Brian Kiser
June 2003

Driver History Records Application

J2EE SDK 1.3

Commonwealth of Kentucky
Frankfort, Kentucky

Table of Contents

1.0 Work Sample Description

DHR, an acronym for Driver History Records, is an application written for the Commonwealth of Kentucky that allows Internet customers to purchase a copy of their driver's record online. My primary programming assignment for DHR was a sesson bean and web pages to allow credit card payment entry and processing.

The session bean, called Process Internet Request, calls methods to generate the email that is sent to the customer, generate the history reports, and process the payment information using an application written by AMS called the ePayment Gateway. It is the most complex piece of code in DHR, and is considered the backbone of the application.

2.0 Skills Demonstrated

2.1 Web Development competency using J2EE, HTML, and JavaScript

The DHR application was written with the Java 1.3 Software Development Kit using the J2EE architecture, which for this project included the Java language, JSP, Entity Beans, Session Beans, and state maintenance. Web pages use a combination of HTML and JSP to display dynamic content, as well as JavaScript for validation. Validation is performed on the server side as well, so that users who have JavaScript disabled are accommodated.

DHR is a 3-tier application with business logic residing in the middle-tier only.

This particular Java solution was implemented using IBM's Websphere Studio Application Developer (WSAD) 4.0 and DB2 for the database.

2.2 Good programming style

In general, I follow established programming conventions. This code will demonstrate:

- Good documentation habits, such as documentation of difficult code and modification history.
- Avoidance of "magic numbers" in favor of constants or database fields.
- Efficient class design and use of object-oriented principles, when appropriate.
- Adherence of other Sun-established Java conventions, such as the positioning of braces and the use of JavaDoc comments.

2.3 Ability to work in a team environment

Versioning software was used to keep track of source code versions, and UML (Unified Modeling Language) was used for application design. The DHR development staff consisted of team members that had a wide variety of programming experience and backgrounds. The team size was originally 7, but shrank to 5 by the completion date.

3.0 Difficulties Encountered

Difficulties encountered were probably more than typical, mostly due to the varied experience levels of the development staff and team member turnaround.

As mentioned in section 2.3 above, there were a variety of personalities and skill levels involved in the project. Effectively communicating technical ideas was a challenge at times, as experience among team members varied greatly. Most of the team had never written Java code before, so this was a learning experience. Additionally, we lost three team members, including our project lead, not long into the project. We were given a new project lead reasonably fast, and near the end of the project we gained a team member.

Overall, the project was successfully completed despite these difficulties.

4.0 Screen Shots

Please see Appendix A for enlarged versions.

4.1 Payment Entry

Note that the total cost is redisplayed so that the user knows what will be billed to his card. Also, dropdowns are included to help eliminate data entry errors. The four types of cards that are accepted are displayed.

Since every attempt to process a credit card costs a fee, thorough payment entry validation is performed using JavaScript. Server-side validation is also performed for those users who have JavaScript disabled in the browser.

The color scheme was designed to match other existing systems, but the screen design itself is mine.

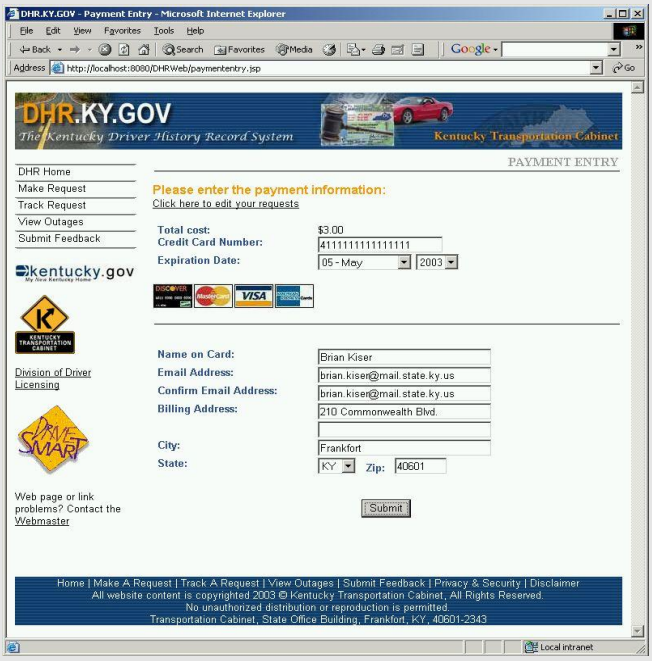


Figure A – The Payment Entry page

This is where the user enters the payment information. Validation is performed on both the client and server-side.

4.2 Payment Confirmation

The payment confirmation page is a simple page that just redisplays payment information that the user can check for accuracy before placing his order.

Figure B – Payment Confirmation

This is where the user reviews the entered payment information.

The menus and associated images were added by using server side includes. All fonts, colors, and positioning were done via style sheet instead of HTML tables.

5.0 Code Listing

Although I added code throughout the DHR application, ProcessRequestSessionBean.java was a class that I wrote almost entirely. In conjunction, I added some methods to the RequestServlet.java class. I have chosen to include only my classes in the interest of brevity, as they offer a sample of my Java code without adding pages upon pages of other people's code.

