

Beginning Perl Programming (X52.9543.001)**Handout 9: July 20, 2000*****Some Elegant Perl Snippets****Self-print-out option*

```
if ($option) {
    open(ME, "<$0") || die "Can't Open $0";
    while (<ME>) {printf ("%d: %s", ++$count, $_);}
}
```

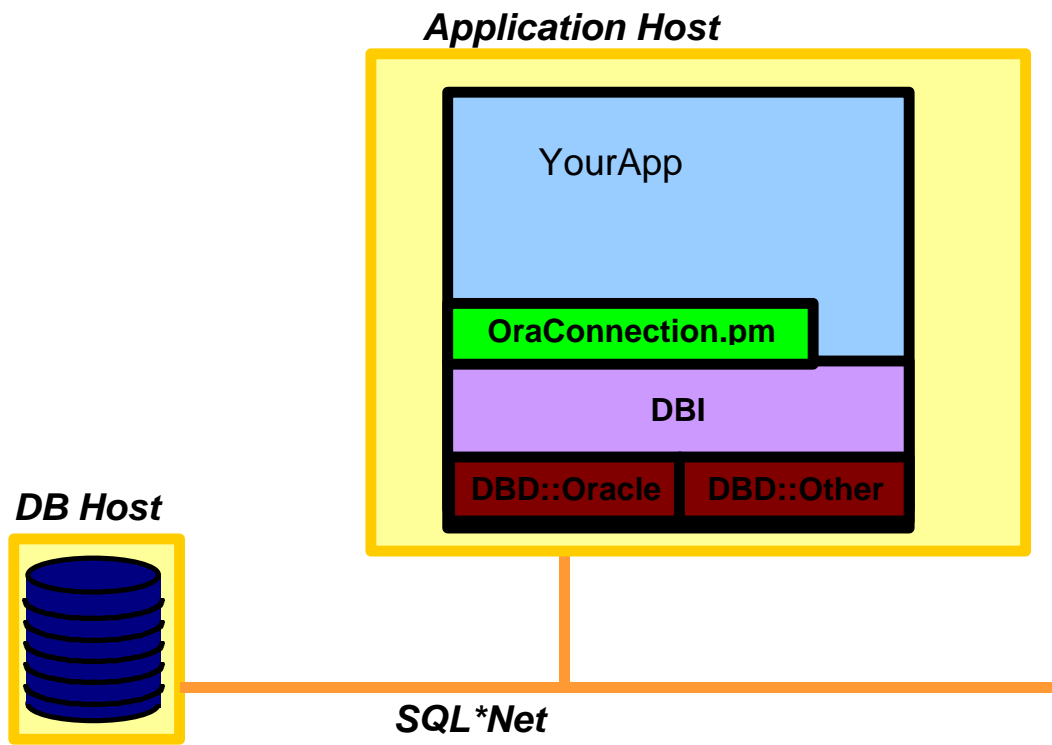
Code references as a jump-table based on URL options:

```
%states = {
    'Default'   =>    \&frontpage,
    'Shirt'    =>    \&shirt,
    'Sweater'  =>    \&sweater,
    'Checkout' =>    \&checkout,
    'Card'     =>    \&credit_card,
    'Order'    =>    \&order,
    'Cancel'   =>    \&frontpage
};

...

if ($states{$URLStateParam}) {
    &{$states{$URLparamState}} (param1, ..., paramN);
} else {
    &unknownURLStateParam();
}
```

