資料編
result.py

```
#!python
# -*- coding: shift_jis -*-

import sys , re

DOCTYPE = '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">'

html = ""
css = ""
css2 = ''
classno = 0


######フォントサイズを変更するテーブル######
font_size_table = {
        "1":"x-small", "2":"small", "3":"medium", "4":"large", "5":"x-large", "6":"xx-large",
        "7":"150%", "-1":"small", "-2":"x-small", "-3":"xx-small", "+1":"large",
        "+2":"x-large", "+3":"xx-large"}

######HTML の属性名を CSS の属性名を変更するテーブル######
attr_table = {
        "bgcolor":"background-color",
        "background":"background-image",
        "bgproperties":"background-attachment",
        "topmargin":"margin-top",
        "bottommargin":"margin-bottom",
        "leftmargin":"margin-left",
        "rightmargin":"margin-right",
        "text":"color",
        "align":"text-align",
        "type":"list-style-type",
        "font-face":"font-family",
        "font-color":"color",
        "table-align":"float",
        "size":"font-size",
        "font-weight":"font-style",
        "table align":"table{float:",
        "ul style":"ul {list-style-type",
        "ul type":"ul {list-style-type",
        "li style":"li {list-style-type",
        "li type":"li {list-style-type",
        "ol style":"ol {list-style-type",
        "ol type":"ol {list-style-type",
        "middle":"center",
        "bordercolor":"border-color" ,
        }
```

```
re_file_name = re.compile(u'¥.html?$')
re_tag_lower = re.compile(r'<.+?>')
re_block_tag = re.compile(r'</(p|blockquote|ol|ul|dl|table|li|caption|dd|dt|head|td|tr|div|tbody)>')
re_block_tag2 = re.compile(r'<(hr|body|br).*?>')
re_spaces = re.compile(r' {2,}')
re_slash_tag = re.compile(u'</(b|i|u|s|strike|brink|strong)>')
re_endbody_tag = re.compile(u'</body>')
re_1attr = re.compile(u'<(h1|h2|h3|h4|h5|h6|tt|ul|li) ?(([A-Za-z]+) ?= ?"?(¥w+)"?)?>')
re_body = re.compile(u'<body (.+?)>')
re_many_attr = re.compile(u'<([0-9a-z]+) (.+?)>')
re_properties = re.compile(u'([a-z]+) ?= ?(¥"([^¥"]+)¥"|([^ ]+))')
re_charset = re.compile(u'charset=(.+?)"', re.I)


######ファイルの文字を全て小文字に変換する######
def tag_lower(m):
        return m.group(0).lower()


######属性名が一つの場合######
def one_attr(m):
        global css, classno
        tag = m.group(1)
        if m.group(2):
                if attr_table.has_key(m.group(3)):
                        attr = attr_table[m.group(3)]
                else:
                        attr = m.group(3)
                value = m.group(4)
                classno += 1
                newclass = tag + "_" + value + str(classno)
                css += tag + '.' + newclass + "¥n{" + attr + ":" + value + ";} ¥n¥n"
                return '<' + tag +  '  class="' + newclass + '">'
        else:
                return "<" + tag + ">"


######<body>に二つ以上の属性がある場合######
def body_each_property(m):
        global css, css2
        if m.group(1):
                attr = m.group(1)
                value = m.group(2)
                if value[0] == '"':
                        value = value[1:-1]
                if attr == 'background':
                        attr = 'background-image'
                        value = 'url(' + value + ')';
                if attr == 'link':
                        css2 += 'A:link { color : ' + value + ';}¥n'
```

```
                elif attr == 'alink':
                        css2 += 'A:active { color : ' + value + ';}¥n'
                elif attr == 'vlink':
                        css2 += 'A:visited { color : ' + value + ';}¥n'
                else:
                        if attr_table.has_key(attr):
                                attr = attr_table[attr]
                        css += attr + ' : ' + value + ';¥n'
        return ''


######<body>の変換######
def replace_body(m):
        global css, css2
        properties = m.group(1)
        css += 'body {¥n'
        properties = re_properties.sub(body_each_property, properties)
        css += '}¥n'
        if css2 != '':
                css += css2 + '¥n'
                css2 = ''
        return '<body>'


######属性名が二つ以上がある場合######
def each_property(m):
        global css
        attr = m.group(1)
        value = m.group(2)
        if value[0] == '"':
                value = value[1:-1]
                return ' ' + attr + '="' + value + '" '
        if attr_table.has_key(attr):
                attr = attr_table[attr]
        css += attr + ' : ' + value + ';¥n'
        return ''


######テーブルタグに属性名がある場合######
def table_each_property(m):
        global css
        attr = m.group(1)
        value = m.group(2)
        if value[0] == '"':
                value = value[1:-1]
        if attr == 'border':
                value = value + 'px'
                attr = 'border-width'
        elif attr == 'width':
                value = value + 'px'
```

```
	elif attr == 'background':
		attr = 'background-image'
		value = 'url(' + value + ')';
	elif attr == 'height':
		value = value + 'px'
