br>

<csp:if:== id="$var1" value="value2" />
    csp:if:== $var1 == "value2" : true<br>
```

```
<csp:if:== id="$var1" value="value3" />
    csp:if:== $var1 == "value3" : true<br>

<csp:if:else>
    csp:if:else : $var1 == <%=$var1%><br>

<csp:end>
```

更多例子可以参考 web/samples/csp-if.csp 文件内容。

第5章 循环 while

csp:while 跟 csp:if 语句完全一样比较方式，遇到 csp:continue 重新执行循环，遇到 csp:break 退出循环；

5.1 格式 Format

```
<csp:while:== id="var_name" value="compare_value" />
...
<csp:end>
```

循环比较变量 id 跟值 value 的值；

true: 执行...语句，直到 false，退出<csp:end>；

5.2 临时变量 Temp Variable

_$index	保存当前索引序列数，从 0 开始，while 每循环一次，_$index 加 1；为避免进行死循环，最大可以执行 1000 次

5.3 全部语句 All csp:while Compare

```
<csp:while:== id="var_name" value="compare_value" />
```

循环判断是否相等

```
<csp:while:!= id="var_name" value="compare_value" />或
<csp:while:<> id="var_name" value="compare_value" />
```

循环判断是否不相等

```
<csp:while:> id="var_name" value="compare_value" />
```

循环判断是否大于

```
<csp:while:>= id="var_name" value="compare_value" />
```

循环判断是否大于或者等于

```
<csp:while:< id="var_name" value="compare_value" />
```

循环判断是否小于

```
<csp:while:<= id="var_name" value="compare_value" />
```

循环判断是否小于或者等于

```
<csp:while:&& id="var_name" value="compare_value" />
<csp:while:|| id="var_name" value="compare_value" />
```

```
<csp:while:empty id="var_name" />
```

循环判断是否为空

```
<csp:while:notEmpty id="var_name" />
```

循环判断是否不为空

```
<csp:while:exist id="var_name" />
```

循环判断是否存在变量

```
<csp:while:notExist id="var_name" />
```

循环判断是否不存在变量

```
<csp:end>
```

结束语句

5.4 例子 Example

```
<h2>csp:while:== Samples</h2>
```

```
<csp:define id="$var1" type="int" value="1" />

$var1 = <%=$var1%><br>
<csp:while:== id="$var1" value="1" />
    csp:while:== $var1 == 1 : true<br>
    <csp:++ id="$var1" />
    $var1 ++<br>
<csp:end>

<h2>csp:while:!= Samples</h2>

<csp:define id="$var2" type="int" value="1" />

<csp:while:!= id="$var2" value="10" />
    _$index = <%=_$index%>;
    $var2 = <%=$var2%>; notEqual 10: true;
    <csp:++ id="$var2" />
    continue;<br>
<csp:end>
```

更多例子可以参考 csp-while.csp 文件内容。

第6章 foreach 循环

foreach 用于处理 VECTOR 和 MAP 变量,遇到 csp:continue 重新执行循环,遇到 csp:break 退出循环;

6.1 格式 Format

```
<csp:foreach id="var_vector" />
...
<csp:end>
```

循环处理 VECTOR 和 MAP 类型变量 id 的值; 同时执行...语句

6.2 临时变量 Temp Variable

_$index	当前索引序列 Current Index Variable
_$value	当前索引值 Current Value Variable
_$filename	文件名称 用于<csp:foreach id="S$UploadFiles" />; (以下同)
_$filepath	文件保存路径
_$filesize	文件大小
_$filetype	文件类型

6.3 例子 Example

```
<csp:foreach id="S$Heads" />
    <%=_$index%>: <%=_$value%><br>
<csp:end>
```

第7章 其他控制 Other Control

7.1 退出循环 break

退出循环语句，用于退出 csp:while 或 csp:foreach 循环语句。

7.1.1 格式 Format

```
<csp:break>
```

无条件退出；

```
<csp:break id="var_name" value="compare_value" />
```

如果 id 等于 value 值，退出循环；