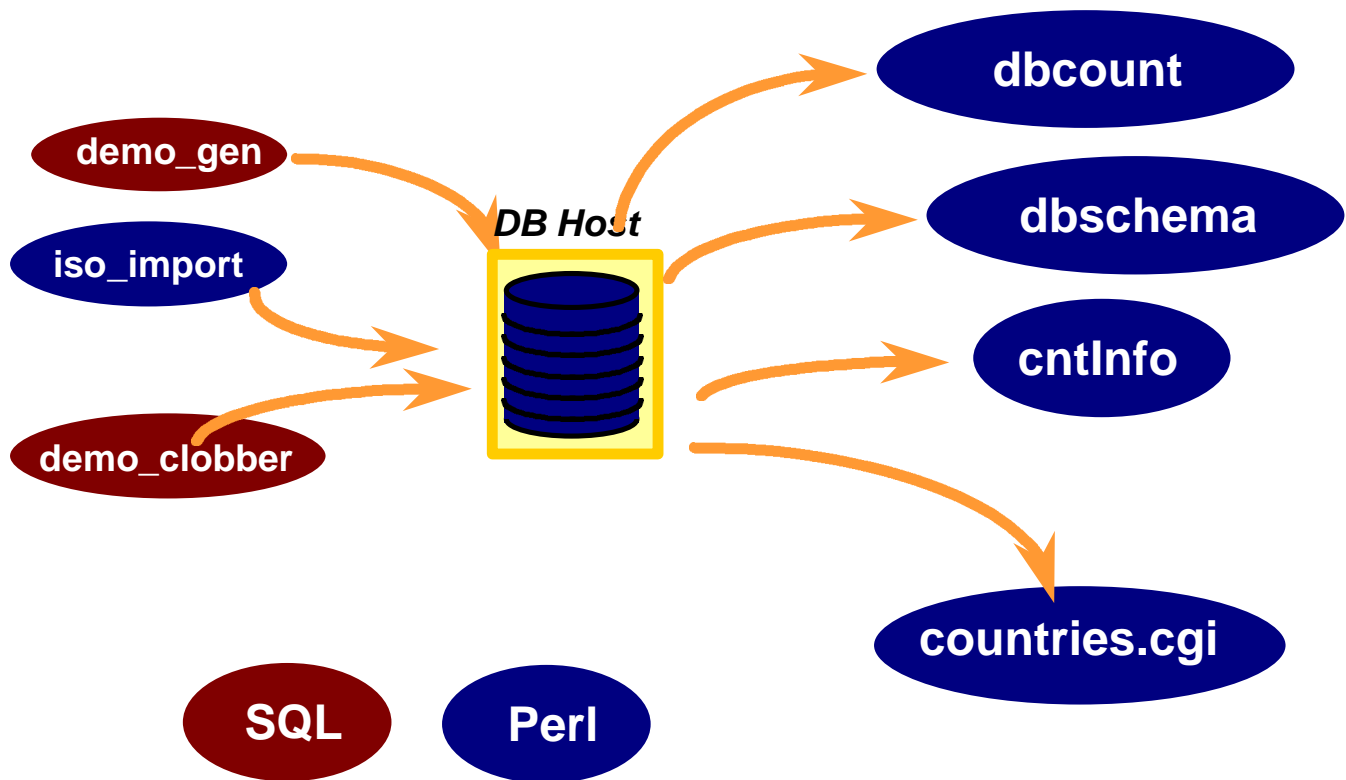
Perl5 DBI / DBD / OraConnection Architecture



OraConnection Interface

- On top of DBI
- Object represents a single connection
- Simplifies error handling
- Provides methods for many things which are “messy” in SQL.
- Facilitates one-time execution.

Demo Scripts



The following listings have been **heavily edited** to remove much of the boilerplate code and leave areas of code relevant for discussions. For the full files, see the class web site at <http://www.goss.com/perl>.

demo_clobber

```
#!/bin/sh
# demo_clobber - clobber the schema for the demo database

...

sqlplus -s $USER/$PSWD@$TNS <<EOF_SCRIPT

...

drop table COUNTRIES          cascade constraints;
drop table STATES_PROVS      cascade constraints;

EOF_SCRIPT
```