5.1 ProcessRequestSessionBean.java

For brevity, I have omitted two methods written by other developers. I have left their entries in the Modification History so as to maintain an accurate listing of changes made. Note that word-wrap may cause code to look messy in Word.

```
package us.ky.state.kytc.dhr.request;

import java.util.*;
import us.ky.state.kytc.util.*;
import us.ky.state.kytc.dhr.common.*;
import us.ky.state.kytc.dhr.entitybeans.*;
import us.ky.state.kytc.dhr.batch.*;
import com.ams.psg.epay.proxy.*;


/**
 * Bean implementation class for Enterprise Bean: ProcessRequestSession
 *
 * This class processes the request by processing the credit card, generating
 * the report, producing the email notification, and saving all appropriate
 * information to the database.
 *
 * @author Brian Kiser
 */

/*
 * Maintenance History
 *
 * Date             Programmer          Description
 * ---------------------------------------------------------------------------
 * 01/29/2003       Brian Kiser         Began initial code
 * 02/??/2003       Brian Kiser         Wrote getNextID, saveRequest, updateRequest methods
 * 02/05/2003       Brian Kiser         Wrote insertAllRecords and various cleanup code
 * 02/12/2003       Brian Kiser         Began processPayment and various cleanup code
 * 02/13/2003       Brian Kiser         Added processGeneralException and continued cleanup
 *                                      Removed unneeded updateNewRequest method
 * 02/19/2003       Brian Kiser         Modified saveRequest and processPayment to match sequence diagram
 * 02/24/2003       Brian Kiser         Made some finishing touches on the first draft
 * 02/27/2003       Brian Kiser         Renamed updateRequest to markPaymentAuthorized
 * 03/06/2003       Brian Kiser         Added e.fillInStackTrace() call to error handling method
 * 03/05/2003       Jennifer D          Added getDHRRequestFromRequestNumber method
 * 03/12/2003       Ben Putnam          Added setRequestValues method
 * 03/12/2003       Ben Putnam          Modified getDHRRequestFromRequestNumber to use setRequestValues
 * 03/26/2003       Brian Kiser         Separated report generation/email notification out of processPayment
 *                                      so that those activities can be done separately.  Created new method
 *                                      called finalizeRequest.
 * 04/22/2003       Cliff Lawson        Added Request Status to DHR Object being populated in setRequestValues
 * 05/13/2003       Brian Kiser         Replaced setRequestValues, per instructions from Jerry/C. Brooks
 * 05/14/2003       Brian Kiser         Put EPAY_MERCHANT_ID in ejb-jar instead of using a constant
 * 05/23/2003       Brian Kiser         Implemented requested changes, per code walkthrough.
```

```
 * 05/27/2003       Brian Kiser          Continued implementing changes, per code walkthrough.
 */

public class ProcessRequestSessionBean implements javax.ejb.SessionBean {

    // Here are constants for the values I am sending to the ePay application.

    private static final String EPAY_SUCCESS                = "0";
    private static final String EPAY_FAILURE                = "1";
    private static final String EPAY_MERCHANT_ID            = "EPAY_MERCHANT_ID";
    private static final String EPAY_TRANSACTION_TYPE       = "CHARGE";
    private static final String EPAY_DESC                   = "DHR REQUEST";
    private static final String EPAY_SMARTCODE              = "";
    private static final String EPAY_TRANSACTION_APPROVED = "0";

    // ------ The "canned" methods start here.

    private javax.ejb.SessionContext mySessionCtx;
    /**
     * getSessionContext
     */
    public javax.ejb.SessionContext getSessionContext() {
            return mySessionCtx;
    }
    /**
     * setSessionContext
     */
    public void setSessionContext(javax.ejb.SessionContext ctx) {
            mySessionCtx = ctx;
    }
    /**
     * ejbActivate
     */
    public void ejbActivate() {
    }
    /**
     * ejbCreate
     */
    public void ejbCreate() throws javax.ejb.CreateException {
    }
    /**
     * ejbPassivate
     */
    public void ejbPassivate() {
    }
    /**
     * ejbRemove
     */
    public void ejbRemove() {
    }


    // ------ Custom methods start here.

    /**
     * wraps several database methods to save a request
     *
     * @throws DRHException error indicator
     * @param DHRRequest reference to the request object
```

```
 * @return Thdrre reference to the new row
 */
public Tdhrre saveRequest (DHRRequest req) throws DHRException {

	Tdhrre row = null;		// Must define outside the try/catch.

	int uniqueID = req.getRequestID();

	// Get a reference to the home object.

	try {
		TdhrreHome home = (TdhrreHome) JNDIUtility.getHomeInterfaceByJNDILookup
			(DHREntityBeanLookupConstants.DHR_TDHRRE_LOOKUP, TdhrreHome.class);

		// If uniqueID is NOT 0, then that means a request has already been made,
		// so I use that existing table record.  If uniqueID is 0, then a new
		// record must be created.

		if (uniqueID == 0) {
			try {
				uniqueID = getNextID();			// Get a new request ID.
				req.setRequestID (uniqueID);	// Then save it to the request object.

				CreditCard cc = req.getCreditCardInfo();

				// Create a new TDHRRE record with the ID ("uniqueID") and the default
				// Request Status Code (DHR_RS_IN_PROCESS) stored initially.
				// Return row so it can be accessed by processPayment

				row = home.create (new Integer (uniqueID), new Short
					(DHRRequest.DHR_RS_IN_PROCESS), req.getBeginningTime(),
					req.getEmailAddress(),

					req.getCreditCardLast4Digits(), cc.getAmount());

					// At this point, a request record should exist, so update the corresponding
					// detail records.

				try {
					// Store a new record for each driver's history record in the request
					// into the Request Record table.
					insertAllRecords (row, req);
				} catch (Exception e) {
					processGeneralException (e, "saveRequest",
						DHRExceptionConstants.EX_REQUEST_NOT_UPDATED);
				}

			} catch (DHRException ex) {
				throw ex;

			} catch (Exception e) {
				processGeneralException (e, "saveRequest",
					DHRExceptionConstants.EX_REQUEST_NOT_UPDATED);
			}
		} else {
			// The requestID is not 0, so a record must already exist.  Look it up.

			row = home.findByPrimaryKey (new TdhrreKey (new Integer (uniqueID)));
		}
```

```
        } catch (Exception e) {
                processGeneralException (e, "saveRequest",
                        DHRExceptionConstants.EX_REQUEST_NOT_UPDATED);
        }

        return row;
}


/**
 * returns a unique, sequential ID number
 *
 * @throws DRHException error indicator
 * @returns int the ID number
 */
private int getNextID() throws DHRException {

        int uniqueID = -1;

        try {
                TdhrpkHome home = (TdhrpkHome) JNDIUtility.getHomeInterfaceByJNDILookup
                        (DHREntityBeanLookupConstants.DHR_TDHRPK_LOOKUP, TdhrpkHome.class);

                // findByPrimaryKey to get the right row.

                Tdhrpk remoteUniqueId = home.findByPrimaryKey (
                        new TdhrpkKey (DHRTableKeyConstants.DHR_PK_TDHRRQ_KEY));

                // Now get the new ID
                uniqueID = remoteUniqueId.getNextUniqueID();

        } catch (Exception e) {
                processGeneralException (e, "getNextID", DHRExceptionConstants.EX_REQUEST_NOT_UPDATED);
        }

        return uniqueID;
}


/**
 * updates the new Internet Request record with info from the request object
 *
 * @param Tdhrre row in the request table
 * @param DHRRequest an instance of the request object
 * @throws DHRException for any other errors
 */
private void markPaymentAuthorized (Tdhrre row, DHRRequest req) throws DHRException {

        // Request number and request status code are already inserted, so
        // put the remaining request object attributes into the request table.

        try {
                row.setRe_charge_auth_code (req.getCreditCardInfo().getAuthorizationCode());
                updateRequestStatus (req, DHRRequest.DHR_RS_AUTHORIZED);

        } catch (Exception e) {
                processGeneralException (e, "markPaymentAuthorized",
                        DHRExceptionConstants.EX_REQUEST_NOT_UPDATED);
        }
```

```
}


/**
 * store a new record for each driver's history record in the request
 *
 * @param Tdhrre from the request table
 * @param DHRRequest an instance of the request object
 * @throws DRHException for any other errors
 */

private void insertAllRecords (Tdhrre row, DHRRequest req) throws DHRException {

        // The information below goes into the Request Record table.

        try {
                TdhrrdHome home = (TdhrrdHome) JNDIUtility.getHomeInterfaceByJNDILookup
                        (DHREntityBeanLookupConstants.DHR_TDHRRD_LOOKUP, TdhrrdHome.class);
                DHRRecord record = null;

                // Loop through all the driving records stored in the
                // request object and create a table record for each one.

                int numRecs = req.getRecordCount();
                for (int i = 0; i < numRecs; i++) {
                        record = req.getRecord (i);

                        // Insert the record and save the data using the create method.

                        Tdhrrd childRow = home.create (record.getFirstName(), record.getLastName(),
                                record.getDriverLicenseNumber(), row);
                }

        } catch (Exception e) {

                try {
                        processGeneralException (e, "insertAllRecords",
                                DHRExceptionConstants.EX_RECORD_NOT_SAVED);
