package ONE; import java.awt.EventQueue; import javax.swing.JFrame; import javax.swing.JPanel; import java.awt.BorderLayout; import java.awt.Color; import javax.swing.border.LineBorder; import org.eclipse.wb.swing.FocusTraversalOnArray; import java.awt.Component; import java.awt.Font; import java.awt.FlowLayout; public class ANVIL8 { private JFrame frmHelyxzionAnvil; /\*\* \* Launch the application. \*/ public static void main(String[] args){ EventQueue.invokeLater(new Runnable() { public void run() { try { ANVIL window = new ANVIL(); window.frmHelyxzionAnvil.setVisible(true); } catch (Exception e) { e.printStackTrace(); } } }); } /\*\* \* Create the application. \*/ public ANVIL() { initialize(); } /\*\* \* Initialize the contents of the frame. \*/ private void initialize() { frmHelyxzionAnvil = new JFrame("HELP"); frmHelyxzionAnvil.setTitle("HELYXZION ANVIL 4.0"); frmHelyxzionAnvil.getContentPane().setFont(new Font("Times New Roman", Font.BOLD, 14)); frmHelyxzionAnvil.getContentPane().setForeground(new Color(0, 0, 0)); frmHelyxzionAnvil.setBounds(100, 100, 900, 600); frmHelyxzionAnvil.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); JPanel panel = new JPanel(); panel.setBorder(new LineBorder(Color.RED, 3)); panel.setBackground(Color.WHITE); frmHelyxzionAnvil.getContentPane().add(panel, BorderLayout.NORTH); panel.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5)); frmHelyxzionAnvil.getContentPane().setFocusTraversalPolicy(new FocusTraversalOnArray(new Component[]{panel})); System.loadLibrary(arg0); } use CGI qw/:standard/; use DBI; $q = new CGI; $CGI = "prodemo.cgi"; $VCGI = "prodemoview.cgi"; $USER\_IDENT = 1; $SWF = "betaviewer.swf?dataSource=".$VCGI."&selectorSource=".$VCGI; $UIFILE = "hlxuif.html"; $multi = ""; #$multi = " ENCTYPE=\"multipart/form-data\" "; $HEADER = 0; $COOKIENAME = "charles"; $THISDOMAIN = "helyxzion.net"; my $myaction = $q->param('myaction') || ""; #print STDERR "\n\n\n\n\nmyaction = $myaction\n"; $getOrg = $q->param('getOrg') || ""; $findlocus = $q->param('findlocus') || ""; $show = $q->param('show') || ""; $view = $q->param('view') || ""; if ($myaction eq "logout") { &logout; exit; } #print $q->header(); ##### REGISTRATION ##### # registered and check\_login will print cookie header if successful if (!®istered(0)) { if (!&check\_login(1)) { print STDERR "Did not pass Check Login - printing Login Page\n"; &printLoginPage; } } else { print STDERR "Registered returned ![".®istered(0)."]\n"; } #$today = &ParseDate("today"); #my @parms = $q->param; #foreach $parm (@parms) { #print STDERR "q->param($parm) = ".$q->param($parm)."  
\n"; #} #print "  
\n"; ########################################################################### ##### ACTION HANDLER ##### if ($myaction eq "" || $myaction eq "getOrgs") { &getCustomData(); } elsif ($myaction eq "userlogin") { &getCustomData(); } elsif ($myaction eq "getData") { &getData; } elsif ($myaction eq "addrecord") { &getNewGene; } elsif ($myaction eq "delrecord") { &printDeleteRecs; } elsif ($myaction eq "addDBRec") { &addDBRec; } elsif ($myaction eq "delDBRec") { &delDBRec; } elsif ($myaction eq "logout") { &logout; exit; } else { &printLoginPage; } print STDERR "\nGot to main exit.\n\n"; exit; ##### SUBROUTINES sub getData { my $locus = $q->param('locus') || 1; my $organism = $q->param('organism'); my $PHASE = $q->param('phase') || 0; my $page = $q->param('page') || 1; my $direction = $q->param('direction') || 0; my $dba = &dbconnect; my $sqlStr0 = "SELECT Locus, Origin from userorigin where OriginID = '$locus' AND UserId = '$USER\_IDENT'"; my $stb = $dba->prepare($sqlStr0) or die "Couldn't prepare statement: ".$dba->errstr; $stb->execute or die "Couldn't execute statement: ".$stb->errstr; @originrecord = $stb->fetchrow(); my $loc = $originrecord[0]; my $org = $originrecord[1]; if ($PHASE == 1) { $org = "..".$org; } elsif ($PHASE == 2) { $org = ".".$org; } #print STDERR "Translating:\n$org\n"; $loc =~ s/([&=])//g;#\\$1/; $loc =~ s/\|/-/g;#\\$1/; $org =~ s/([&=])//g;#\\$1/; print STDERR "\nSending name = $loc \n\n"; #$TRANS = "NAME=".$loc; $TRANS = "NAME=ForFranz"; #$TRANS .= "&MAXDP=".sprintf("%d",(length($org)/3)); $TRANS .= "&TRANS=".&translateHelyxzion($org); print $q->header('application/x-www-form-urlencoded'); print $TRANS; print STDERR "Sending:\n$TRANS\n"; exit; } sub translateHelyxzion { $codons = shift(); my @helx; my $transstring = ""; while ($codons =~ s/^(...)//) { my $codon = $1; #print "Converting $codon   