demo_gen

```
#!/bin/sh
# demo_gen - generate the schema for the demo database
...

sqlplus -s $USER/$PSWD@$TNS <<EOF_SCRIPT

...

prompt Creating Table: COUNTRIES
prompt -----

REM This table holds a copy of the information from the
REM ISO-3166 country-codes standard.
REM
REM Note, the input tbl was 'hacked' to include:
REM
REM   SCO   SCOTLAND
REM   ZZZ   FORMER SOVIET UNION WITHOUT RUSSIA

REM ISO-3166 country-codes

create table COUNTRIES (
    cnt_code_a2          char (2)          not null      ,
    cnt_code_a3          char (3)          not null      ,
    cnt_code_number      number (3, 0)     not null      ,
    cnt_name_upper_case  varchar2 (48)     not null      ,
    cnt_name_mixed_case  varchar2 (48)     not null
)
    pctfree 10 pctused 50
    storage (initial 15K next 10K pctincrease 20);

alter table COUNTRIES
    add constraint CNT_PK
        primary key (cnt_code_a3);

comment on table COUNTRIES is
'Table of world countries listed in the ISO-3166 standard.';

comment on column COUNTRIES.cnt_code_a2 is
'ISO 3166 country digraph (2-character country code).';

...

create table STATES_PROVS (
    sta_code_a2          char (2)          not null      ,
    sta_cnt_code_a3      char (3)          not null      ,
    sta_name_upper_case  varchar2 (48)     not null      ,
    sta_name_mixed_case  varchar2 (48)     not null
)
    pctfree 10 pctused 50
    storage (initial 15K next 10K pctincrease 20);

EOF_SCRIPT
```

iso_3166.txt

Updated by the RIPE Network Coordination Centre.

From: ftp://ftp.ripe.net/iso3166-countrycodes

Source: ISO 3166 Maintenance Agency

Latest change: Thu Aug 7 17:59:51 MET DST 1997

Country	A 2	A 3	Number
AFGHANISTAN	AF	AFG	004
ALBANIA	AL	ALB	008
ALGERIA	DZ	DZA	012
AMERICAN SAMOA	AS	ASM	016
...			
UNITED STATES	US	USA	840
UNITED STATES MINOR OUTLYING ISLANDS	UM	UMI	581
URUGUAY	UY	URY	858
UZBEKISTAN	UZ	UZB	860
VANUATU	VU	VUT	548
VENEZUELA	VE	VEN	862
VIET NAM	VN	VNM	704
VIRGIN ISLANDS (BRITISH)	VG	VGB	092
VIRGIN ISLANDS (U.S.)	VI	VIR	850
WALLIS AND FUTUNA ISLANDS	WF	WLF	876
WESTERN SAHARA	EH	ESH	732
YEMEN	YE	YEM	887
YUGOSLAVIA	YU	YUG	891
ZAMBIA	ZM	ZMB	894
ZIMBABWE	ZW	ZWE	716

List of changes applied, as specified in registration newsletters:

Newsletter III-1, 1989-12-5:

Burma deleted, Myanmar added (same numeric value, change of country name)

... remainder of file contains update history - deleted by Clint Goss

iso_import

```
#!/perl
# iso_import -- Import ISO standard information - countries
#
#   Written by Clint Goss <clint@goss.com>, November 1998

BEGIN {
    # Access Perl modules
    #push (@INC, ...your lib directory here...);
    # Point to our preferred installation of Oracle
    #ENV{'ORACLE_HOME'} = ...your Oracle Home here...;
}

# Perl Modules
```

```

use OraConnection;      # Connection to Oracle, incl DBI and DBD::Oracle
use FileHandle;
use strict;            # Restrict unsafe variables, refs, barewords

# Set up the connection to Oracle

my ($database, $username, $password) = qw (ora9 demo demo);
my ($dbc) = new OraConnection;
$dbc->connect ($database, $username, $password);

# Statistics

my ($countCntIns) = 0;

&importRef ("iso_3166.txt");

print "Inserted $countCntIns into Countries table.\n";

exit;

# Import one ISO Country file containing Reference table information

sub importRef {
    my ($fileName) = @_;

    open (ISO_LOG, "<" . $fileName) || die "Can't open " . $fileName;

    # Flag if we have found the "-----" line which marks the
    # start of the country data.

    my ($foundHeader);

    my ($line);
    while ($line = <ISO_LOG>) {

        # "Just tidy us up a bit"

        chomp $line;
        $line =~ s/^\s*//;
        $line =~ s/\s*$//;

        # Check if this is the header line

        if (! $foundHeader) {
            if ($line =~ /^-----/) { $foundHeader = 1; }
            next;
        }

        # Here if we've already found header
        # Check if we're at the end of the data

        if ($line =~ /^-----/) { last; }
        if ($line eq "") { next; }

        # Here if we got a real line of country data.

# The format is (for example):
#      1      2      3      4      5      6      7
# 1234567890123456789012345678901234567890123456789012345678901234567890