	elif attr == 'align' and value == 'middle':
		attr = 'vertical-align'
	elif attr_table.has_key(attr):
		attr = attr_table[attr]
	css += attr + ' : ' + value + ';¥n'
	return ''


#######フォントタグに属性名がある場合######
def font_each_property(m):
	global css
	attr = m.group(1)
	value = m.group(2)
	if attr == 'style':
		return ' ' + attr + '=' + value + ' '
	if value[0] == '"':
		value = value[1:-1]
	if attr == 'size':
		attr = 'font-size'
		if font_size_table.has_key(value):
			value = font_size_table[value]
		else:
			value = "medium"
	elif attr_table.has_key(attr):
		attr = attr_table[attr]
	css += attr + ' : ' + value + ';¥n'
	return ''


######属性名が二つ以上あるタグの変換######
def replace_many_attr(m):
	global css, classno
	property = ''
	tag = m.group(1)
	properties = m.group(2)
	if 'class=' in properties:
		return '<' + tag + ' ' + properties + '>'
	if tag == 'a' or tag == 'img' or tag == 'br' or tag == 'div' or tag == 'input' or tag == 'form' or tag == 'script':
		return m.group(0)
	if tag == 'table':
		css += ""
table, tr, th, td {
	border: 1px solid black;
}
""
```

```
		classno +=1
		newclass = tag + "_" + str(classno)
		css += tag + '.' + newclass + ' {¥n'
		if tag == 'font':
			properties = properties.replace(u'MS ゴシック', 'serif')
			properties = re_properties.sub(font_each_property, properties)
		if tag == 'table' or tag == 'col' or tag == 'td' or tag == 'th':
			properties = re_properties.sub(table_each_property, properties)
		else:
			properties = re_properties.sub(each_property, properties)
		css += '}¥n¥n'
		return '<' + tag + ' class="' + newclass + '"' + property + '>'


############################################################
############################################################

input_file_name = sys.argv[1]
output_file_name = re_file_name.sub(".x.html", input_file_name)


n = 0
inhead = 0
head = ''
for line in open(input_file_name, 'rU'):
		line = line.rstrip()
		if line == '' and n == 0:
			continue
		n += 1
		if '<body' in line or '<BODY' in line:
			inhead = 0
		if n == 1:
			inhead = 1
			if '<!DOCTYPE' in line:
				line = DOCTYPE
			else:
				line = DOCTYPE + '¥n' + line
		elif inhead == 1:
			line = re_tag_lower.sub(tag_lower, line)

		if inhead == 1 and ('content-style-type' in line or 'stylesheet' in line):
			continue
		if inhead == 1 and '</head>' in line:
			line = '''<meta http-equiv="Content-Style-Type" content="text/css">
<link rel="stylesheet" type="text/css" href="style.css">
''' + line
		if inhead == 1:
			head += line + '¥n'
		else:
			if line == '':
```

```
                        html += '¥n'
                else:
                        html += line + ' '

m = re_charset.search(head)
if m:
        charset = m.group(1)
else:
        code = raw_input("文字コードを指定してください。番号を押して《Enter》を押してください。¥n"
"1.Shift_JIS ¥n2.euc-jp ¥n3.iso-2022-jp ¥n0.終了¥n")
        if code == "1":
                charset = 'Shift_JIS'
        elif code == "2":
                charset = 'euc-jp'
        elif code == "3":
                charset = 'iso-2022-jp'
        else:
                sys.exit(0)

head = unicode(head, charset)
html = unicode(html, charset)

html = re_tag_lower.sub(tag_lower, html)
html = re_body.sub(replace_body, html)

if '<b>' in html:
        html = html.replace('<b>', '<span class="bold">')
        css += 'span.bold { font-weight: bold }¥n¥n'
if '<i>' in html:
        html = html.replace('<i>', '<span class="italic">')
        css += 'span.italic { font-style: italic }¥n¥n'
if '<u>' in html:
        html = html.replace('<u>', '<span class="underline">')
        css += 'span.underline { text-decoration: underline }¥n¥n'
if '<s>' in html:
        html = html.replace('<s>', '<span class="strike">')
        css += 'span.strike { text-decoration: line-though }¥n¥n'
if '<strike>' in html:
        html = html.replace('<strike>', '<span class="strike-through">')
        css += 'span.strike-through { text-decoration: line-through }¥n¥n'
if '<brink>' in html:
        html = html.replace('<brink>', '<span class="brink">')
        css += 'span.brink { text-decoration: brink }¥n¥n'
if '<strong>' in html:
        html = html.replace('<strong>','<span class="bold">')
        css += 'span.bold { font-weight: bold }¥n¥n'
if '<center>' in html:
        html = html.replace('<center>','<div align="center">')
        css += 'div{text align: center }¥n¥n'
        html = html.replace('</center>', '</div>')
```

```
html = re_slash_tag.sub('</span>', html)
html = re_endbody_tag.sub('</style>', html)
html = re_block_tag.sub('¥g<0>¥n', html)
html = re_block_tag2.sub('¥g<0>¥n', html)
html = re_spaces.sub(' ', html)
html = re_1attr.sub(one_attr, html)


html = re_many_attr.sub(replace_many_attr, html)



css_file = open('style.css', 'w')
css = css.encode(charset)
css_file.write(css)
css_file.close()


head = head.encode(charset)
html = html.encode(charset)
output_file_name = open('style.html','w')
output_file_name.write(head + html)
output_file_name.close()

sys.exit()
```