\n"; # !"#$c{~()\*a,-./0123456789:;<}>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[|]^\_b push @helx, &convertLetter($codon); } #print STDERR "[$codons] of length(".length($codons).") is leftover from the translation.\n"; if (length($codons) > 0) { for (my $l=0; $l < (3-length($codons)); $l++) { $codons .= "."; } push @helx, &convertLetter($codons); } for (my $k=0; $k <= $#helx; $k++) { $paramstring .= $helx[$k]; } return $paramstring; } sub convertLetter { my $codon = shift(); $codon =~ tr/a-z/A-Z/; for ($codon) { # !"#$c{~()\*a,-./0123456789:;<}>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[|]^\_b /AAA/ && do { return '!'; }; /AAC/ && do { return '"'; }; /ACA/ && do { return '#'; }; /CAA/ && do { return '$'; }; /AAG/ && do { return 'c'; }; /AGA/ && do { return '{'; }; /GAA/ && do { return '~'; }; /CCA/ && do { return '('; }; /CAC/ && do { return ')'; }; /ACC/ && do { return '\*'; }; /AAT/ && do { return 'a'; }; /ATA/ && do { return ','; }; /TAA/ && do { return '-'; }; /CCC/ && do { return '.'; }; /CAG/ && do { return '/'; }; /ACG/ && do { return '0'; }; /AGC/ && do { return '1'; }; /GAC/ && do { return '2'; }; /GCA/ && do { return '3'; }; /CGA/ && do { return '4'; }; /GGA/ && do { return '5'; }; /GAG/ && do { return '6'; }; /AGG/ && do { return '7'; }; /CCG/ && do { return '8'; }; /CAT/ && do { return '9'; }; /ACT/ && do { return ':'; }; /TCA/ && do { return ';'; }; /CTA/ && do { return '<'; }; /CGC/ && do { return '}'; }; /GCC/ && do { return '>'; }; /TAC/ && do { return '?'; }; /ATC/ && do { return '@'; }; /TTT/ && do { return 'A'; }; /TTG/ && do { return 'B'; }; /TGT/ && do { return 'C'; }; /GTT/ && do { return 'D'; }; /TTC/ && do { return 'E'; }; /TCT/ && do { return 'F'; }; /CTT/ && do { return 'G'; }; /GGT/ && do { return 'H'; }; /GTG/ && do { return 'I'; }; /TGG/ && do { return 'J'; }; /TTA/ && do { return 'K'; }; /TAT/ && do { return 'L'; }; /ATT/ && do { return 'M'; }; /GGG/ && do { return 'N'; }; /GTC/ && do { return 'O'; }; /TGC/ && do { return 'P'; }; /TCG/ && do { return 'Q'; }; /CTG/ && do { return 'R'; }; /CGT/ && do { return 'S'; }; /GCT/ && do { return 'T'; }; /CCT/ && do { return 'U'; }; /CTC/ && do { return 'V'; }; /TCC/ && do { return 'W'; }; /GGC/ && do { return 'X'; }; /GTA/ && do { return 'Y'; }; /TGA/ && do { return 'Z'; }; /AGT/ && do { return '['; }; /GAT/ && do { return '|'; }; /GCG/ && do { return ']'; }; /CGG/ && do { return '^'; }; /ATG/ && do { return '\_'; }; /TAG/ && do { return 'b'; }; /\.+/ && do { print STDERR "[$codon] is hanger.\n"; return 'y'.$codon; }; /[^ACTG]/i && do { print STDERR "[$codon] is ambig.\n"; return 'x'.$codon; }; } } sub url\_encode { my $str = shift(); $str =~ s/([^\$])/uc sprintf("%%%02x",ord($1))/eg; $str =~ s/([^a-zA-Z0-9\_\-.])/uc sprintf("%%%02x",ord($1))/eg; return $str; } sub url\_decode { my $str = shift(); $str =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg; return $str; } sub delDBRec { my $dba = &dbconnect; my $ORGNID = $q->param('OriginID') || ""; if ($ORGNID eq "") { &printError("No Origin ID given"); } $ORGNID =~ s/[^0-9]//g; my $sqlStr0 = "DELETE FROM userorigin WHERE OriginID='$ORGNID'"; $stb = $dba->prepare($sqlStr0); $stb->execute or &printError("Couldn't execute statement: ".$stb->errstr); $stb->finish; $dba->disconnect; &getCustomData; #my $linktext = "[Add a new gene](file:///C:\%22$CGI?myaction=addrecord\%22)  
\n"; #$linktext .= "[Delete gene](file:///C:\%22$CGI?myaction=delrecord\%22)  
\n"; #&printHlxHeader("Custom Data","Custom Data","",$linktext); #print "Record Deleted.  
  