                } catch (Exception ex) {}               // Error trying to log error to database
        }
}


/**
 * calls methods to generate reports, produce the email
 * notification, mark the request complete and display it.
 *
 * @param DHRRequest an instance of the request object
 * @param byte number of attempts to process a transaction
 * @return DHRRequest a reference to the updated request object
 * @throws DHRException error indicator
 */

public DHRRequest processPayment (DHRRequest req, byte numAttempts, byte maxAttempts) throws DHRException {

        CreditCard cc = req.getCreditCardInfo();

        // SaveRequest gets the sequential ID number and inserts the request information
        // into the database.  It will return a reference to the new record in Tdhrre.
```

```
Tdhrre row = saveRequest (req);


// Use the ePayment Gateway to attempt to charge the user's credit card.

EpayProxy proxy = new EpayProxy();
proxy.StartRequest (JNDIUtility.getEJBEnvironmentValue (EPAY_MERCHANT_ID), "", "",
        Integer.toString (req.getRequestID()), "", EPAY_TRANSACTION_TYPE, EPAY_DESC,
        Integer.toString (req.getRequestID()), "", "", "");
proxy.AddItem (cc.getAmount().toString(), EPAY_SMARTCODE, EPAY_DESC + Integer.toString
        (req.getRequestID()), "");
proxy.AddBillingInfo (cc.getNameOnCard(), cc.getBillingAddress1() + " " + cc.getBillingAddress2(),
        cc.getCity(), cc.getState(), cc.getZip(), "", "", "");
proxy.ChargeCard (cc.getAccountNumber(), cc.getExpirationMonth(), cc.getExpirationYear(), "");


// Check the status code of the transaction.

if (proxy.getEpayCompletionCode().equals (EPAY_SUCCESS)) {
        String responseCode  = proxy.getMSP_ResponseCode();
        String authNum       = proxy.getMSP_AuthCode();

        cc.setAuthorizationCode (authNum);

        if (responseCode.equals (EPAY_TRANSACTION_APPROVED)) {

                try {
                        req.getCreditCardInfo().setAuthorizationCode (authNum);

                        // Update the request table with info from the DHRRequest object.
                        markPaymentAuthorized (row, req);

                          } catch (DHRException e) {
                        throw e;
                }

                          // finalizeRequest is called to generate the reports and produce the email notification.  It is
                          // separated so that other outside methods can use it without having to deal with payment
                          // authorization.

                finalizeRequest (req);

        } else {                // The transaction was not approved.

                DHRCreditCardDeclinedException ccd;

                if (numAttempts >= maxAttempts) {

                        // If the user has exceeded the maximum number of attempts,
                        // then clear the fields on the screen.

                        cc.resetAllFields();

                        updateRequestStatus (req, DHRRequest.DHR_RS_REJECTED);
                        ccd = new DHRCreditCardDeclinedException
                                (DHRUtility.getUserErrorDescription
                                (DHRExceptionConstants.EX_REJECTED) + " (Code " + responseCode + ")");

                } else {
```

```
                        // The authorization attempt was unsuccessful, but
                        // they've not yet went over the maximum number of attempts.
                        updateRequestStatus (req, DHRRequest.DHR_RS_DECLINED);
                        ccd = new DHRCreditCardDeclinedException
                                (DHRUtility.getUserErrorDescription
                                (DHRExceptionConstants.EX_NOT_APPROVED) + " (Error " +
                                responseCode + ")");
                    }

                    // Save the request object to the exception object
                        // so that the servlet can reference it.

                    ccd.setAffectedObject (req);
                    throw ccd;
                }
            } else {

                // If ePay screws up, log an error and display a message.

                DHRException ccd = new DHRException (DHRUtility.getUserErrorDescription
                    (DHRExceptionConstants.EX_EPAY_PROBLEM));
                processGeneralException (ccd, "processPayment", DHRExceptionConstants.EX_EPAY_PROBLEM);
            }

            return req;
    }


    /**
     * calls methods to generate reports, create a link, and
     * produce the email notification.
     *
     * @param DHRRequest an instance of the request object
     * @throws DHRException error indicator
     */

        public void finalizeRequest (DHRRequest req) throws DHRException {

            // This method can be used by other methods that wish to generate the report
            // and the email, but not deal with payment authorization.

            try {
                // Generate the reports.

                GenerateReportSessionHome reportsHome =
                    (GenerateReportSessionHome)JNDIUtility.getHomeInterfaceByJNDILookup
                    (DHRSessionBeanLookupConstants.DHR_GENERATE_REPORTS_LOOKUP,
                    GenerateReportSessionHome.class);
                GenerateReportSession remoteReport = reportsHome.create();
                remoteReport.generate3YearReports (req);

                markComplete (req);

                // By embedding the below try/catch around the email code, we ensure
                // that nothing happens if the email is not sent successfully.
                // The program will continue normally, minus the email.
                try {
                    SendEmailSessionHome emailHome =
```

```
                        (SendEmailSessionHome)JNDIUtility.getHomeInterfaceByJNDILookup
                        (DHRSessionBeanLookupConstants.DHR_PRODUCE_EMAIL_LOOKUP,
                        SendEmailSessionHome.class);
                SendEmailSession remoteEmail = emailHome.create();
                remoteEmail.sendEmailNotification (req);
            } catch (DHRException e) {}

            // After this method has completed, the servlet will
            // forward to the displayRequest page.

        } catch(DHRException e) {
            e.setAffectedObject(req);
            throw e;

        } catch(Exception e) {
            try {
                processGeneralException (e, "processPayment",
                        DHRExceptionConstants.EX_REPORT_NOT_GENERATED);
            } catch (Exception ex) {}
        }
    }


/**
 * sets the request ending time and updates request status
 * to indicate that the request is complete
 *
 * @param DHRRequest a reference to the request object
 * @throws DHRException error indicator
 */

private void markComplete (DHRRequest req) throws DHRException {

        TdhrreHome reHome = null;
        Tdhrre reRemote = null;

        req.setEndingTime (new java.sql.Timestamp (new Date().getTime()));

        try {
            reHome = (TdhrreHome) JNDIUtility.getHomeInterfaceByJNDILookup (
                    DHREntityBeanLookupConstants.DHR_TDHRRE_LOOKUP, TdhrreHome.class);
            reRemote = reHome.findByPrimaryKey(new TdhrreKey(new Integer(req.getRequestID())));
            reRemote.setRe_end_timestamp(req.getEndingTime());
        } catch (Exception e) {
            DHRUtility.logError ("ProcessRequestSessionBean", "markComplete", e.toString(), true);
            DHRException DHRe = new DHRException (DHRExceptionConstants.EX_DEFAULT_MESSAGE);
            throw DHRe;
        }

        reHome = null;
        reRemote = null;

        updateRequestStatus (req, DHRRequest.DHR_RS_COMPLETED);
    }


/**
 * General error handling method
 *
```

```
     * @param Exception the exception object
     * @param String location of the error (typically a method name)
     * @param int a constant indicating the error type
     * @throws DHRException error indicator
     */

    private void processGeneralException (Exception e, String errorLocation, int DHRExConstant) throws DHRException{

        try {
            e.fillInStackTrace();
            DHRUtility.logError ("ProcessRequestSessionBean", errorLocation, e.toString(), true);

        } catch (Exception ex) {}                    // If logging fails, then we're hosed.

            DHRException DHRe = new DHRException (DHRExConstant,
                DHRUtility.getUserErrorDescription (DHRExConstant));
        throw DHRe;
    }


    /**
     * Updates the request status in both the request object
     * and the appropriate table.
     *
     * @param DHRRequest reference to the request object
     * @param status request status
     * @throws DHRException error indicator
     */

    private void updateRequestStatus (DHRRequest req, short status) throws DHRException {

        req.setRequestStatus (status);

        try {
            TdhrreHome reHome = (TdhrreHome) JNDIUtility.getHomeInterfaceByJNDILookup
                (DHREntityBeanLookupConstants.DHR_TDHRRE_LOOKUP, TdhrreHome.class);

            Tdhrre reRow = reHome.findByPrimaryKey (new TdhrreKey (new Integer (req.getRequestID())));

            TdhrrsHome rsHome = (TdhrrsHome) JNDIUtility.getHomeInterfaceByJNDILookup
                (DHREntityBeanLookupConstants.DHR_TDHRRS_LOOKUP, TdhrrsHome.class);

            Tdhrrs rsRow = rsHome.findByPrimaryKey (new TdhrrsKey (new Short (req.getRequestStatus())));

            reRow.setRs_re_fk (rsRow);

        } catch (Exception e) {
            DHRUtility.logError ("ProcessRequestSessionBean", "updateRequestStatus", e.toString(), true);
            DHRException DHRe = new DHRException (DHRExceptionConstants.EX_DEFAULT_MESSAGE);
            throw DHRe;
        }
    }
}
```