資料編

main.py

```
#!python
# -*- coding: euc_jp -*-

import sys , re

DOCTYPE = '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">'

html = ""
css = ""
css2 = ''
classno = 0
char_set = ""
border = ""


######フォントサイズを変更するテーブル######
font_size_table = {
        "1":"x-small", "2":"small", "3":"medium", "4":"large", "5":"x-large", "6":"xx-large",
        "7":"150%", "-1":"small", "-2":"x-small", "-3":"xx-small", "+1":"large",
        "+2":"x-large", "+3":"xx-large"}


######HTML の属性名を CSS の属性名に変更するテーブル######
attr_table = {
        "bgcolor":"background-color",
        "background":"background-image",
        "bgproperties":"background-attachment",
        "topmargin":"margin-top",
        "bottommargin":"margin-bottom",
        "leftmargin":"margin-left",
        "rightmargin":"margin-right",
        "text":"color",
        "align":"text-align",
        "type":"list-style-type",
        "face":"font-family",
        "font-color":"color",
        "table align":"float",
        "size":"font-size",
        "font-weight":"font-style",
        "table align":"table{float:",
        "caption align":"caption {caption-side",
        "ul style":"ul {list-style-type",
        "ul type":"ul {list-style-type",
        "li style":"li {list-style-type",
        "li type":"li {list-style-type",
        "ol style":"ol {list-style-type",
        "ol type":"ol {list-style-type",
        "middle":"center",
        "bordercolor":"border-color"
```

```
	}


#####正規表現をコンパイルして変数に代入#####

re_file_name = re.compile(u'¥.html?$')
re_tag_lower = re.compile(u'<.+?>')
re_block_tag = re.compile(u'</(p|blockquote|ol|ul|dl|table|li|caption|dd|dt|head|td|tr|div|tbody)>')
re_block_tag2 = re.compile(u'<(hr|body|br).*?>')
re_spaces = re.compile(u' {2,}')
re_slash_tag = re.compile(u'</(b|i|u|s|strike|brink|strong)>')
re_endbody_tag = re.compile(u'</body>')
re_body = re.compile(u'<body (.+?)>')
re_many_attr = re.compile(u'<([0-9a-z]+) (.+?)>')
re_properties = re.compile(u'([a-z]+) ?= ?(¥"([^¥"]+)¥"|([^ ]+))')
re_charset = re.compile(u'charset=(.+?)"', re.I)


######ファイルの文字を全て小文字に変換する######
def tag_lower(m):
	return m.group(0).lower()


######body 要素に二つ以上の属性がある場合######
def body_each_property(m):
	global css, css2
	if m.group(1):
		attr = m.group(1)
		value = m.group(2)
		if value[0] == '"':
			value = value[1:-1]
		if attr == 'background':
			attr = 'background-image'
			value = 'url(' + value + ')';
		if attr == 'link':
			css2 += 'A:link { color : ' + value + ';}¥n'
		elif attr == 'alink':
			css2 += 'A:active { color : ' + value + ';}¥n'
		elif attr == 'vlink':
			css2 += 'A:visited { color : ' + value + ';}¥n'
		else:
			if attr_table.has_key(attr):
				attr = attr_table[attr]
			css += attr + ' : ' + value + ';¥n'
	return ''


######body 要素の変換######
def replace_body(m):
	global css, css2
```

```
	properties = m.group(1)
	css += 'body {¥n'
	properties = re_properties.sub(body_each_property, properties)
	css += '}¥n'
	if css2 != '':
		css += css2 + '¥n'
		css2 = ''
	return '<body>'


######属性名が二つ以上がある場合######
def each_property(m):
	global css
	attr = m.group(1)
	value = m.group(2)
	if value[0] == '"':
		value = value[1:-1]
	if attr_table.has_key(attr):
		print "table.has_key"
		attr = attr_table[attr]
		css += attr + ' : ' + value + ';¥n'
		return ''
	else:
		return ' ' + attr + '="' + value + '"'


######table 要素に属性名がある場合######
def table_each_property(m):
	global css, border
	attr = m.group(1)
	value = m.group(2)
	if value[0] == '"':
		value = value[1:-1]
	if attr == 'border':
		attr = 'border'
		border = value
		value = value + 'px solid #000000'
	elif attr == 'width':
		if '%' not in value:
			value = value + 'px'
	elif attr == 'background':
		attr = 'background-image'
		value = 'url(' + value + ')';
	elif attr == 'height':
		value = value + 'px'
	elif attr == 'align' and value == 'middle':
		attr = 'vertical-align'
	elif attr == 'align':
		return ' align="' + value + '"'
	elif attr_table.has_key(attr):