```
<csp:break name="app_name" property="app_property" value="compare_value" />
```

如果应用组件的参数等于 value 值，退出循环；

```
<csp:break name="app_name" property="app_property" id="compare_id" />
<csp:break id="compare_id" name="app_name" property="app_property" />
```

如果应用组件参数等于 id 值，退出循环；

7.2 重新执行循环 contine

重新执行循环语句，用于 csp:while 和 csp:foreah 循环，跳过后面语句，直接重新执行循环。

7.2.1 格式 Format

```
<csp:continue>
```

无条件，重新执行循环；

```
<csp:continue id="var_name" value="value" />
```

如果 id 等于 value，重新执行循环；

```
<csp:continue name="app_name" property="app_property" value="value" />
```

如果应用组件参数等于 value 值，重新执行循环；

```
<csp:continue name="app_name" property="app_property" id="compare_id" />
<csp:continue id="compare_id" name="app_name" property="app_property" />
```

如果参数等于 id 值，重新执行循环；

第8章 CSP 运算符 Operator

8.1 赋值 equal

```
<?csp
$var_int = 2;
$var1 = 'abc';
?>
或者：
<csp:= id="" value="" />
```

id 变量等于 value 的值;

8.2 加 增加 add

```
<?csp
$var_int += 200;
$var1 += 'abc';
?>
或者：
<csp:+= id="" value="" />
```

id 变量加上 value 的值;

string：添加字符串到末尾

vector: 添加元素到末尾

8.3 减 subtract

```
<?csp
$var_int -= 100;
?>
或者：
<csp:-= id="" value="" />
```

id 变量减去 value 的值;

8.4 乘 multiplication

```
<?csp
$var_int *= 100;
?>
或者：
<csp:*= id="" value="" />
```

id 变量乘以 value 的值;

8.5 除 division

```
<?csp
$var_int /= 100;
?>
或者：
<csp:/= id="" value="" />
```

id 变量除以 value 的值;

8.6 模 modulus

```
<?csp
$var_int %= 100;
?>
或者：
<csp:%= id="" value="" />
```

id 变量取模 value 的值;

8.7 加 1 increate

```
<?csp
$var_int++;
?>
或者：
<csp:++ id="" />
```

id 变量值加 1，变量必须为 int 整数类型，非整数类型转换为整数类型;

8.8 减 1 decrease

```
<?csp
$var_int--;
?>
或者：
<csp:-- id="" />
```

id 变量值减 1，变量必须为 int 整数类型，非整数类型转换为整数类型；

8.9 例子 Example

```
define $var1 = "text1"<br>
<csp:define id="$var1" value="text1" />
$var1 += "text2"<br>
<csp:+= id="$var1" value="text2" />
$var1 = <%=$var1%><br>

define vector type $var2 = "text1"<br>
<csp:define id="$var2" value="text1" type="vector" />
$var2 += "text2"<br>
<csp:+= id="$var2" value="text2" />

<csp:sizeof id="$var2" />
$var2 size = <%=_$size%><br>
<csp:foreach id="$var2" />
    <%=_$index%>: <%=_$value%><br>
<csp:end>
```

更多例子可以参考 web/samples/ csp-operator.csp 、 csp-define2.csp 和 csp-define.csp 文件内容。

第9章 其他变量操作 Basic Operate

9.1 判断变量是否为空 empty

```
<?csp
$var = is_empty(id);
?>
或者;
<csp:empty id="" out="var_saveto" />
```

判断 id 变量是否为空，放到 value 变量中；id 变量不存在，返回 true；

9.2 清空变量的值 reset

```
<?csp
reset(id);
?>
或者;
<csp:reset id="" scopy="" />
```

清空 id 变量的值；

9.3 获取变量的大小 sizeof

```
<?csp
$var = sizeof(id);
?>
或者;
<csp:sizeof id="var_name" />
```

获取 id 变量的 size，放到_%size 临时变量中；比如 VECTOR 的 size 大小、string 的长度等；id 变量不存在，返回-1；

9.4 获取变量的类型 typeof