```

```

my ($uName) = substr ($line, 0, 48);
$uName =~ s/^\s*//;
$uName =~ s/\s*$//;

my ($remainder) = substr ($line, 48);

$remainder =~ s/^\s*//;
$remainder =~ s/\s*$//;

my ($a2, $a3, $number) = split (/\\s+/, $remainder, 3);

# Convert the country name to upper lower case

my ($ulName) = ucfirst lc $uName;

# Convert the first character of the second name
# of two-part and hyphenated names to upper case.

$ulName =~ s/([ -\\])([a-z])/\\1.uc($2)/ge;

# Restore English-language articles back to lower case

$ulName =~ s/ Of / of /g;
$ulName =~ s/ Of\\)/ of)/g;
$ulName =~ s/ Of$/ of/;

$ulName =~ s/ And / and /g;
$ulName =~ s/ And$/ and/;

$ulName =~ s/ The / the /g;
$ulName =~ s/ The$/ the/;

# Restore Word'S back to Word's

$ulName =~ s/'S /'s /g;

# Debugging:
# print "Country $number $a2 $a3 $ulName\\n";

# Insert into COUNTRIES table

my ($stmt) = "INSERT into COUNTRIES (" .
    "cnt_code_a2, " .
    "cnt_code_a3, " .
    "cnt_code_number, " .
    "cnt_name_upper_case, " .
    "cnt_name_mixed_case) " .
    "VALUES (" .
    $dbc->quote ($a2) . ", " .
    $dbc->quote ($a3) . ", " .
    $dbc->quote ($number) . ", " .
    $dbc->quote ($uName) . ", " .
    $dbc->quote ($ulName) . ")";

$dbc->exec ($stmt);
$countCntIns++;
}
}

```

Extract from the OraConnection Man Page

OraConnection(3) User Contributed Perl Documentation OraConnection(3)

NAME

OraConnection - Connection to an Oracle Database

SYNOPSIS

```
use OraConnection;

$db = new OraConnection;
$db->connect ($databaseName, $username, $password,
             $optionalOptimizerGoal, $optionalDateFormat);

$tableNames = $db->tableNames ();
$viewNames = $db->viewNames ();
$tableOrViewNames = $db->tableOrViewNames ();
$dbLinkNames = $db->dbLinkNames ();
$sequenceNames = $db->sequenceNames ();

$colNames = $db->columnNames ("MY_TABLE");

$quotedValue = $db->quote ("Don't shoot the piano player");

# Repetitive execution

$insertStmt = "INSERT INTO TBL (who, when) VALUES (" .
             db->quote ("Clint") . ", SYSDATE)";

# First execution, fetching statement handle for other exec's
$sth = $db->execMulti ($insertStmt);

# Second execution
$db->reExec ($sth);

# You MUST call finish to free the statement handle
$db->finish ($sth);

# One-time execution - no finish() required or allowed
$db->exec ($insertStmt);

$query = "SELECT height, weight from INFO " .
        db->quote ("Clint");

# Fetch EXACTLY one result row. Fails if 0 or multiple rows
$resultRow = $db->fetchSingle ($query);
($height, $weight) = @$resultRow;

# Direct fetch of height
$height = $db->get ("INFO", "height", "name", "Clint");

# Fetch all result rows.
($resultRows, $rowCount) = $db->fetchAll ($query);
foreach $resultRow (@$resultRows) {
    ($height, $weight) = @$resultRow;
}

# Count of all rows and rows matching various
# column values
```

```
$rowCount = $db->rowCount      ("INFO");
$rowCount = $db->matchingRowCount  ("INFO", "height", "72", "weight", "195");

#      Sequence number  values
$sequenceVal = $db->currval ($sequenceName);
$sequenceVal = $db->nextval ($sequenceName);
$minValue = $db->minValue ($sequenceName);
$maxValue = $db->maxValue ($sequenceName);
$incrementBy = $db->incrementBy ($sequenceName);

# Not required - done automatically if $db goes "out of scope"
$db->disconnect ();
```

