# How to Embed HTML Form (generated by HTML Form Entry module) in a custom module

This guide is targeted for openmrs developers who are looking to Embed HTML Form Entry module generated Form into a Separate OpenMRS module view/jsp

Something look like this: -->

Actually we need only two files to get things done:

1. HtmlFormEntryController.java : https://github.com/openmrs/openmrs-module-htmlformentry/blob/master/omod/src/main/java/org/openmrs/module/htmlformentry/web/controll/HtmlFormEntryController.java

   But to backup the HTML form entry dependent files, you have to add the HTML form entry API maven dependency into the POM

```
    <properties>
      ...
        <htmlformentryModuleVersion>2.1.1</htmlformentryModuleVersion>
      ...
    </properties>

        <dependency>
            <groupId>org.openmrs.module</groupId>
            <artifactId>htmlformentry-api</artifactId>
            <version>${htmlformentryModuleVersion}</version>
            <scope>provided</scope>
        </dependency>
```

2. htmlFormEntry.jsp : https://github.com/openmrs/openmrs-module-htmlformentry/blob/master/omod/src/main/webapp/htmlFormEntry.jsp

```
     Notice the  ${command.htmlToDisplay}  <%-- THIS is where the Magic happens: Rendering the HTML form and
displaying on this particular Div --%>
```

```
package org.openmrs.module.patientnarratives.web.controller;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.util.Collections;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.openmrs.Encounter;
import org.openmrs.Form;
import org.openmrs.Patient;
import org.openmrs.api.context.Context;
import org.openmrs.module.htmlformentry.BadFormDesignException; // <------- these imports will be taken after
you update the maven dependencies in POM
import org.openmrs.module.htmlformentry.FormEntryContext.Mode;
import org.openmrs.module.htmlformentry.FormEntrySession;
import org.openmrs.module.htmlformentry.FormSubmissionError;
import org.openmrs.module.htmlformentry.HtmlForm;
import org.openmrs.module.htmlformentry.HtmlFormEntryUtil;
import org.openmrs.module.htmlformentry.ValidationException;
import org.openmrs.util.OpenmrsUtil;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
import org.springframework.validation.Errors;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;
```

```
import org.springframework.web.servlet.view.RedirectView;
/**
 * The controller for entering/viewing a form.
 * <p/>
 * Handles {@code htmlFormEntry.form} requests. Renders view {@code htmlFormEntry.jsp}.
 * <p/>
 * TODO: This has a bit too much logic in the onSubmit method. Move that into the FormEntrySession.
 */
@Controller
public class HtmlFormEntryEmbedController {
    protected final Log log = LogFactory.getLog(getClass());
//    public final static String closeDialogView = "/module/htmlformentry/closeDialog";
    public final static String FORM_IN_PROGRESS_KEY = "HTML_FORM_IN_PROGRESS_KEY";
    public final static String FORM_IN_PROGRESS_VALUE = "HTML_FORM_IN_PROGRESS_VALUE";
    public final static String FORM_PATH = "/module/patientnarratives/htmlFormEntry";
    @RequestMapping(method=RequestMethod.GET, value=FORM_PATH)
    public void showForm() {
        // Intentionally blank. All work is done in the getFormEntrySession method
    }
    @ModelAttribute("command")
    public FormEntrySession getFormEntrySession(HttpServletRequest request,
                                                // @RequestParam doesn't pick up query parameters (in the url)
in a POST, so I'm handling encounterId, modeParam, and which specially
                                                /*@RequestParam(value="mode", required=false) String modeParam,
*/
                                                /*@RequestParam(value="encounterId", required=false) Integer
encounterId,*/
                                                /*@RequestParam(value="which", required=false) String which,*/
                                                @RequestParam(value="patientId", required=false) Integer
patientId,
                                                /*@RequestParam(value="personId", required=false) Integer
personId,*/
                                                @RequestParam(value="formId", required=false) Integer formId,
                                                @RequestParam(value="htmlformId", required=false) Integer
htmlFormId,
                                                @RequestParam(value="returnUrl", required=false) String
returnUrl,
                                                @RequestParam(value="formModifiedTimestamp", required=false)
Long formModifiedTimestamp,
                                                @RequestParam(value="encounterModifiedTimestamp",
required=false) Long encounterModifiedTimestamp,
                                                @RequestParam(value="hasChangedInd", required=false) String
hasChangedInd) throws Exception {
        long ts = System.currentTimeMillis();