```
<?csp
$var = typeof(id);
```

```
?>
或者;
<csp:typeof id="var_name" out="var_saveto" />
```

获取 id 变量类型信息，放到 out 变量中；类型信息如下，id 变量不存在，返回 null；

int,bitint,time,boolean,float,string,vector,map,app

9.5 转换变量类型 totype

```
<?csp
to_type(id, type);
?>
或者;
<csp:totype id="" type="" />
```

把 id 变量转换为 type 指定类型；

9.6 获取变量指定序列值 index

```
<?csp
$var = $var_vector[index];
$var = $var_map[ 'index' ];
?>
或者;
<csp:index id="" index="" out="" />
```

获取 id 变量的指定 index 值,存放到 out 用户变量中;

9.7 例子 Example

第10章　页面操作 Page Operate

10.1 包含页面 Include

```
<?csp include(url); ?>
或者；
<page:include url="" />
```

包括 url 页面，支持动态 CSP、静态 html 等页面文件。

10.2 设置页面类型 Set ContentType

```
<page:contentType type="" />
```

设置当前页面的 ContentType 内容，默认为"text/html"

10.3 退出当前页面 Return Current Page

```
<page:return>
```

退出当前页面，不执行后面操作

10.4 清空输出 Reset Output

```
<page:reset>
```

清空页面输出内容；

10.5 转发跳转 Forward

10.5.1 格式 Format

```
<?csp forward(url); ?>
或者；
<page:forward url="" />
```

转发跳转操作；执行该操作，会自动退出该页面后面语句；

url	转发的新 URL 地址

10.5.2 例子 Example

CSP 转发可以包含参数，例如 Sample:

```
<?csp forward( ‘/index.csp?a=b’ ); ?>
或者；
<page:forward url="/index.csp" />
<page:forward url="/index.csp?a=b" />
```

10.6 重定向跳转 Location

10.6.1 格式 Format

```
<?csp location(url); ?>
或者；
<page:location url="" property="" />
```

重定向跳转操作；执行该操作，会自动退出该页面后面语句；

url	重定向的新 URL；该 URL 将显示中客户浏览器地址栏上；Location URL address。
property	重定向状态；301 ： 永久重定向；302 ： 临时重定向；默认不填为 301。

10.6.2 例子 Example

以下为重定向跳转例子 Sample：

```
<?csp location( ‘/index.csp’ ); ?>
或者；
<page:location url="http://www.google.com" />
<page:location url="/index.csp" />
```

第11章 HTTP/HTML

11.1 HTTP 自动验证 Authorization

11.1.1 格式 Format