```

```
                attr = attr_table[attr]
        else:
                return ' ' + attr + '=″' + value + '″'
        css += attr + ' : ' + value + ';¥n'
        return ''


#######font 要素に属性名がある場合######
def font_each_property(m):
        global css
        attr = m.group(1)
        value = m.group(2)
        if attr == 'style':
                return ' ' + attr + '=' + value + ' '
        if value[0] == '″':
                value = value[1:-1]
        if attr == 'size':
                attr = 'font-size'
                if font_size_table.has_key(value):
                        value = font_size_table[value]
                else:
                        value = ″medium″
        elif attr_table.has_key(attr):
                attr = attr_table[attr]
        css += attr + ' : ' + value + ';¥n'
        return ''


######属性名が二つ以上あるタグ要素の変換######
def replace_many_attr(m):
        global css, classno, border
        property = ''
        tag = m.group(1)
        properties = m.group(2)
        if 'class=' in properties:
                return '<' + tag + ' ' + properties + '>'
        if tag == 'a' or tag == 'img' or tag == 'br' or tag == 'div' or tag == 'input' or tag == 'form' or tag == 'script':
                return m.group(0)
        classno +=1
        newclass = tag + ″_″ + str(classno)
        css += tag + '.' + newclass + ' {¥n'
        if tag == 'font':
                properties = properties.replace(u'MS ゴシック', 'serif')
                properties = re_properties.sub(font_each_property, properties)
        if tag == 'table' or 'col' or tag == 'td' or tag == 'tr' or tag == 'th':
                properties = re_properties.sub(table_each_property, properties)
        else:
                properties = re_properties.sub(each_property, properties)
        css += '}¥n¥n'
        if tag == 'table' and border != ″″:
```

```
                newdivclass = "div" + str(classno)
                css += 'div.div' + str(classno) + ' tr,td,th {¥n¥tborder:1px solid #000000;¥n'
                border = ""
                css += 'border-collapse:collapse;}¥n¥n'
                return '<div class="div' + str(classno) + '">¥n<' + tag + ' class="' + newclass + '"' + properties + '>'
        else:
                return '<' + tag + ' class="' + newclass + '"' + properties + '>'


############################################################
############################################################


re_newline = re.compile(r'[¥n¥r]{1,2}')

def main(data, charset):
        global html, css, char_set
        char_set = charset;
        data = unicode(data, char_set)
        lines = re_newline.split(data)
        n = 0
        inhead = 0
        head = ''
        html = ''
        for line in lines:
                line = line.rstrip()
                if line == '' and n == 0:
                        continue
                n += 1
                if '<body' in line or '<BODY' in line:
                        inhead = 0
                if n == 1:
                        inhead = 1
                        if '<!DOCTYPE' in line:
                                line = DOCTYPE
                        else:
                                line = DOCTYPE + '¥n' + line
                elif inhead == 1:
                        line = re_tag_lower.sub(tag_lower, line)

                if inhead == 1 and ('content-style-type' in line or 'stylesheet' in line):
                        continue
                if inhead == 1 and '</head>' in line:
                        line = '<meta http-equiv="Content-Style-Type" content="text/css">'
                        line += '¥n<style type="text/css">¥n<!--¥n__css__¥n--></style>¥n</head>' + line
                if inhead == 1:
                        head += line + '¥n'
                else:
                        if line == '':
                                html += '¥n'
```

```
                else:
                        html += line + ' '
```