DESCRIPTION

...

Case Sensitivity Issues

...

RDB Support

...

Support for LONG and LONG RAW

...

FUNCTIONS

...

WARNING

The `fetchAll()` method can potentially return a huge amount of data.

NOTES

This module has not yet been set up for "installation" in your local Perl.

BUGS

The `rowCountOfQuery()` method does not properly handle complex SQL SELECT statements. If more than one "FROM" token appears in the query, (eg. in a subquery subordinated in a WHERE clause), then this routine will NOT work properly - it will mangle the query and cause an SQL parse error.

EXAMPLES

...

SEE ALSO

DBI and DBD::Oracle man pages.

AUTHOR

Clint Goss <clint@goss.com>, August 1997 - April 1999

db_count

```
#!/perl
# dbcount -- Emit a count of the table sizes in a given Oracle database
#   Written by Clint Goss <clint@goss.com>, November 1998
...

# Perl Modules

use OraConnection;      # Connection to Oracle, incl DBI and DBD::Oracle
use Getopt::Std;      # Command-line option processing
...

# Set up the connection to Oracle

my ($dbc) = new OraConnection;
$dbc->connect ($database, $username, $password);

# Fetch the list of Table / View names

my ($tableOrViewNames) = $dbc->tableOrViewNames ();

printf "%30s %5s %7s\n", "Table", "Type", "Record Count";
printf "%30s %5s %7s\n", "-----", "-----", "-----";

foreach $tableOrViewName (@$tableOrViewNames) {

    printf "%30s %5s %7d\n", $tableOrViewName,
        $dbc->tabtype ($tableOrViewName),
        $dbc->rowCount ($tableOrViewName);
}
```

Code for fetchAll() from OraConnection.pm

```
# Once-only execution of a query.
#
# Returns the entire result set of a query as an array reference.
# The array contains zero or more array references, each of which represents
# one row of the result. Each result array/row holds the field values.
#
# If the 'get_col_names' param is set, it will return the column names of the
# columns returned in the first row.
#
# For example, if the resulting rows of "SELECT name, consort FROM KINGS" is:
#   henry, joan
#   henry, marie-gallant
#
# the resulting anonymous hash would be:
#   [ [ "henry", "joan" ], [ "henry", "marie-gallant" ] ]
#
# As a convenience, this routine returns the count of records/rows returned.
#
# For example, the following will fetch a ref to a list (and count) of
# the tables in the database.
#
```

```

# my ($rows, $rowCount) = $connection->fetchAll (
#     "SELECT * FROM user_tables");
#
# This method is potentially a huge memory hog.
# "Hello Sun? Yea, we need summore of those reeeely big servers"

sub fetchAll {
    my ($self, $query, $get_col_names) = @_;

    my ($sth) = $self->execMulti ($query);

    # Fetch an array with all the resulting rows

    my ($resultRows) = [];

    # DOES NOT WORK!!! The fetchall_arrayref does not seem to be
    # present in the Oracle DBD!!
    # $resultRows = $sth->fetchall_arrayref;

    # Build the result array ourselves.

    my ($rowCount) = 0;
    my ($singleRow);

    # Build the result array ourselves.

    if ($get_col_names) {
        my ($nfields) = $sth->{NUM_OF_FIELDS};

        my ($result) = [];
        my ($f);
        for ($f = 0; $f < $nfields; $f++) {
            push @$result, $sth->{NAME}->[$f];
        }
        push @$resultRows, $result;
        $rowCount++;
    }

    while ($singleRow = $sth->fetch) {
        my ($localRow) = [];
        my ($fieldValue);
        foreach $fieldValue (@$singleRow) {
            push @$localRow, $fieldValue;
        }
        push @$resultRows, $localRow;
        $rowCount++;
    }

    # And a fine meal it was

    $sth->finish || $self->fatal ("Can't finish");
    return ($resultRows, $rowCount);
}