5.2 RequestServlet.java

The servlet is a reasonably large piece of code and my contribution was minimal, so for brevity I have included only the comments at the beginning and the code I wrote. Note that word-wrap may cause code to look messy.

```
package us.ky.state.kytc.dhr.request.servlet;

import javax.servlet.http.*;
import javax.servlet.*;
import javax.naming.*;
import javax.rmi.PortableRemoteObject;
import java.io.*;
import java.util.*;

import us.ky.state.kytc.dhr.entitybeans.*;
import us.ky.state.kytc.dhr.common.*;
import us.ky.state.kytc.dhr.request.*;
import us.ky.state.kytc.util.*;


/**
 * This class controls the flow for a DHR Request.
 *
 * @author Jerry Duffy
 * @author Cliff Brooks
 * @author Cliff Lawson
 * @author Jim Winders
 * @author Jennifer Dearborn
 * @author Brian Kiser
 * @author Ben Putnam
 */


/*
 * Maintenance History
 *
 * Date             Programmer          Description
 * ---------------------------------------------------------------------------------------------
 * 10/02/2002       DHR Team            Began initial code
 * 02/06/2003       Brian Kiser         Added first draft of code to update Request table after payment was authorized
 * 02/13/2003       Brian Kiser         Worked on processRequest and various cleanup of code
 * 02/18/2003       Brian Kiser         Began verification that sequence diagram matches source code
 * 02/19/2003       Brian Kiser         Modified saveRequest and finalizeRequest to match sequence diagram
 * 02/25/2003       Cliff Lawson        Added first draft of code for View Outages
 * 03/06/2003       Brian Kiser         Added e.fillInStackTrace() call to processGeneralException method
 * 04/11/2003       Brian Kiser         Removed unnecessary finalizeRequest method
 * 04/15/2003       Cliff Lawson        Added first draft of code for Display Request (Basic Flow 1)
 * 04/22/2003       Cliff Lawson        Added first draft of code for Display Request (Basic Flow 2)
 * 04/29/2003       Brian Kiser         Added sanity checks to credit card processing portion
 * 05/16/2003       Kiser/Duffy         Added getEnvVariable and buildDisplayRequestURL
 */

public class RequestServlet extends HttpServlet {

        […Lots of code that others wrote is deleted here…]

    /**
     * Processes the Payment Request
     *
     * @param HttpServletRequest represents the request from the server
     * @return String page to call next
     */
```

```
private String processPaymentEntry (HttpServletRequest request) {

    // Below is the page that should run if things go well (default value).
    String returnPage = DHRServletConstants.PAGE_CREDIT_CARD_VERIFICATION;

    // Get references to the session and to the DHRRequest object, then populate
    // the CreditCard object with the data that the user entered.
    HttpSession session = request.getSession();
    DHRRequest newRequest = (DHRRequest)session.getAttribute (DHRServletConstants.DHRREQUEST);


    if (newRequest == null)                    // Sanity check: newRequest must not be null.
        returnPage = DHRServletConstants.PAGE_HOME;
    else {
        // Populate the CreditCard object with data retrieved from the Payment Entry page.
        CreditCard cc = newRequest.getCreditCardInfo();

        cc.setAccountNumber (request.getParameter
            (DHRServletConstants.PARAM_CREDIT_CARD_NUMBER));
        cc.setExpirationDate (request.getParameter
            (DHRServletConstants.PARAM_CREDIT_CARD_EXPIRATION_MONTH),
            request.getParameter
            (DHRServletConstants.PARAM_CREDIT_CARD_EXPIRATION_YEAR));
        cc.setAmount (new java.math.BigDecimal ((double)(newRequest.calculateTotal())));
        cc.setNameOnCard (request.getParameter
            (DHRServletConstants.PARAM_CREDIT_CARD_NAME));
        cc.setBillingAddress1 (request.getParameter
            (DHRServletConstants.PARAM_CREDIT_CARD_BILLING_ADDRESS1));
        cc.setBillingAddress2 (request.getParameter
            (DHRServletConstants.PARAM_CREDIT_CARD_BILLING_ADDRESS2));
        cc.setCity (request.getParameter (DHRServletConstants.PARAM_CREDIT_CARD_CITY));
        cc.setState (request.getParameter (DHRServletConstants.PARAM_CREDIT_CARD_STATE));
        cc.setZip (request.getParameter (DHRServletConstants.PARAM_CREDIT_CARD_ZIP));

        // Save the current time and date.
        newRequest.setBeginningTime (new java.sql.Timestamp (new Date().getTime()));
        newRequest.setEmailAddress (request.getParameter
            (DHRServletConstants.PARAM_EMAIL_ADDRESS));
        newRequest.setConfirmEmailAddress(request.getParameter
            (DHRServletConstants.PARAM_EMAIL_ADDRESS_CONFIRMATION));

        // Get the account number, then get the last 4 digits from that number and store them.
        String acct = cc.getAccountNumber();
        newRequest.setCreditCardLast4Digits (acct.substring (acct.length() - 4, acct.length()));

        // Save the DHR Request object back to the session.
        session.setAttribute (DHRServletConstants.DHRREQUEST, newRequest);

        // Verify that the user entered the credit card information correctly.
        try {
            cc.verify();

            if (!Email.areEmailsTheSame(newRequest.getEmailAddress(),
                newRequest.getConfirmEmailAddress())) {
                DHRDataValidationException dhre = new DHRDataValidationException
                (DHRExceptionConstants.EX_EMAILS_DONT_MATCH,
                DHRUtility.getUserErrorDescription
                (DHRExceptionConstants.EX_EMAILS_DONT_MATCH));
```

```
                        throw dhre;
                    }

                } catch(DHRDataValidationException e) {
                    request.setAttribute (DHRServletConstants.USER_MESSAGE,e.getUserErrorDescription());
                    returnPage = DHRServletConstants.PAGE_BILLING_INFO_ENTRY;

                } catch (CreditCardValidationException e) {

                    // If an error was found in the entered information, then return the user
                    // to the billing entry screen.
                    request.setAttribute (DHRServletConstants.USER_MESSAGE,e.getUserErrorDescription());
                    returnPage = DHRServletConstants.PAGE_BILLING_INFO_ENTRY;

                } catch (Exception e) {
                    // Some unexpected error occurred.
                    returnPage = processGeneralException (e, request, "processPaymentEntry");
                }
            }

            return returnPage;
        }


    /**
     * This method processes the request.  It executes after the user
     * hits the Submit button on the credit card verification page.
     *
     * @param HttpServletRequest the request from the server