###文字コードの変換###

```
        head = re_charset.sub('charset=' + char_set + '"', head)
        html = re_tag_lower.sub(tag_lower, html)
        html = re_body.sub(replace_body, html)
```

###HTML と CSS でタグ名が変わるものの変換###

```
        if '<b>' in html:
                html = html.replace('<b>', '<span class="bold">')
                css += 'span.bold { font-weight: bold }¥n¥n'
        if '<i>' in html:
                html = html.replace('<i>', '<span class="italic">')
                css += 'span.italic { font-style: italic }¥n¥n'
        if '<u>' in html:
                html = html.replace('<u>', '<span class="underline">')
                css += 'span.underline { text-decoration: underline }¥n¥n'
        if '<s>' in html:
                html = html.replace('<s>', '<span class="strike">')
                css += 'span.strike { text-decoration: line-through }¥n¥n'
        if '<strike>' in html:
                html = html.replace('<strike>', '<span class="strike-through">')
                css += 'span.strike-through { text-decoration: line-through }¥n¥n'
        if '<brink>' in html:
                html = html.replace('<brink>', '<span class="brink">')
                css += 'span.brink { text-decoration: brink }¥n¥n'
        if '<strong>' in html:
                html = html.replace('<strong>','<span class="bold">')
                css += 'span.bold { font-weight: bold }¥n¥n'
        if '<center>' in html:
                html = html.replace('<center>',' ')
                html = html.replace('</center>', ' ')
        if '</table>' in html:
                html = html.replace('</table>', '</div>¥n</table>')
```

###正規表現で置換###

```
        html = re_slash_tag.sub('</span>', html)
        html = re_endbody_tag.sub('</style>', html)
        html = re_block_tag.sub('¥g<0>¥n', html)
        html = re_block_tag2.sub('¥g<0>¥n', html)
        html = re_spaces.sub(' ', html)
        html = re_many_attr.sub(replace_many_attr, html)
```

###HTML と CSS を書き出す###

```
        css = css.encode(char_set)
```

```
        head = head.encode(char_set)
        html = html.encode(char_set)
        head = head.replace("__css__", css)
        return head + html

if __name__ == "__main__":
        input_file = open(sys.argv[1])
        data = input_file.read()
        print main(data, "euc_jp")
```

資料編
change.py

```
#!/usr/bin/env python
# -*- coding: euc_jp -*-

import cgi, cgitb, main

print "Content-Type: text/html"
print

cgitb.enable()
form_data = cgi.FieldStorage()
data = form_data.getfirst("Data", "")

print main.main(data, "euc-jp")
```

資料編

index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<title>HTML ファイルのオンライン変換ツール</title>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-jp">
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<style type="text/css">
<!--
body {
	font-size: 95%;
	font-family: Tahoma, Arial, Helvetica, sans-serif;
	background: #fafafa;
}
div#pagetop {
	width: 800px;
	margin: 0 auto;
	border: 1px solid #003973;
}
div#header {
	border-top: 5px solid #003973;
	border-bottom: 1px solid #7fb3b3;
	background: #7fb3b3;
}
h1 {
	text-align: center;
	font-size: 160%;
	letter-spacing: 0.2em;
	color: #fafafa;
}
div#kiji {
	border: solid 4px #7fb3b3;
	background: #fafafa;
	font-size: midium;
	text-align: left;
	padding: 1em 15px;
}
p.kiji {
	color: #ff0000;
}
p#kiji {
	margin-left: 10px;
}
pre.pre{
	margin-left: 20px;
}
p.pre {
	color: #dc143c;
}
table {
```