        Mode mode = Mode.VIEW;
        Integer personId = null;
        patientId   = 2;
        formId = 3; // <-----------------------this is where you need to mention the FORM ID or HTML Form id in
below variable
        htmlFormId = 1; //<--------------------|
         //Integer.parseInt(request.getParameter("formId"));
//        if (StringUtils.hasText(request.getParameter("personId"))) {
//            personId = Integer.valueOf(request.getParameter("personId"));
//        }
        String modeParam = request.getParameter("mode");
        if ("enter".equalsIgnoreCase(modeParam)) {
            mode = Mode.ENTER;
        }
        else if ("edit".equalsIgnoreCase(modeParam)) {
            mode = Mode.EDIT;
        }
        Patient patient = null;
        Encounter encounter = null;
        Form form = null;
        HtmlForm htmlForm = null;
        if (StringUtils.hasText(request.getParameter("encounterId"))) {
            Integer encounterId = Integer.valueOf(request.getParameter("encounterId"));
            encounter = Context.getEncounterService().getEncounter(encounterId);
            if (encounter == null)
                throw new IllegalArgumentException("No encounter with id=" + encounterId);
            patient = encounter.getPatient();
```

```
            patientId = patient.getPatientId();
            personId = patient.getPersonId();
            if (formId != null) { // I think formId is allowed to differ from encounter.form.id because of
HtmlFormFlowsheet
                form = Context.getFormService().getForm(formId);
                htmlForm = HtmlFormEntryUtil.getService().getHtmlFormByForm(form);
                if (htmlForm == null)
                    throw new IllegalArgumentException("No HtmlForm associated with formId " + formId);
            } else {
                form = encounter.getForm();
                htmlForm = HtmlFormEntryUtil.getService().getHtmlFormByForm(encounter.getForm());
                if (htmlForm == null)
                    throw new IllegalArgumentException("The form for the specified encounter (" + encounter.
getForm() + ") does not have an HtmlForm associated with it");
            }
        } else { // no encounter specified
            // get person from patientId/personId (register module uses patientId, htmlformentry uses personId)
            if (patientId != null) {
                personId = patientId;
            }
            if (personId != null) {
                patient = Context.getPatientService().getPatient(personId);
            }
            // determine form
            if (htmlFormId != null) {
                htmlForm = HtmlFormEntryUtil.getService().getHtmlForm(htmlFormId);
            } else if (formId != null) {
                form = Context.getFormService().getForm(formId);
                htmlForm = HtmlFormEntryUtil.getService().getHtmlFormByForm(form);
            }
            if (htmlForm == null) {
                throw new IllegalArgumentException("You must specify either an htmlFormId or a formId for a
valid html form");
            }
            String which = request.getParameter("which");
            if (StringUtils.hasText(which)) {
                if (patient == null)
                    throw new IllegalArgumentException("Cannot specify 'which' without specifying a person
/patient");
                List<Encounter> encs = Context.getEncounterService().getEncounters(patient, null, null, null,
Collections.singleton(form), null, null, false);
                if (which.equals("first")) {
                    encounter = encs.get(0);
                } else if (which.equals("last")) {
                    encounter = encs.get(encs.size() - 1);
                } else {
                    throw new IllegalArgumentException("which must be 'first' or 'last'");
                }
            }
        }
        if (mode != Mode.ENTER && patient == null)
            throw new IllegalArgumentException("No patient with id of personId=" + personId + " or patientId="
+ patientId);
        FormEntrySession session = null;
        if (mode == Mode.ENTER && patient == null) {
            patient = new Patient();
        }
        if (encounter != null) {
            session = new FormEntrySession(patient, encounter, mode, htmlForm, request.getSession());
        }
        else {
            session = new FormEntrySession(patient, htmlForm, request.getSession());
        }
        if (StringUtils.hasText(returnUrl)) {
            session.setReturnUrl(returnUrl);
        }
        // Since we're not using a sessionForm, we need to check for the case where the underlying form was
modified while a user was filling a form out
        if (formModifiedTimestamp != null) {
            if (!OpenmrsUtil.nullSafeEquals(formModifiedTimestamp, session.getFormModifiedTimestamp())) {
                throw new RuntimeException(Context.getMessageSourceService().getMessage("htmlformentry.error.