```
<csp:authenticate>
```

自动请求浏览器，验证用户帐号信息；

11.1.2 例子 Example

Firefox 浏览弹出验证窗口如下：

CSP 后台通过 S$AuthAccount 和 S$AuthSecure 二个系统变量可以获得验证用户帐号信息：

11.2 HTTP 头信息 HTTP Header

用于获取浏览器相关 HTTP 协议头信息。

11.2.1 系统变量 System Variable

变量名称	数据类型	描述
S$HeadNames	vector	所有 HEADER 名称 All Header Name
S$HeadValues	vector	所有 HEADER 值 All Header Value
S$Heads	vector	所有 HEADER 名称值 格式 Format: HEADER_NAME: VALUE

11.2.2 自定义变量 Other Header Variable

格式 Format ： H$HEADER_NAME

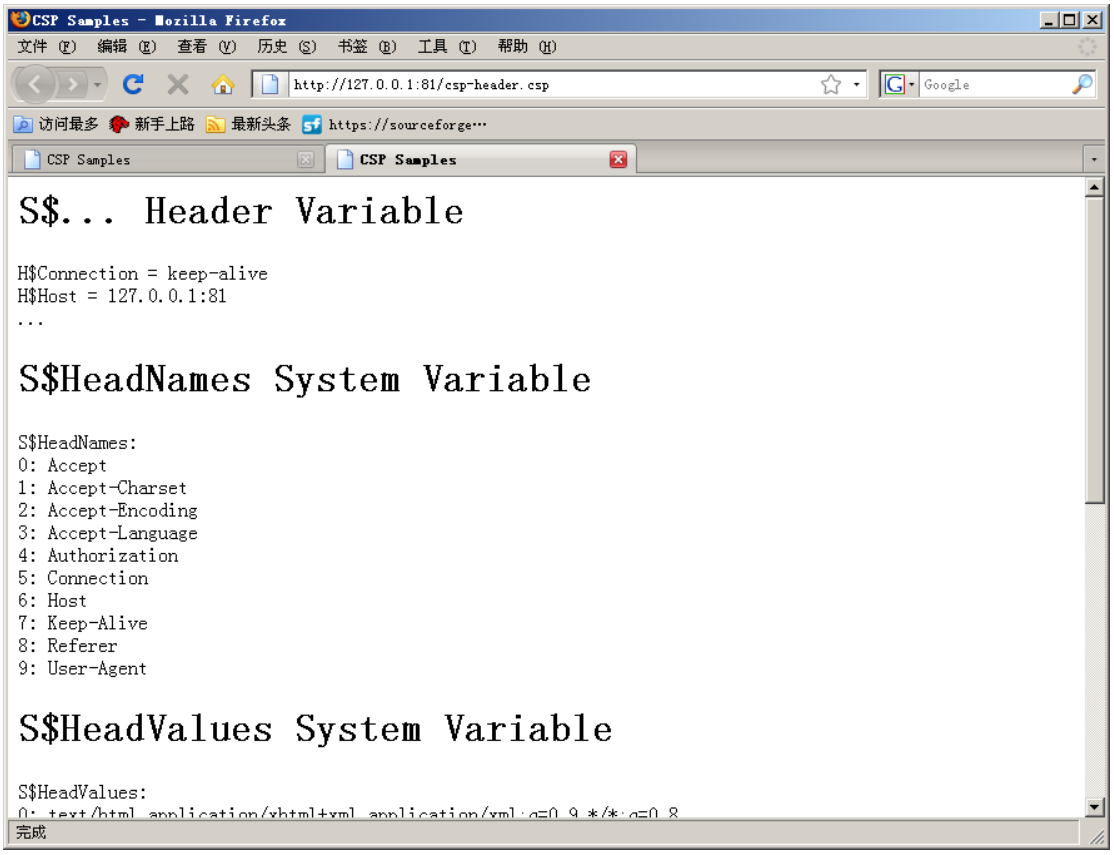
例子 Sample：

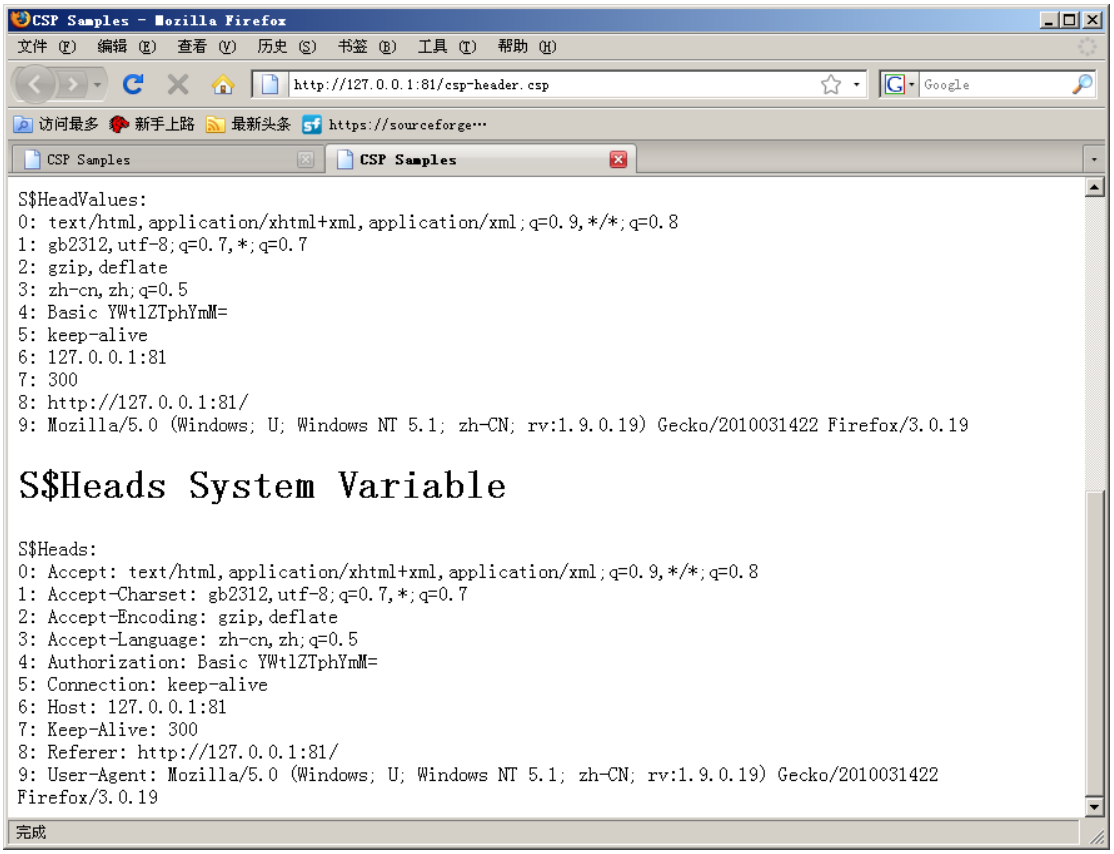
H$Connection

H$Content-Type

H$...

11.2.3 例子 Example





11.3 表单参数 Parameter

用于获取 HTML 表单参数信息。

11.3.1 系统变量 System Variable