```

cntFind

```
#!/usr/bin/perl
# cntFind -- Show information for all countries matching a pattern
#   Written by Clint Goss <clint@goss.com>, November 1998
...

# Command line processing

my ($pattern) = $ARGV[0];
if (! $pattern) { &usage(); exit; }

# Set up the connection to Oracle

my ($database, $username, $password) = qw (ora9 demo demo);
my ($dbc) = new OraConnection;
$dbc->connect ($database, $username, $password);

# Form a query

my ($uPattern) = uc ($pattern);

my ($stmt) = "SELECT " .
    "cnt_code_a2, cnt_code_a3, cnt_code_number, " .
    "cnt_name_upper_case, cnt_name_mixed_case " .
"FROM COUNTRIES " .
"WHERE cnt_name_upper_case LIKE " . $dbc->quote ("% " . $uPattern . "%");

# Look up multiple records.

my ($infoRows, $infoCount) = $dbc->fetchAll ($stmt);
if (! $infoCount) {
    print STDERR "No countries match your pattern '$pattern'\n";
    exit;
}

# Extract each answer, and display the result.
# Country number is always shown as 3 digits.

my ($infoRow);

foreach $infoRow (@$infoRows) {

    my ($a2, $a3, $number, $upper, $mixed) = @$infoRow;

    my ($number3) = sprintf ("%03d", $number);

    print <<END_OUTPUT ;
        A2 code: $a2
        A3 code: $a3
Country number: $number3
Name (UPPER): $upper
Name (Mixed): $mixed

END_OUTPUT
}
```

cntInfo

```
#!/usr/bin/perl
# cntInfo -- Return information about a country, given a country code
#   Written by Clint Goss <clint@goss.com>, November 1998
...

# Command line processing

my ($cntID) = $ARGV[0];
if ((length ($cntID) < 2) || (length ($cntID) > 3)) { &usage(); exit; }

# Set up the connection to Oracle

my ($database, $username, $password) = qw (ora9 demo demo);
my ($dbc) = new OraConnection;
$dbc->connect ($database, $username, $password);

# Form a query, based on a 2- or 3-character code

$cntID = uc ($cntID);

my ($stmt) = "SELECT " .
    "cnt_code_a2, cnt_code_a3, cnt_code_number, " .
    "cnt_name_upper_case, cnt_name_mixed_case " .
"FROM COUNTRIES ";

if (length ($cntID) == 2) {
    $stmt .= "WHERE cnt_code_a2 = " . $dbc->quote ($cntID);
} else {
    $stmt .= "WHERE cnt_code_a3 = " . $dbc->quote ($cntID);
}

# Look up the single record.
# Don't complain on a missing record, as we handle the error ourselves.

my ($cntInfo) = $dbc->fetchSingle ($stmt, "Y");
if (! $cntInfo) {
    print STDERR "There is no country with country code '$cntID'\n";
    exit;
}

# Extract the fields, and display the result.
# Country number is always shown as 3 digits.

my ($a2, $a3, $number, $upper, $mixed) = @$cntInfo;

my ($number3) = sprintf ("%03d", $number);

print <<END_OUTPUT ;
    A2 code: $a2
    A3 code: $a3
Country number: $number3
    Name (UPPER): $upper
    Name (Mixed): $mixed
END_OUTPUT
```