```

```
formModifiedBeforeSubmission"));
            }
        }
        // Since we're not using a sessionForm, we need to make sure this encounter hasn't been modified since
the user opened it
        if (encounter != null) {
            if (encounterModifiedTimestamp != null && !OpenmrsUtil.nullSafeEquals(encounterModifiedTimestamp,
session.getEncounterModifiedTimestamp())) {
                throw new RuntimeException(Context.getMessageSourceService().getMessage("htmlformentry.error.
encounterModifiedBeforeSubmission"));
            }
        }
        if (hasChangedInd != null) session.setHasChangedInd(hasChangedInd);
        Context.setVolatileUserData(FORM_IN_PROGRESS_KEY, session);
        log.info("Took " + (System.currentTimeMillis() - ts) + " ms");
        return session;
    }
    /*
     * I'm using a return type of ModelAndView so I can use RedirectView rather than "redirect:" and preserve
the fact that
     * returnUrl values from the pre-annotated-controller days will have the context path already
     */
    @RequestMapping(method=RequestMethod.POST, value=FORM_PATH)
    public ModelAndView handleSubmit(@ModelAttribute("command") FormEntrySession session,
                                     Errors errors,
                                     HttpServletRequest request,
                                     Model model) throws Exception {
        try {
            List<FormSubmissionError> validationErrors = session.getSubmissionController().validateSubmission
(session.getContext(), request);
            if (validationErrors != null && validationErrors.size() > 0) {
                errors.reject("Fix errors");
            }
        } catch (Exception ex) {
            log.error("Exception during form validation", ex);
            errors.reject("Exception during form validation, see log for more details: " + ex);
        }
        if (errors.hasErrors()) {
            return new ModelAndView(FORM_PATH, "command", session);
        }
        // no form validation errors, proceed with submission
        session.prepareForSubmit();
        if (session.getContext().getMode() == Mode.ENTER && session.hasPatientTag() && session.getPatient() ==
null
                && (session.getSubmissionActions().getPersonsToCreate() == null || session.
getSubmissionActions().getPersonsToCreate().size() == 0))
            throw new IllegalArgumentException("This form is not going to create an Patient");
        if (session.getContext().getMode() == Mode.ENTER && session.hasEncouterTag() && (session.
getSubmissionActions().getEncountersToCreate() == null || session.getSubmissionActions().