     * @param HttpResponseRequest the response from the server
     * @return String return page
     */

        private String processRequest (HttpServletRequest request, HttpServletResponse response) {

            // Get the maximum number of unsuccessful epay attempts from the xml
            // environment variable file.
            String epayMaxAttempts = getEnvVariable ("MAX_EPAY_ATTEMPTS");
            byte maxAttempts = Byte.valueOf (epayMaxAttempts, 10).byteValue();


            HttpSession session = request.getSession();

            // Using a byte avoids lots of repeated casting, because numbers are
            // the int data type by default.

            Byte numAttempts = (Byte)session.getAttribute (DHRServletConstants.NUMATTEMPTS);

            if (numAttempts == null) {

                // This is the first time on this page, so initialize all the fields.

                numAttempts = new Byte ((byte)0);
                session.setAttribute (DHRServletConstants.NUMATTEMPTS, new Byte ((byte)0));
            }

            byte byteNum = numAttempts.byteValue();
```

```
// If all goes well, we will forward to the displayrequest.jsp page when finished.
String returnPage = null;

// Retrieve an instance of the DHR Request object.
DHRRequest req = (DHRRequest) session.getAttribute (DHRServletConstants.DHRREQUEST);

if (req == null)                        // Sanity check

        returnPage = DHRServletConstants.PAGE_HOME;

else {
        try {
                // Get a reference to the home object

                ProcessRequestSessionHome prHome =
                        (ProcessRequestSessionHome)JNDIUtility.getHomeInterfaceByJNDILookup
                        (DHRSessionBeanLookupConstants.DHR_PROCESS_REQUEST_LOOKUP,
                        ProcessRequestSessionHome.class);
                ProcessRequestSession remote = prHome.create();

                // processPayment calls methods to generate reports, create a link,
                // produce the email notification, mark the request complete and display it.

                // Use the Session to track the number of unsuccessful authorization attempts.
                // Get it from the session, increment it by 1, and then save it back.

                byteNum = numAttempts.byteValue();
                byteNum++;
                req = remote.processPayment (req, byteNum, maxAttempts);

                // Invalidate the session to prevent user from
                // hitting back button and getting charged again.
                request.getSession().invalidate();

                request.setAttribute (DHRServletConstants.DHRREQUEST, req);

                // Tell the browser to access the displayrequest.jsp page
                // using our custom link.
                response.sendRedirect (buildDisplayRequestURL (req));

        } catch (DHRCreditCardDeclinedException ccd) {

                // Store the error message that was thrown back from ProcessRequestSessionBean.java
                // to be displayed by paymententry.jsp.
                request.setAttribute (DHRServletConstants.USER_MESSAGE, ccd.getUserErrorDescription());

                // Store the request object, despite getting an exception above.
                session.setAttribute (DHRServletConstants.DHRREQUEST, ccd.getAffectedObject());


                // After exceeding the maximum number of attempts, blank the credit
                // card fields and reset the counter.

                if (byteNum >= maxAttempts)
                        byteNum = 0;

                session.setAttribute (DHRServletConstants.NUMATTEMPTS, new Byte (byteNum));
```

```
                        // Go back to the payment entry page.
                        returnPage = DHRServletConstants.PAGE_BILLING_INFO_ENTRY;
                } catch (DHRException e) {
                        request.getSession().invalidate();

                        //If we catch a report not generated exception, we need to send them
                        //to an alternate page which tells the customer the order will be
                        //completed at a later date
                        if (e.getDHRErrorNumber() ==
                                        DHRExceptionConstants.EX_REPORT_NOT_GENERATED) {
                                request.setAttribute(DHRServletConstants.DHRREQUEST,
                                        (DHRRequest)e.getAffectedObject());
                                returnPage = DHRServletConstants.PAGE_RPT_CREATE_ERR;
                        } else {
                                request.setAttribute (DHRServletConstants.USER_MESSAGE,
                                                e.getUserErrorDescription());
                                returnPage = DHRServletConstants.PAGE_ERROR;
                        }

                } catch (Exception e) {
                        returnPage = processGeneralException (e, request, "processRequest");
                }
        }

return returnPage;
}


/**
 * General error handling method
 *
 * @param Exception the exception object
 * @param HttpServletRequest represents the request from the server
 * @param String location of the error (typically a method name)
 * @return String constant for the general error page
 */
private String processGeneralException (Exception e, HttpServletRequest request, String errorLocation) {

        try {
                e.fillInStackTrace();
                DHRUtility.logError ("RequestServlet", errorLocation, e.toString(), true);

        } catch (Exception ex) {}                       // If logging fails, then we're hosed.

        return DHRServletConstants.PAGE_ERROR;
}


/**
 * Returns an environment variable from the web descriptor file
 *
 * @param String the environment variable
 * @return String the value of the variable
 */

    private String getEnvVariable (String envVar) {
        String environmentVariable = "";

        try {
```

```
                // Get an environment variable from
                javax.naming.Context ctx = new javax.naming.InitialContext();
                javax.naming.Context env = (javax.naming.Context)ctx.lookup("java:comp/env");
                environmentVariable = (String)env.lookup (envVar);

            } catch (javax.naming.NamingException ne) {}

            return environmentVariable;
        }
}
```

5.3 PaymentEntry.java

CSS (Cascading Style Sheets) were used heavily throughout this presentation layer, rather than HTML tables, so that we could achieve an extremely high level of compliance with the W3C's (World Wide Web Consortium) published accessibility guidelines. This initiative encourages a high degree of usability for people with disabilities.