```
		border-collapse:collapse;
		border-spacing:0;
		margin-left: 20px;
}
div#henkan {
		margin-right: 100px;
		text-align: right;
}
div#footer {
		background: #7fb3b3;
		border-top: 1px solid #7fb3b3;
		border-bottom: 5px solid #003973;
		font-size: small;
		text-align: center;
		padding: 1px;
}
-->
</style>
</head>


<body>
<div id="pagetop">

<div id="header"><h1>HTML ファイルのオンライン変換</h1></div>

<div id="kiji">
<p>本ツールはデザインタグを含んだ HTML 文書を HTML 部分とスタイルシートに分離させるツールです。<br>
下記テキストエリアに変換させたい HTML 文書を記述してください。<br>
変換後は、変換された後のページがプレビューされます。<br>
また、変換する前に HTML の文法チェックを行うことをお勧めします。</p>
<p class="kiji">※注意※</p>
<p>文字コードは自動的に euc-jp が適用されます。<br>
ブラウザが Internet Explorer の場合、ソースを正しく表示することができません。<br>
<br>
スタイルシートは&lt;head&gt;要素内に以下のように表示されます。</p>
<pre class="pre">
&lt;style type="text/css"&gt;
&lt;!- -
<p class="pre">ここにスタイルシートが書き込まれます。</p>
- -&gt;
&lt;/style&gt;
</pre>
<br>
<hr>

<!-ここから変換->
<form action="/cgi-bin/change.cgi" method="POST" enctype="multipart/form-data">
<input type="hidden" name="Stat" value="on">
<table>
```

```
<!-HTML 文書を記述して変換//->
<tr><td width="150" valign="top">HTML 文書</td>
<td><textarea name="Data" cols="55" rows="20"></textarea></td></tr>
</table>

<br>

<!-変換実行！//->
<div id="henkan">
<input type="submit" value="変換"> <input type="reset"  value="リセット">
</div>
</form>

</div>

<div id="footer">
&#xA9; 2007 Taira Megumi @ Fukuda zeminer 2007
</div>

</div>

</body>
</html>
```

資料編
readme.txt

==========

==== HTML ファイルのオンライン変換ツールについて ====

====

====

==== (C)2007 Taira Megumi @ Fukuda zeminer2007

====

==========

******* はじめに **********

この Web サイトはデザイン要素(<font color="#ff0000">や<b>など)を含む HTML ファイルを
HTML 部分とスタイルシート部分に分離させて再構成した HTML ファイルに変換するツールである。

******* 資料構成 **********

main.py	変換ファイル
change.cgi	変換ファイル
index.html	本体ファイル
sample.html	サンプルファイル
readme.txt	このファイル

******* 使用方法 **********

1. "index.html"とは別に"main.py"と"change.cgi"を CGI を設置するディレクトリに移動する。
2. デザイン要素を含んでいる HTML 文書を書き込み、変換ボタンを押す。
   (見つけられない場合には資料の CD-R 内に"sample.html"がある)
3. 変換された後のページがプレビュー表示される。
4. ソースを表示してもらい、文書の構成とデザイン要素を書き出したものに変換されているか確認する。
5. テキストエディタ(サクラエディタ、秀丸、emacs など)にコピーし、保存する。
6. スタイルシート部分については変換されていないタグや属性名を追加し、反映されるか確認する。

※正しい変換結果を出すためにも変換前に HTML 文書の文法チェックを行うことをお勧めします。
参考 Web サイト http://openlab.ring.gr.jp/k16/htmllint/htmllint.html

******* 注意事項 **********

ブラウザによって変換後のページに差異が起こることがあります。

本ツールは UNIX 環境下で作成したため、文字コードが euc-jp となります。
変換後、保存の際には指定した文字コードに指定し直してください。

ブラウザが Internet Explorer では変換後のページ確認はできますが
文字コードの関係上、ソースが文字化けします。Firefox を推奨します。