变量名称	数据类型	描述
S$ParamNames	vector	所有参数名称；All Parameter Name
S$ParamValues	vector	所有参数值 All Parameter Value
S$Params	vector	所有参数名称值 格式 Format: PARAM_NAME=PARAM_VALUE

11.3.2 自定义变量 Other Parameter Variable

格式 Format ： P$PARAM_NAME

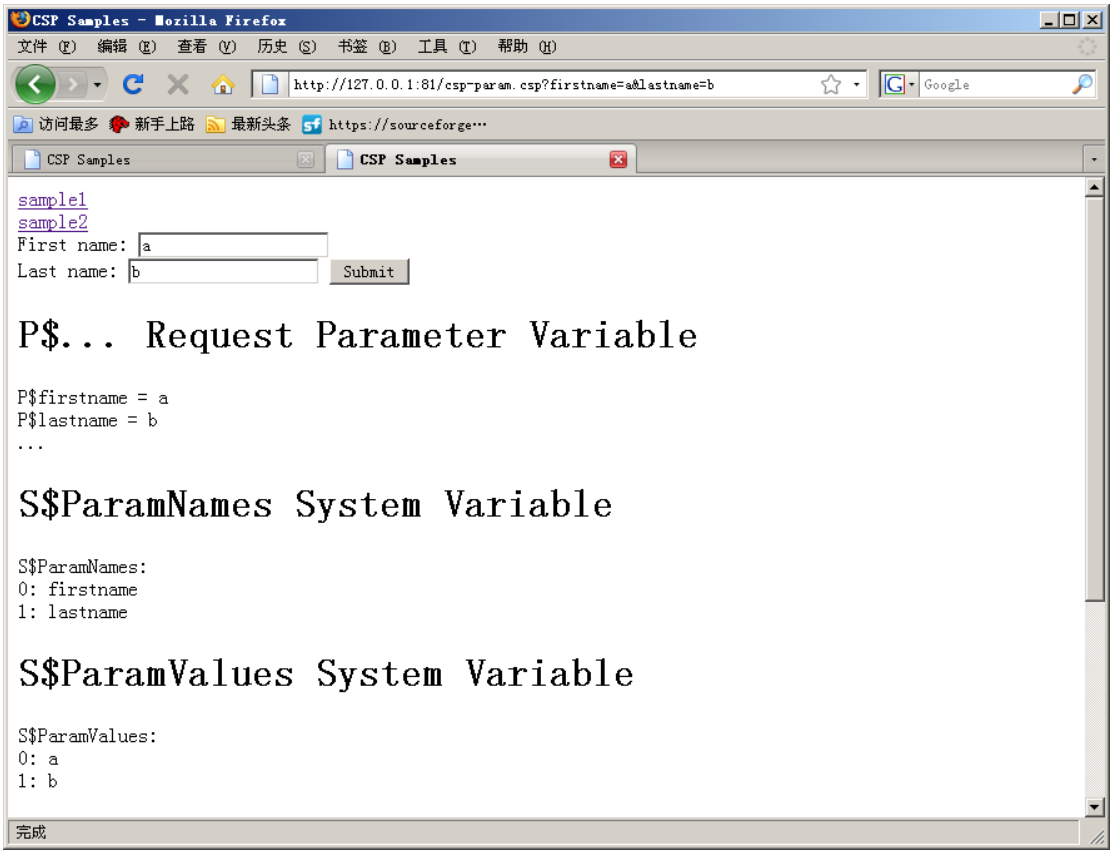
例子 Sample ：

P$User

P$FirstName

P$...

11.3.3 例子 Example



第12章　C++ APP

支持在 CSP 页面跟 C++ APP 应用组件进行交互，包括执行组件函数，调用服务接口方法，管理组件参数等等。

12.1 组件交互概述

可以通过配置 apps.xml 文件，不用在 CSP 页面定义和初始化，直接使用 C++ APP 应用组件；通过 A$Variable 进行访问。

通过 apps.xml 配置的应用组件变量，scopy 保存区域为 application；通过 define 语句定义的应用组件变量，可以自定义 scopy 保存区域。

通过 apps.xml 文件配置的应用组件变量，不必使用 final 语句注销退出。

12.1.1 执行应用组件函数

以下格式应用组件函数，可以在 CSP 页面中直接调用执行：