\n"; #&printLogoutButton; #&printHlxFooter; } sub addDBRec { my $dba = &dbconnect; my $uid = $q->param('uid') || 5; # Default beta tester = 5 my $NAME = $q->param('geneName') || "Unknown"; my $ORGN = $q->param('codons') || ""; if ($ORGN eq "") { &printError("No codons given"); } $ORGN =~ tr/A-Z/a-z/; $ORGN =~ s/[^a-z]//g; $NAME =~ s/'/\\'/g; $NAME =~ s/\|/-/g; my $sqlStr0 = "INSERT INTO userorigin VALUES ('','$uid','$NAME','$ORGN')"; $stb = $dba->prepare($sqlStr0); $stb->execute or &printError("Couldn't execute statement: ".$stb->errstr); $stb->finish; $dba->disconnect; &getCustomData; } sub getNewGene { &printHlxHeader("Enter Custom Data","Enter Custom Data","",""); print <

Top of Form

|  |  |
| --- | --- |
| Enter new gene for custom data | |
| You may cut and paste [ACTG] for translation.  https://www.ncbi.nlm.nih.gov/genome/gdv/    Locus:       Nucleic Acid String: | |
| Visible by: \n"; } print < |  |
| [Cancel](file:///C:\Users\Dr.%20Charles%20Stevens\Desktop\$CGI) |  |