getEncountersToCreate().size() == 0))
            throw new IllegalArgumentException("This form is not going to create an encounter");
        try {
            session.getSubmissionController().handleFormSubmission(session, request);
            HtmlFormEntryUtil.getService().applyActions(session);
            String successView = session.getReturnUrlWithParameters();
            if (successView == null)
                successView = request.getContextPath() + "/patientDashboard.form" + getQueryPrameters(request,
session);
            if (StringUtils.hasText(request.getParameter("closeAfterSubmission"))) {
//                return new ModelAndView(closeDialogView, "dialogToClose", request.getParameter
("closeAfterSubmission"));
            } else {
                return new ModelAndView(new RedirectView(successView));
            }
        } catch (ValidationException ex) {
            log.error("Invalid input:", ex);
            errors.reject(ex.getMessage());
        } catch (BadFormDesignException ex) {
            log.error("Bad Form Design:", ex);
            errors.reject(ex.getMessage());
        } catch (Exception ex) {
```

```
            log.error("Exception trying to submit form", ex);
            StringWriter sw = new StringWriter();
            ex.printStackTrace(new PrintWriter(sw));
            errors.reject("Exception! " + ex.getMessage() + "<br/>" + sw.toString());
        }
        // if we get here it's because we caught an error trying to submit/apply
        return new ModelAndView(FORM_PATH, "command", session);
    }
    protected String getQueryPrameters(HttpServletRequest request, FormEntrySession formEntrySession) {
        return "?patientId=" + formEntrySession.getPatient().getPersonId();
    }
}
```

```
<%@ include file="/WEB-INF/template/include.jsp" %>

<openmrs:htmlInclude file="${pageContext.request.contextPath}/moduleResources/patientnarratives/css/styles.css"
/>

<%@ page import="net.tanesha.recaptcha.ReCaptcha" %>
<%@ page import="net.tanesha.recaptcha.ReCaptchaFactory" %>

<c:set var="OPENMRS_DO_NOT_SHOW_PATIENT_SET" scope="request" value="true"/>
<c:set var="pageFragment" value="${param.pageFragment != null && param.pageFragment}"/>
<c:set var="inPopup" value="${pageFragment || (param.inPopup != null && param.inPopup)}"/>

<c:if test="${not pageFragment}">
    <c:set var="DO_NOT_INCLUDE_JQUERY" value="true"/>
    <c:choose>
        <c:when test="${inPopup}">
            <%@ include file="/WEB-INF/template/headerMinimal.jsp" %>
        </c:when>
        <c:otherwise>
            <%@ include file="/WEB-INF/template/header.jsp" %>
        </c:otherwise>
    </c:choose>

    <openmrs:htmlInclude file="/dwr/engine.js" />
    <openmrs:htmlInclude file="/dwr/util.js" />
    <openmrs:htmlInclude file="/dwr/interface/DWRHtmlFormEntryService.js" />
    <openmrs:htmlInclude file="/moduleResources/htmlformentry/htmlFormEntry.js" />
    <openmrs:htmlInclude file="/moduleResources/htmlformentry/htmlFormEntry.css" />
    <openmrs:htmlInclude file="/moduleResources/htmlformentry/jquery-ui-1.8.17.custom.css" />
    <openmrs:htmlInclude file="/moduleResources/htmlformentry/jquery-1.4.2.min.js" />
    <script type="text/javascript">
        $j = jQuery.noConflict();
    </script>
    <openmrs:htmlInclude file="/moduleResources/htmlformentry/jquery-ui-1.8.17.custom.min.js" />
</c:if>

<script type="text/javascript">
    var propertyAccessorInfo = new Array();

    // individual forms can define their own functions to execute before a form validation or submission by
adding them to these lists
    // if any function returns false, no further functions are called and the validation or submission is
cancelled
    var beforeValidation = new Array();     // a list of functions that will be executed before the validation
of a form
    var beforeSubmit = new Array();         // a list of functions that will be executed before the submission
of a form

    // boolean to track whether or not jquery document ready function fired
    var initInd = true;

   // booleans used to track whether we are in the process of submitted or discarding a formk
   var isSubmittingInd = false;
   var isDiscardingInd = false;

    $j(document).ready(function() {
```

```
        $j('#deleteButton').click(function() {
            // display a "deleting form" message
            $j('#confirmDeleteFormPopup').children("center").html('<spring:message code="htmlformentry.
deletingForm"/>');

            // do the post that does the actual delete
            $j.post("<c:url value="/module/htmlformentry/deleteEncounter.form"/>",
                {     encounterId: "${command.encounter.encounterId}",
                    htmlFormId: "${command.htmlFormId}",
                    returnUrl: "${command.returnUrlWithParameters}",
                    reason: $j('#deleteReason').val()
                 },
                 function(data) {
                     var url = "${command.returnUrlWithParameters}";
                     if (url == null || url == "") {
                         url = "${pageContext.request.contextPath}/patientDashboard.form?patientId=${command.
patient.patientId}";
                     }
                     window.parent.location.href = url;
                 }
             );
        });

        // triggered whenever any input with toggleDim attribute is changed.  Currently, only supports
        // checkbox style inputs.