```
extern "C" HTTP_STATUSCODE CGC_API doServiceTest(const cgcHttpRequest::pointer & request, cgcHttpResponse::pointer response)
```

12.1.2 调用应用组件服务接口方法

可以在 CSP 页面中，调用 C++ APP 应用组件服务接口方法，服务接口例子：

```
class CAppService
: public cgcServiceInterface
```

12.1.3 管理应用组件服务接口参数

可以在 CSP 页面中，管理 C++ APP 应用组件服务接口参数，包括有 get set add del 等方法。

12.2 执行组件函数 Execute APP Function

12.2.1 格式 Format

```
<?csp
$var = exe_servlet(id, function);
?>
或者;
<csp:execute id="" function="" />
```

执行 id 组件的 function 方法;

12.2.2 例子 Example

```
<?csp
$var_status = exe_servlet("cspApp", "ServiceTest");
?>

<csp:execute id="cspApp" function="ServiceTest" />
```

12.3 调用组件服务接口方法 Call APP Function

12.3.1 定义组件实例 Define APP Variable

```
<csp:define type="app" id="" name="" scopy="" property="" />
```

定义一个 C++ APP 应用组件类型变量。

id	存放实例变量 APP Variable
name	组件名称 APP Name
scopy	(可选)生命周期，默认不填 page；optional
property	（可选）组件默认参数 optional

12.3.2 初始化组件 Init APP

```
<csp:app:init id="" in="" />
```

id	组件变量；（以下同） APP Variable
in	输入参数 Input Property
_$result	返回结果 Out Result, boolean

12.3.3 注销退出组件 Final APP

```
<csp:app:final id="" />
```

可选：退出组件，同时删除 id 变量； Opentional

12.3.4 调用组件服务接口方法 APP Call

```
<?csp
$out = app_call(id, name, in);
?>
或者；
<csp:app:call id="" name="" in="" out="" />
```

name	接口方法 Call Name
in	输入参数 Input Property
out	输出参数 Output Value

12.4 管理应用组件接口参数 APP Property

12.4.1 获取组件参数 Get APP Property

```
<?csp
$out = app_get(id, name);
?>
或者；
<csp:app:get id="" name="" out="" />
```

name	参数名称 Property Name
out	输出参数 Output Value

12.4.2 设置组件参数 Set APP Property