Some of the "pretty" formatting has been removed to prevent word wrapping, which affects the readability of the JSP in Word.

```
<%@ page import="java.util.GregorianCalendar" %>
<%@ page import="java.util.TimeZone" %>
<%@ page import="java.util.Locale" %>
<%@ page import="us.ky.state.kytc.util.CreditCard" %>
<%@ page import="java.text.NumberFormat" %>
<%@ include file="/includes/import.inc" %>
<%@ taglib prefix="ps" uri="http://www.ky.gov/HTTPPrefixSelectionLibrary"%>

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

<html lang="en">
<head>
    <title>DHR.KY.GOV - Payment Entry</title>

    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <meta name="robots" content="NONE">

    <META name="GENERATOR" content="IBM WebSphere Studio">
    <link rel="stylesheet" href="<ps:getPrefix prefix="HTTPS" />stylesheets/styles.css" type="text/css">
    <style type="text/css">
        @import url(<ps:getPrefix prefix="HTTPS" />stylesheets/styles_ie.css);
    </style>

    <script language="JavaScript" type="text/javascript">
    <!--
        function verifyForm() {

            // ^ = start at beginning of string
            // [0-9] = accept digits only
            // 8,20 = min length of 8, max of 20
            var REGEXP_CREDIT_CARD = /^[0-9]{8,20}/;        // CC# must be all numbers and 8-20 digits.
            var REGEXP_ZIP         = /^[0-9]{5}/;           // Zip code must be all numbers and length of 5.
            var REGEXP_NAME    = /[^0-9]/;                  // Name can contain no numbers.
            var REGEXP_EMAIL   = /^\w+[\w-\.]+@([\w\-]+\.){1,}[a-zA-Z]{2,4}$/;

            var errMsg = "";                                // An empty string means there are no errors.
```

```
var name = document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_NAME%>.value;
var city = document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_CITY%>.value;
var zip = document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_ZIP%>.value;
var billing1 =
        document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_BILLING_ADDRESS1
        %>.value;
var billing2 =
        document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_BILLING_ADDRESS2
        %>.value;
var acctNum =
        document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_NUMBER%>.value;
var email = document.Form1.<%=DHRServletConstants.PARAM_EMAIL_ADDRESS%>.value;
var emailConfirmed =
        (document.Form1.<%=DHRServletConstants.PARAM_EMAIL_ADDRESS%>.value ==
        document.Form1.<%=DHRServletConstants.PARAM_EMAIL_ADDRESS_CONFIRMATION%
        >.value);

if (!REGEXP_CREDIT_CARD.test (acctNum)) {
        targetField =
                document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_NUMBER%>;
        errMsg = "** A valid credit card number is required. **";

} else if (!REGEXP_NAME.test (name)) {
        targetField =
                document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_NAME%>;
        errMsg = "** Name is required. **";

} else if (!REGEXP_EMAIL.test (email)) {
        targetField = document.Form1.<%=DHRServletConstants.PARAM_EMAIL_ADDRESS%>;
        errMsg = "** Email address is not formatted properly or is blank. **";

} else if (!emailConfirmed) {
        targetField =
            document.Form1.<%=DHRServletConstants.PARAM_EMAIL_ADDRESS_CONFIRMATI
            ON%>;
        errMsg = "** Email and confirmation email do not match. **";

} else if (billing1 + billing2 == "") {
        targetField =
            document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_BILLING_ADDR
            ESS1%>;
        errMsg = "** Billing Address is required. **";

} else if (city == "") {
        targetField = document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_CITY%>;
        errMsg = "** City is required. **";

} else if (!REGEXP_ZIP.test (zip)) {
        targetField = document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_ZIP%>;
        errMsg = "** A 5-digit zip code is required. **";
}


// If the error message string contains text, an error was found.

if (errMsg != "") {
        alert (errMsg);
        targetField.focus();
        targetField.select();
```

```
                }

                return errMsg == "";
            } //-->
        </script>
</head>

<!-- Make the first textbox get the focus. -->
<body onload="document.Form1.<%=DHRServletConstants.PARAM_CREDIT_CARD_NUMBER%>.focus();"
bgcolor='#FFFFFF' text='#000000' vlink='#0000CC'>

<body bgcolor="#FFFFFF" text="#000000" vlink="#0000CC">

    <div id ="main">

        <%@ include file="/includes/sheader.inc" %>

        <div id="docbody">
        <%@ include file="/includes/smenu.inc" %>

        <div id = "contentarea" align="left">

        <!—Content starts here -->
        <DIV id = "pagelabel">
            PAYMENT ENTRY
            <img alt="" src="<ps:getPrefix prefix="HTTPS" />images/1px_black.gif" width="575" height="1"
                class="floatright">
        </DIV>
        <BR>
        <%
            DHRRequest req = (DHRRequest) session.getAttribute (DHRServletConstants.DHRREQUEST);
            CreditCard cc = req.getCreditCardInfo();

            String displayMessage = null;

            DisplayMessage =
                (String)request.getAttribute (us.ky.state.kytc.dhr.request.servlet.DHRServletConstants.USER_ME
                SSAGE);
            displayMessage = displayMessage==null?"":displayMessage + "<BR><BR>";

            Byte numAttempts = (Byte)request.getAttribute (DHRServletConstants.NUMATTEMPTS);
        %>
        <DIV ID = 'errordescription3'><%=displayMessage%></DIV>
            <DIV><SPAN class='olabel'>Please enter the payment information.</SPAN></DIV>

            <!-- FORM starts here -->

            <FORM NAME="Form1" METHOD="post" onsubmit="return verifyForm();" action="RS">

            <%
                float totalCost = req.calculateTotal();
                NumberFormat nf = NumberFormat.getCurrencyInstance();
            %>

            <DIV class='divrow'>
                <DIV class='col35'>
                <SPAN class='blueb'>
                    Total cost:
                </SPAN>
```

```
        </DIV>
        <DIV class='col65'>
            <SPAN class='blueb'>
                <%=nf.format(totalCost)%>
            </SPAN>
        </DIV>
</DIV>
<BR>

<div class='divrow'>
<DIV class='col35'><SPAN class='blueb'>Credit Card Number:</SPAN></DIV>
        <DIV class='col65'>

        <%
            out.print ("<INPUT NAME = '" + DHRServletConstants.PARAM_CREDIT_CARD_NUMBER +
                "' maxlength='20' size='20' " + "value='" + (cc.getAccountNumber() == null ? "" :
                cc.getAccountNumber()) + "'>");
        %>
        </DIV>
</DIV>

<div class='divrow'>
<DIV class='col35'><SPAN class='blueb'>Expiration Date:</SPAN></DIV>
<DIV class='col65'>

<SELECT
NAME="<%=DHRServletConstants.PARAM_CREDIT_CARD_EXPIRATION_MONTH%>"
SIZE="1">

<%
// The current year must be computed for use by both the month and year
// sections below.

GregorianCalendar cal = new GregorianCalendar();
int curYear = cal.get (cal.YEAR);

// Output an option group.  The expiration month that exists in the
// credit card object will be the default month in the dropdown.

out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("01") ? ">" : " SELECTED>") + "01 - January");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("02") ? ">" : " SELECTED>") + "02 - February");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("03") ? ">" : " SELECTED>") + "03 - March");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("04") ? ">" : " SELECTED>") + "04 - April");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("05") ? ">" : " SELECTED>") + "05 - May");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("06") ? ">" : " SELECTED>") + "06 - June");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("07") ? ">" : " SELECTED>") + "07 - July");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("08") ? ">" : " SELECTED>") + "08 - August");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("09") ? ">" : " SELECTED>") + "09 - September");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("10") ? ">" : " SELECTED>") + "10 - October");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("11") ? ">" : " SELECTED>") + "11 - November");
out.print ("<OPTION" + (!cc.getExpirationMonth().equals ("12") ? ">" : " SELECTED>") + "12 - December");
%>
</SELECT>

<SELECT NAME="<%=DHRServletConstants.PARAM_CREDIT_CARD_EXPIRATION_YEAR%>"
        SIZE="1">
<%
// Convert the selected year String to an int so I can use it in the FOR loop below.
String test = cc.getExpirationYear();
```