        $j('input[toggleDim]').change(function () {
            var target = $j(this).attr("toggleDim");
            if ($j(this).is(":checked")) {
                $j("#" + target + " :input").removeAttr('disabled');
                $j("#" + target).animate({opacity:1.0}, 0);
                restoreContainerInputs($j("#" + target));
            } else {
                $j("#" + target + " :input").attr('disabled', true);
                $j("#" + target).animate({opacity:0.5}, 100);
                clearContainerInputs($j("#" + target));
            }
        })
        .change();

        // triggered whenever any input with toggleHide attribute is changed.  Currently, only supports
        // checkbox style inputs.
        $j('input[toggleHide]').change(function () {
            var target = $j(this).attr("toggleHide");
            if ($j(this).is(":checked")) {
                $j("#" + target).fadeIn();
                restoreContainerInputs($j("#" + target));
            } else {
                $j("#" + target).hide();
                clearContainerInputs($j("#" + target));
            }
        })
        .change();

        // triggered whenever any input widget on the page is changed
           $j(':input').change(function () {
            $j(':input.has-changed-ind').val('true');
        });

        // warn user that his/her changes will be lost if he/she leaves the page
        $j(window).bind('beforeunload', function(){
            var hasChangedInd = $j(':input.has-changed-ind').val();
            if (hasChangedInd == 'true' && !isSubmittingInd && !isDiscardingInd) {
                return '<spring:message code="htmlformentry.loseChangesWarning"/>';
            }
        });

        // catch form submit button (not currently used)
        $j('form').submit(function() {
            isSubmittingInd = true;
            return true;
        });
```

```
        // catch when button with class submitButton is clicked (currently used)
        $j(':input.submitButton').click(function() {
            isSubmittingInd = true;
            return true;
        });

        // catch when discard link clicked
        $j('.html-form-entry-discard-changes').click(function() {
            isDiscardingInd = true;
            return true;
        });

        // indicates this function has completed
        initInd = false;

        //managing the id of the newly generated id's of dynamicAutocomplete widgets
        $j('div .dynamicAutocomplete').each(function(index) {
            var string=((this.id).split("_div",1))+"_hid";
            if(!$j('#'+string).attr('value'))
                $j('#'+this.id).data("count",0);
            else
                $j('#'+this.id).data("count",parseInt($j('#'+string).attr('value')));
            });
        //add button for dynamic autocomplete
        $j(':button.addConceptButton').click(function() {
                  var string=(this.id).replace("_button","");
                var conceptValue=$j('#'+string+'_hid').attr('value')
                if($j('#'+string).css('color')=='green'){
                    var    divId=string+"_div";
                     var spanid=string+'span_'+ $j('#'+divId).data("count");
                     var count= $j('#'+divId).data("count");
                     $j('#'+divId).data("count",++count);
                     $j('#'+string+'_hid').attr('value',$j('#'+divId).data("count"));
                     var hidId=spanid+'_hid';
                       var v='<span id="'+spanid+'"></br>'+$j('#'+string).val()+'<input id="'+hidId+'"  class="
autoCompleteHidden" type="hidden" name="'+hidId+'" value="'+conceptValue+'">';
                     var q='<input id="'+spanid+'_button" type="button" value="Remove" onClick="$j
(\'#'+spanid+'\').remove();openmrs.htmlformentry.refresh(this.id)"></span>';
                     $j('#'+divId).append(v+q);
                     $j('#'+string).val('');
            }
            });
    });

    // clear toggle container's inputs but saves the input values until form is submitted/validated in case the
user
    // re-clicks the trigger checkbox.  Note: These "saved" input values will be lost if the form fails
validation on submission.