```
<?csp
app_set(id, name, in);
?>
或者;
<csp:app:set id="" name="" in="" />
```

name	参数名称 Property Name
in	输入参数 Input Property

12.4.3 添加组件参数 Add APP Property

```
<?csp
app_add(id, name, in);
?>
或者;
<csp:app:add id="" name="" in="" />
```

name	参数名称 Property Name
in	输入参数 Input Property

12.4.4 删除组件参数 Delete APP Property

```
<?csp
app_del(id, name);
?>
或者;
<csp:app:del id="" name="" />
```

name	参数名称 Property Name

12.4.5 获取组件信息 Get APP Info

```
<?csp
$out = app_info(id);
?>
或者;
<csp:app:info id="" out="" />
```

12.5 例子 Example

第13章 CDBC

CDBC（C/C++ Data Base Connectivity）用于连接数据源 DataSource，执行 SQL 语句。

13.1 CDBC 使用概述

可以通过配置 datasources.xml 文件，不必在 CSP 页面定义 CDBC 数据源，直接在 CSP 页面进行使用，调用 CDBC 数据源，执行 SQL 语句；通过 D$Variable 进行访问。

通过 datasources.xml 配置的 CDBC 数据源变量，scopy 保存区域为 application；通过 define 语句定义的 CDBC 数据源变量，可以自定义 scopy 保存区域。

通过 CDBC 标准接口可以连接所有标准关系型数据库系统或其他私有协议数据库，目前提供 BODB 和 MYSQL 二个连接数据库组件。

13.2 定义 CDBC 实例 Define CDBC Variable

```
<csp:define type="cdbc" id="" name="" scopy="" />
```

id	存放实例变量 CDBC Variable
name	CDBC DataSource, cdbcs.xml file
scopy	(可选)生命周期，默认不填 page

13.3 更新操作 Execute SQL

```
<?csp
$var = cdbc_exec(id, sql);
?>
或者;
<csp:cdbc:exec id="" sql="" />
```

sql	SQL 语句 INSERT,UPDATE,DELETE,…

13.4 查询操作 Select SQL

```
<?csp
$var = cdbc_select(id, sql);
?>
或者;
<csp:cdbc:select id="" sql="" out="" />
```

sql	SELECT SQL 语句
out	成功返回 RESULTSET COOKIE

13.5 获取第几条记录 Get Index

```
<?csp
$var = cdbc_move_index(id, in, index);
?>
或者;
<csp:cdbc:moveindex id="" in="" index="" />
```

in	RESULTSET COOKIE
index	第几条
_$result	返回指定记录，失败返回 null 数据；使用"csp:size"和"csp:foreach"等语句提取数据；（下同）

13.6 获取第一条记录 Get First

```
<?csp
$var = cdbc_move_first(id, in);
?>
或者;
<csp:cdbc:movefirst id="" in="" />
```

in	RESULTSET COOKIE
_$result	返回指定记录，失败返回 null 数据；使用"csp:size"和"csp:foreach"等语句提取数据；（下同）

13.7 获取下一条记录 Get Next

```
<?csp
$var = cdbc_move_next(id, in);
?>
或者;
<csp:cdbc:movenext id="" in="" />
```

13.8 获取上一条记录 Get Previous

```
<?csp
$var = cdbc_move_previous(id, in);
?>
或者;
<csp:cdbc:moveprev id="" in="" />
```

13.9 获取最后一条记录 Get Last

```
<?csp
$var = cdbc_move_last(id, in);
?>
或者;
<csp:cdbc:movelast id="" in="" />
```

13.10 清空记录集 Reset Result

```
<?csp
$var = cdbc_reset(id, in);
?>
或者;
<csp:cdbc:reset id="" in="" />
```

成功 SELECT 的记录，必须 reset 清空资源。

13.11 获取记录集大小 Result Size

```
<?csp
$var = cdbc_get_size(id, in);
```

```
?>
或者;
<csp:cdbc:getsize id="" in="" />
```

_$size	返回记录集大小

13.12 获取当前记录索引 Result Index