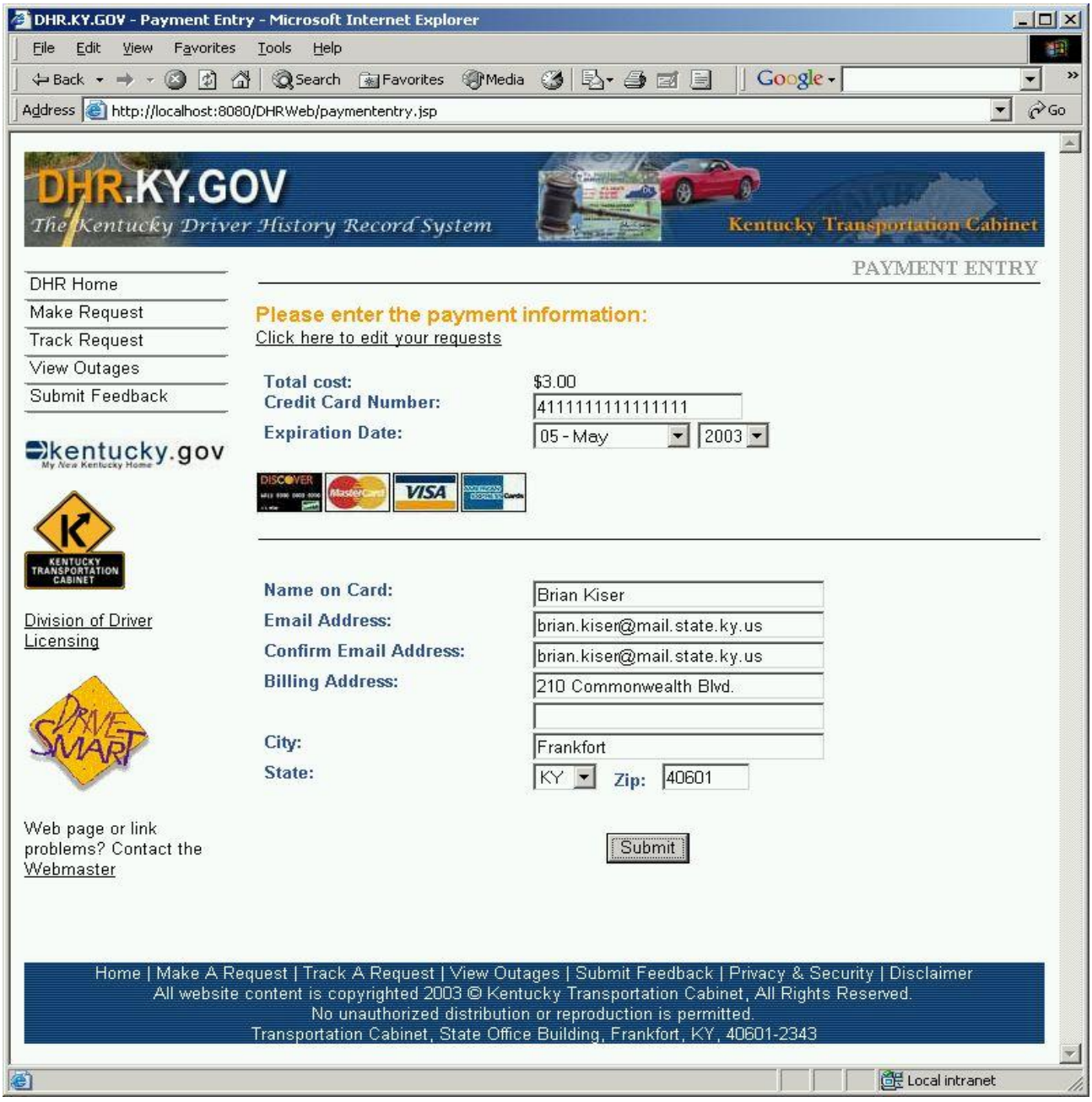
```
int selectedYear = new Integer (test).intValue();    // Convert Integer to int value.

// Display next 5 years starting from the current year (which happens
// to be the default).

for (int i = curYear; i < curYear + DHRServletConstants.PARAM_NUM_EXPIRATION_YEARS_TO_SHOW;
        i++)
        out.println (i != selectedYear ? "<OPTION>" + i : "<OPTION SELECTED>" + selectedYear);
%>

</SELECT>
</DIV>
</DIV>
<br>
<img alt="Discover, MasterCard, Visa and American Express accepted" src="<ps:getPrefix prefix="HTTPS"
        />images/cc.jpg">
<br><br>
<img alt="" src="<ps:getPrefix prefix="HTTPS" />images/1px_black.gif" width="575" height="1"
        class="floatright">
<br><br>
<div class='divrow'>
<DIV class='col35'><SPAN class='blueb'>Name on Card:</SPAN></DIV>
<DIV class='col65'>

<% out.print ("<INPUT NAME = '" + DHRServletConstants.PARAM_CREDIT_CARD_NAME + "'
        maxlength='20' size='30' " + "value='" + (cc.getNameOnCard() == null ? "" : cc.getNameOnCard()) +
        "'>");%>
</DIV></DIV>

<div class='divrow'>
<DIV class='col35'><SPAN class='blueb'>Email Address:</SPAN></DIV>
<DIV class='col65'>
        <% out.print ("<INPUT NAME = '" + DHRServletConstants.PARAM_EMAIL_ADDRESS + "'
        maxlength='50' size='30' " + "value='" + (req.getEmailAddress() == null ? "" : req.getEmailAddress()) +
        "'>");%>
</DIV></DIV>

<div class='divrow'>
        <DIV class='col35'><SPAN class='blueb'>Confirm Email Address:</SPAN></DIV>
        <DIV class='col65'>

<% out.print ("<INPUT NAME = '" + DHRServletConstants.PARAM_EMAIL_ADDRESS_CONFIRMATION
        + "' maxlength='50' size='30' " +     "value='" + (req.getConfirmEmailAddress() == null ? "" :
        req.getConfirmEmailAddress()) + "'>");%>
</DIV></DIV>

<div class='divrow'>
        <DIV class='col35'><SPAN class='blueb'>Billing Address:</SPAN></DIV>
        <DIV class='col65'>

        <% out.print ("<INPUT NAME = '" +
        DHRServletConstants.PARAM_CREDIT_CARD_BILLING_ADDRESS1 + "'
        maxlength='50' size='30' " + "value='" + (cc.getBillingAddress1() == null ? "" : cc.getBillingAddress1())
        + "'>");%>

        <% out.print ("<INPUT NAME = '" +
                DHRServletConstants.PARAM_CREDIT_CARD_BILLING_ADDRESS2 + "'
                maxlength='50' size='30' " + "value='" + (cc.getBillingAddress2() == null ? "" :
                cc.getBillingAddress2()) + "'>");%>
```

```
        </DIV></DIV>

<div class='divrow'>
<DIV class='col35'><SPAN class='blueb'>City:</SPAN></DIV>
<DIV class='col65'>

        <% out.print ("<INPUT NAME = '" + DHRServletConstants.PARAM_CREDIT_CARD_CITY + "'
                maxlength='30' size='30' " + "value='" + (cc.getCity() == null ? "" : cc.getCity()) + "'>");%>
</DIV></DIV>

<div class='divrow'>
        <DIV class='col35'><SPAN class='blueb'>State:</SPAN></DIV>
        <DIV class='col65'><SELECT
                NAME="<%=DHRServletConstants.PARAM_CREDIT_CARD_STATE%>"
                size='1'>
<%
out.print ("<OPTION" + (!cc.getState().equals ("AL") ? ">" : " SELECTED>") + "AL");
out.print ("<OPTION" + (!cc.getState().equals ("AK") ? ">" : " SELECTED>") + "AK");
out.print ("<OPTION" + (!cc.getState().equals ("AZ") ? ">" : " SELECTED>") + "AZ");
out.print ("<OPTION" + (!cc.getState().equals ("AR") ? ">" : " SELECTED>") + "AR");
out.print ("<OPTION" + (!cc.getState().equals ("CA") ? ">" : " SELECTED>") + "CA");
out.print ("<OPTION" + (!cc.getState().equals ("CO") ? ">" : " SELECTED>") + "CO");
out.print ("<OPTION" + (!cc.getState().equals ("CT") ? ">" : " SELECTED>") + "CT");
out.print ("<OPTION" + (!cc.getState().equals ("DE") ? ">" : " SELECTED>") + "DE");
out.print ("<OPTION" + (!cc.getState().equals ("FL") ? ">" : " SELECTED>") + "FL");
out.print ("<OPTION" + (!cc.getState().equals ("GA") ? ">" : " SELECTED>") + "GA");
out.print ("<OPTION" + (!cc.getState().equals ("HA") ? ">" : " SELECTED>") + "HA");
out.print ("<OPTION" + (!cc.getState().equals ("IA") ? ">" : " SELECTED>") + "IA");
out.print ("<OPTION" + (!cc.getState().equals ("ID") ? ">" : " SELECTED>") + "ID");
out.print ("<OPTION" + (!cc.getState().equals ("IL") ? ">" : " SELECTED>") + "IL");
out.print ("<OPTION" + (!cc.getState().equals ("IN") ? ">" : " SELECTED>") + "IN");
out.print ("<OPTION" + (!cc.getState().equals ("IO") ? ">" : " SELECTED>") + "IO");