Bottom of Form

endOfFormPage2 &printLogoutButton; &printHlxFooter; } sub printDeleteRecs { my $orgName = $organism; print STDERR "ORGANISM = $orgName\n"; $orgName = &url\_decode($orgName); $orgNameEnc = &url\_encode($orgName); my $numCols = 3; my $dba = &dbconnect; my ($sta,$stb,$numOfLocs,$numOfPages,$rowsPerCol,$sqlStr0,$numRows,$instruct); my $i = 0; my $start = ($page - 1) \* $ataTime; my $sqlStr0 = "SELECT OriginID, Locus from userorigin WHERE UserId = '".$USER\_IDENT."' ORDER BY OriginID"; $stb = $dba->prepare($sqlStr0); $stb->execute or &printError("Couldn't execute statement: ".$stb->errstr); $numRows = $stb->rows; my $linktext = ""; #$linktext = "[Add a new gene](file:///C:\%22$CGI?myaction=addrecord\%22)  
\n"; #$linktext .= "[Delete gene](file:///C:\%22$CGI?myaction=delrecord\%22)  
\n"; $linktext .= "[View a gene](file:///C:\%22$CGI\%22)  
\n"; &printHlxHeader("Custom Data","Custom Data","",$linktext); print "**Please choose a record number to delete:**  
\n"; print "

Top of Form

\n"; print "\n"; print "\n"; print " \n"; print " \n"; } print " \n"; print " \n"; print "

|  |  |
| --- | --- |
| \n"; if ($numRows > 0) { while (@found = $stb->fetchrow()) { print " $found[1] \n"; } } else { print " | Found no results. |

\n"; print "

Bottom of Form

\n"; print $pageText; $stb->finish; $dba->disconnect; &printLogoutButton; &printHlxFooter; } sub getCustomData { my $orgName = $organism; print STDERR "ORGANISM = $orgName\n"; $orgName = &url\_decode($orgName); $orgNameEnc = &url\_encode($orgName); my $numCols = 3; my $dba = &dbconnect; my ($sta,$stb,$numOfLocs,$numOfPages,$rowsPerCol,$sqlStr0,$numRows,$instruct); my $i = 0; my $start = ($page - 1) \* $ataTime; my $sqlStr0 = "SELECT OriginID, Locus from userorigin WHERE UserId = '".$USER\_IDENT."' ORDER BY OriginID"; $stb = $dba->prepare($sqlStr0); $stb->execute or &printError("Couldn't execute statement: ".$stb->errstr); $numRows = $stb->rows; my $linktext = ""; #if ($USER\_IDENT == 3 || $USER\_IDENT == 1) { #if ($USER\_IDENT == 3 || $USER\_IDENT == 1 || $USER\_IDENT == 5) { $linktext = "[Add a new gene](file:///C:\%22$CGI?myaction=addrecord\%22)  
\n"; $linktext .= "[Delete gene](file:///C:\%22$CGI?myaction=delrecord\%22)  
\n"; #} $linktext .= "[View a gene](file:///C:\%22$CGI\%22)  
\n"; &printHlxHeader("Custom Data","Custom Data","",$linktext); print "**Please note this current version will not automatically load sequence. When Pro Viewer loads, select 'Load new data set' from one of the Graph Control panes.**  
  
"; print "\n"; print " \n"; print " \n"; } print " \n"; print " \n"; print "

|  |  |
| --- | --- |
| \n"; if ($numRows > 0) { while (@found = $stb->fetchrow()) { print "[$found[1]](file:///C:\%22$SWF\%22) \n"; } } else { print " | Found no results. |

\n"; print $pageText; $stb->finish; $dba->disconnect; &printLogoutButton; &printHlxFooter; } sub printHlxHeader { my $str1 = shift() || ""; # Title my $str2 = shift() || ""; # Page Header my $str3 = shift() || ""; # Scripts my $str4 = shift() || ""; # LeftBar if (!$HEADER) { print $q->header(); print STDERR "Manually printed header without cookie.  
\n"; } $thescript = <