countriesAll.cgi

```
#!/usr/bin/perl
#
# countries.cgi -- CGI to display a table of ISO-3166 country information
#
#   Written by Clint Goss <clint@goss.com>, November 1998
#
BEGIN {
    # Access Perl modules

    #push (@INC, ...your lib directory here...);

    # Point to our preferred installation of Oracle

    # $ENV{'ORACLE_HOME'} = ...your Oracle Home here...;
}

# Perl Modules and Libraries

use OraConnection;      # Connection to Oracle, incl DBI and DBD::Oracle
require 'GBasic.pl';    # Generic CGI-handling perl library
use strict;             # Restrict unsafe variables, refs, barewords

# Read any data from the requesting form and then pick up the parameters
# for this script (from form data or specified on the URL).

my (%in) = &GBasic::ReadParse;

my ($page) = $in{"page"};

# Set up the connection to Oracle

my ($database, $username, $password) = qw (ora9 demo demo);
my ($dbc) = new OraConnection;
$dbc->connect ($database, $username, $password);
$dbc->setMsgFormat ("html");

print <<END_HEADER;
Content-type: text/html

<html><head><title>Country Information</title></head>
<body>
This page displays information on world countries.
The information is from the ISO-3166 standard.
<p>
END_HEADER

# Fetch the country info

my ($stmt) = "SELECT " .
    "cnt_code_a2, cnt_code_a3, cnt_code_number, " .
    "cnt_name_upper_case, cnt_name_mixed_case " .
"FROM COUNTRIES " .
"ORDER BY cnt_code_a3 ASC";

# Fetch multiple records.

my ($infoRows, $infoCount) = $dbc->fetchAll ($stmt);
```

```

if (! $infoCount) {
    print "There is no countries in the database.\n";
    print "</body></html>\n";
    exit;
}

# Extract each answer, and display the result.
# Country number is always shown as 3 digits.

print <<END_TABLE_HEADER;
<table border=1>
<tr> <th>A2</th> <th>A3</th> <th>Number</th>
    <th>Name (UPPER)</th> <th>Name (Mixed)</th>
END_TABLE_HEADER

my ($infoRow);

foreach $infoRow (@$infoRows) {

    my ($a2, $a3, $number, $supper, $mixed) = @$infoRow;

    my ($number3) = sprintf ("%03d", $number);

    print <<END_OUTPUT ;
<tr>
<td>$a2</td>
<td>$a3</td>
<td>$number3</td>
<td>$supper</td>
<td>$mixed</td>
END_OUTPUT
}

print "</table>\n";
print "</body></html>\n";

exit;

```

countries.cgi

```

#!/perl
#
# countries.cgi -- CGI to display a table of ISO-3166 country information
#
#   Written by Clint Goss <clint@goss.com>, November 1998
#
BEGIN {
    # Access Perl modules

    #push (@INC, ...your lib directory here...);

    # Point to our preferred installation of Oracle

    #ENV{'ORACLE_HOME'} = ...your Oracle Home here...;
}

# Perl Modules and Libraries

```

```

use OraConnection;      # Connection to Oracle, incl DBI and DBD::Oracle
use NavigationBar;     # Display an html page selection / navigation bar
require 'GBasic.pl';   # Generic CGI-handling perl library
use strict;            # Restrict unsafe variables, refs, barewords

# Read any data from the requesting form and then pick up the parameters
# for this script (from form data or specified on the URL).

my (%in) = &GBasic::ReadParse;

my ($currentPage) = $in{"currentPage"};
if (! $currentPage) { $currentPage = 1 }

# Set up the connection to Oracle

my ($database, $username, $password) = qw (ora9 demo demo);
my ($dbc) = new OraConnection;
$dbc->connect ($database, $username, $password);
$dbc->setMsgFormat ("html");

print <<END_HEADER;
Content-type: text/html

<html><head><title>Country Information</title></head>
<body>
This page displays information on world countries.
The information is from the ISO-3166 standard.
<p>
END_HEADER

# Fetch the country info

my ($stmt) = "SELECT " .
    "cnt_code_a2, cnt_code_a3, cnt_code_number, " .
    "cnt_name_upper_case, cnt_name_mixed_case " .
"FROM COUNTRIES " .
"ORDER BY cnt_code_a3 ASC";

# Fetch multiple records.

my ($infoRows, $infoCount) = $dbc->fetchAll ($stmt);
if (! $infoCount) {
    print "There is no countries in the database.\n";
    print "</body></html>\n";
    exit;
}

# Calculate the total number of pages of countries to display

my ($rowsPerPage) = 10;

my ($totalPages) = int (($infoCount + $rowsPerPage - 1) / $rowsPerPage);
if ($totalPages <= 0) { $totalPages = 1; }
if ($currentPage > $totalPages) { $currentPage = $totalPages; }

# Construct a navigation bar. It looks something like:
#
#           Page:  [Prev] 1 2 3 [Next]

my ($nav) = new NavigationBar ("block");