```
<?csp
$var = cdbc_get_index(id, in);
?>
或者;
<csp:cdbc:getindex id="" in="" />
```

_$index	返回当前记录集索引

13.13 例子 Example

第14章 CCP 语言

CCP 是一套简单的客户端页面语言，CCP 类似 JavaScript 语言一样，在客户端执行，CCP 由 MYC 开发平台支持服务。

可以利用 CCP 语言，帮助用户实现以下功能；

A、 定制客户端程序窗体属性，包括窗体大小、标题栏、最大化、系统托盘、界面右键菜单，滚动条等；

B、 定制客户端页面调用 C++组件；

C、 定制 HTML 页面元素的事件处理，支持从最简单的 onclick、ondblclick 到 onload、onkeyup 等所有标准 HTML 事件；例如实现点击页面按钮，执行本地一个 DLL 组件的某个接口等；

14.1 定制客户端程序窗体属性

```
<ccp id="window-size" autosize="0" enablemax="1" width="500" height="300" />
```

autosize	窗体是否随页面文档自动大小； 1：是 0：不是（后面 width 和 height 有效）
enablemax	窗体是否有最大化功能；1/0（以下同）
width	窗体宽，单位像素；autosize=1 时无效。
height	窗体高，单位像素；autosize=1 时无效。

```
<ccp id="window-setting" systoolbar="0" contextmenu="0" 3dborder="0" scrollbar="0" topmost="0" closingtip="" />
```

systoolbar	窗体是否有系统默认标题栏；1/0
contextmenu	窗体页面界面是否有右键菜单；1/0
3dborder	是否有页面边框；1/0
scrollbar	是否有页面滚动条；1/0

topmost	窗体是否在最上方；1/0
closingtip	窗体关闭前提示确认信息；

14.2 定制客户端页面调用 C++组件

14.2.1 功能与原理

利用 javascript:window.external.method 扩展函数来调用 C++组件，实现在客户端页面调用本地组件功能。

14.2.2 使用方法

配置：

```
<ccp:ext:method name="file_copyto" app="FileSystemService" method="copyto" />
<ccp:ext:method name="file_delete" app="FileSystemService" method="delete" />
```

使用：

```
<input type="button" onclick="javascript:window.external.file_copyto('D:/my_test.txt','D:/deletest_test2.txt');" value="复制文件测试" >
<input type="button" onclick="javascript:window.external.file_delete('D:/deletest_test2.txt');" value="删除文件测试" >
```

先配置一个扩展函数 file_copyto（file_delete）调用 FileSystemService 组件和 copyto（delete）方法；

配置后就可以利用 javascript:window.external.file_copyto(…)函数来调用，调用函数的参数支持所有的 javascript 基本数据库，参数数量和类型保持跟调用的组件函数一致即可。

详细例子可以参考 web/samples/win_test.csp 文件内容。

第15章　附录 A 系统变量 System Variable

变量名称	数据类型	描述
S$HeadNames	vector	所有 HEADER 名称 All Header Name
S$HeadValues	vector	所有 HEADER 值 All Header Value
S$Heads	vector	所有 HEADER 名称值 格式 Format: HEADER_NAME: VALUE
S$ParamNames	vector	所有参数名称；All Parameter Name
S$ParamValues	vector	所有参数值 All Parameter Value
S$Params	vector	所有参数名称值 格式 Format: PARAM_NAME=PARAM_VALUE
S$Scheme	string	
S$Method	int	
S$MethodString	string	
S$Protocol	string	
S$ContentLength	int	
S$ContentType	string	
S$RequestURL	string	
S$RequestURI	string	
S$QueryString	string	
S$RemoteIp	bigint	
S$RemoteAddr	string	
S$RemoteHost	string	
S$AuthAccount	string	

S$AuthSecure	string	
S$ServerName	string	
S$ServerPort	int	
S$ContextPath	string	
S$ServletName	string	
S$IsNewSession	boolean	
S$IsInvalidate	boolean	
S$SessionId	string	
S$SessionCreationTime	time	
S$SessionLastAccessed	time	
S$UploadFiles	vector	