out.print ("<OPTION" + (!cc.getState().equals ("KS") ? ">" : " SELECTED>") + "KS");
out.print ("<OPTION" + (!cc.getState().equals ("KY") ? ">" : " SELECTED>") + "KY");
out.print ("<OPTION" + (!cc.getState().equals ("LA") ? ">" : " SELECTED>") + "LA");
out.print ("<OPTION" + (!cc.getState().equals ("MN") ? ">" : " SELECTED>") + "MN");
out.print ("<OPTION" + (!cc.getState().equals ("MD") ? ">" : " SELECTED>") + "MD");
out.print ("<OPTION" + (!cc.getState().equals ("MA") ? ">" : " SELECTED>") + "MA");
out.print ("<OPTION" + (!cc.getState().equals ("MI") ? ">" : " SELECTED>") + "MI");
out.print ("<OPTION" + (!cc.getState().equals ("MN") ? ">" : " SELECTED>") + "MN");
out.print ("<OPTION" + (!cc.getState().equals ("MS") ? ">" : " SELECTED>") + "MS");
out.print ("<OPTION" + (!cc.getState().equals ("MO") ? ">" : " SELECTED>") + "MO");
out.print ("<OPTION" + (!cc.getState().equals ("NE") ? ">" : " SELECTED>") + "NE");
out.print ("<OPTION" + (!cc.getState().equals ("NV") ? ">" : " SELECTED>") + "NV");
out.print ("<OPTION" + (!cc.getState().equals ("NH") ? ">" : " SELECTED>") + "NH");
out.print ("<OPTION" + (!cc.getState().equals ("NJ") ? ">" : " SELECTED>") + "NJ");
out.print ("<OPTION" + (!cc.getState().equals ("NM") ? ">" : " SELECTED>") + "NM");
out.print ("<OPTION" + (!cc.getState().equals ("NY") ? ">" : " SELECTED>") + "NY");
out.print ("<OPTION" + (!cc.getState().equals ("NC") ? ">" : " SELECTED>") + "NC");
out.print ("<OPTION" + (!cc.getState().equals ("ND") ? ">" : " SELECTED>") + "ND");
out.print ("<OPTION" + (!cc.getState().equals ("OH") ? ">" : " SELECTED>") + "OH");
out.print ("<OPTION" + (!cc.getState().equals ("OK") ? ">" : " SELECTED>") + "OK");
out.print ("<OPTION" + (!cc.getState().equals ("OR") ? ">" : " SELECTED>") + "OR");
out.print ("<OPTION" + (!cc.getState().equals ("PN") ? ">" : " SELECTED>") + "PN");
out.print ("<OPTION" + (!cc.getState().equals ("RI") ? ">" : " SELECTED>") + "RI");
out.print ("<OPTION" + (!cc.getState().equals ("SC") ? ">" : " SELECTED>") + "SC");
out.print ("<OPTION" + (!cc.getState().equals ("SD") ? ">" : " SELECTED>") + "SD");
out.print ("<OPTION" + (!cc.getState().equals ("TN") ? ">" : " SELECTED>") + "TN");
out.print ("<OPTION" + (!cc.getState().equals ("TX") ? ">" : " SELECTED>") + "TX");
```

```
                out.print ("<OPTION" + (!cc.getState().equals ("UT") ? ">" : " SELECTED>") + "UT");
                out.print ("<OPTION" + (!cc.getState().equals ("VT") ? ">" : " SELECTED>") + "VT");
                out.print ("<OPTION" + (!cc.getState().equals ("VA") ? ">" : " SELECTED>") + "VA");
                out.print ("<OPTION" + (!cc.getState().equals ("WA") ? ">" : " SELECTED>") + "WA");
                out.print ("<OPTION" + (!cc.getState().equals ("WV") ? ">" : " SELECTED>") + "WV");
                out.print ("<OPTION" + (!cc.getState().equals ("WI") ? ">" : " SELECTED>") + "WI");
                out.print ("<OPTION" + (!cc.getState().equals ("WY") ? ">" : " SELECTED>") + "WY");
                %>
                </SELECT>
                  <SPAN class='blueb'>Zip:</SPAN>  

                <% out.print ("<INPUT NAME = '" + DHRServletConstants.PARAM_CREDIT_CARD_ZIP + "' maxlength='5'
                        size='5' " + "value='" + (cc.getZip() == null ? "" : cc.getZip()) + "'>");%>

                </DIV></DIV>

                <BR><BR>

                <INPUT type="hidden" NAME="AC"
                        value="<%=DHRServletConstants.AC_PAYMENT_ENTRY_SUBMIT%>">
                <CENTER><INPUT type="submit" value="Submit"></CENTER>
                </FORM>

                <FORM NAME="Form2" METHOD="post" action="<ps:getPrefix prefix="HTTP" />RS">
                        <INPUT type="hidden" NAME="AC"
                                value="<%=DHRServletConstants.AC_CANCEL_REQUEST%>">
                        <CENTER><INPUT type="submit" value="Cancel Order and Return to Home Page"></CENTER>
                </FORM>

                <BR><BR>
                <!-- contentarea ends here -->
                </div>

                <%@ include file="/includes/sfooter.inc" %>
                <!-- docbody -->
                </div>

                <!-- main -->
        </DIV>
</BODY>
</HTML>
```

6.0 Appendix A

6.1 Payment Entry



6.2 Payment Confirmation