```

```

print $nav->bar ($totalPages, $currentPage, &GBasic::MyURL,
                "currentPage", 20, "img");

# Extract each answer, and display the result.
# Country number is always shown as 3 digits.

print <<END_TABLE_HEADER;
<center>
<table border=1>
<tr> <th>A2</th> <th>A3</th> <th>Number</th>
    <th>Name (UPPER)</th> <th>Name (Mixed)</th>
END_TABLE_HEADER

my ($topRow) = ($currentPage - 1) * $rowsPerPage + 1;
my ($botRow) = $topRow + $rowsPerPage - 1;
if ($botRow > $infoCount) { $botRow = $infoCount; }

my ($row);
foreach $row ($topRow .. $botRow) {
    my ($infoRow) = $infoRows->[$row - 1];

    my ($a2, $a3, $number, $upper, $mixed) = @$infoRow;

    my ($number3) = sprintf ("%03d", $number);

    print <<END_OUTPUT ;
<tr>
<td>$a2</td>
<td>$a3</td>
<td>$number3</td>
<td>$upper</td>
<td>$mixed</td>
END_OUTPUT
}

print "</table>\n</center>\n";
print "</body></html>\n";

exit;

```

GBasic.pl

```
# GBasic.pl - Basic Perl Routines to deal with the HTTP protocol
#
# Originally written by Steven E. Brenner (S.E.Brenner@bioc.cam.ac.uk) and
# modified/hacked by Clint Goss (clint@goss.com) December 1995 and
# November 1998.
```

```
package GBasic;
```

```
# ReadParse
#
# Reads in GET or POST data, converts it to unescaped text, and puts
# one key=value in each member of the list "@in". Also creates key/value
# pairs in %in, using '\0' to separate multiple selections.
#
# Returns TRUE if there was input, FALSE if there was no input.
```

```
sub ReadParse {
    my ($in);

    # Read in the text, depending on the method

    if (&MethGet) {
        $in = $ENV{'QUERY_STRING'};
    } elsif (&MethPost) {
        read (STDIN, $in, $ENV{'CONTENT_LENGTH'});
    }

    my (@in) = split (/[&;]/, $in);

    my ($i, %in);
    foreach $i (0 .. $#in) {

        # Split into key and value - splits on the first =.

        my ($key, $val) = split ( /=/, $in[$i], 2);

        # Convert embedded '+' characters and any embedded %XX
        # hex numbers to their corresponding characters.
        # Also save the undecoded values, for anyone who is interested

        $key =~ tr/+/ /;
        $key =~ s/%(..)/pack ("c", hex ($1))/ge;

        $val =~ tr/+/ /;
        $val =~ s/%(..)/pack ("c", hex ($1))/ge;

        # Associate key and value
        # \0 is the multiple separator

        $in{$key} .= "\0" if (defined ($in{$key}));
        $in{$key} .= $val;
    }

    return %in;
}
```

...

Homework Assignment

1. Hand in your final project at the beginning of next class.
2. Download the handout for Session 10. This should be available at <http://www.goss.com/perl> by Wednesday, July 26 at 9AM. Bring a printout to the last session.¹

This handout is Copyright © 2000 Clint Goss, All Rights Reserved.