    function clearContainerInputs($container) {
        if (!initInd) {
            $container.find('input:text, input:password, input:file, select, textarea').each( function() {
                $j(this).data('origVal',this.value);
                $j(this).val("");
            });
            $container.find('input:radio, input:checkbox').each( function() {
                if ($j(this).is(":checked")) {
                    $j(this).data('origState','checked');
                    $j(this).removeAttr("checked");
                } else {
                    $j(this).data('origState','unchecked');
                }
            });
        }
    }

    // restores toggle container's inputs from the last time the trigger checkbox was unchecked
    function restoreContainerInputs($container) {
        if (!initInd) {
            $container.find('input:text, input:password, input:file, select, textarea').each( function() {
                $j(this).val($j(this).data('origVal'));
```

```
        });
        $container.find('input:radio, input:checkbox').each( function() {
            if ($j(this).data('origState') == 'checked') {
                $j(this).attr("checked", "checked");
            } else {
                $j(this).removeAttr("checked");
            }
        });
    }
}

var tryingToSubmit = false;

function submitHtmlForm() {
    if (!tryingToSubmit) {
        tryingToSubmit = true;
        DWRHtmlFormEntryService.checkIfLoggedIn(checkIfLoggedInAndErrorsCallback);
    }
}

function findAndHighlightErrors(){
    /* see if there are error fields */
    var containError = false;
    var ary = $j(".autoCompleteHidden");
    $j.each(ary,function(index, value){
        if(value.value == "ERROR"){
            if(!containError){
                alert("<spring:message code='htmlformentry.error.autoCompleteAnswerNotValid'/>");
                var id = value.id;
                id = id.substring(0,id.length-4);
                $j("#"+id).focus();
            }
            containError=true;
        }
    });
    return containError;
}

function findOptionAutoCompleteErrors() {
    /* see if there are  errors in option fields */
    var containError = false;
    var ary = $j(".optionAutoCompleteHidden");
    $j.each(ary,function(index, value){
        if(value.value == "ERROR"){
            if(!containError){
                alert("<spring:message code='htmlformentry.error.autoCompleteOptionNotValid'/>");
                var id = value.id;
                id = id.substring(0,id.length-4);
                $j("#"+id).focus();
            }
            containError=true;
        }
    });
    return containError;
}

/*
    It seems the logic of  showAuthenticateDialog and
    findAndHighlightErrors should be in the same callback function.
    i.e. only authenticated user can see the error msg of
*/
function checkIfLoggedInAndErrorsCallback(isLoggedIn) {

    var state_beforeValidation=true;

    if (!isLoggedIn) {
        showAuthenticateDialog();
    }else{

        // first call any beforeValidation functions that may have been defined by the html form
        if (beforeValidation.length > 0){
```

```
            for (var i=0, l = beforeValidation.length; i < l; i++){
                if (state_beforeValidation){
                    var fncn=beforeValidation[i];
                    state_beforeValidation=eval(fncn);
                }
                else{
                    // forces the end of the loop
                    i=l;
                }
            }
        }

        // only do the validation if all the beforeValidationk functions returned "true"
        if (state_beforeValidation){
            var anyErrors = findAndHighlightErrors();
            var optionSelectErrors = findOptionAutoCompleteErrors();

            if (anyErrors || optionSelectErrors) {
                tryingToSubmit = false;
                return;
            }else{
                doSubmitHtmlForm();
            }
        }
    }
}

function showAuthenticateDialog() {
    $j('#passwordPopup').show();
    tryingToSubmit = false;
}

function loginThenSubmitHtmlForm() {

    $j('#passwordPopup').hide();
    var username = $j('#passwordPopupUsername').val();
    var password = $j('#passwordPopupPassword').val();
    $j('#passwordPopupUsername').val('');
    $j('#passwordPopupPassword').val('');
    DWRHtmlFormEntryService.authenticate(username, password, submitHtmlForm);
}

function doSubmitHtmlForm() {

    // first call any beforeSubmit functions that may have been defined by the form
    var state_beforeSubmit=true;
    if (beforeSubmit.length > 0){
        for (var i=0, l = beforeSubmit.length; i < l; i++){
            if (state_beforeSubmit){
                var fncn=beforeSubmit[i];
                state_beforeSubmit=fncn();
            }
            else{
                // forces the end of the loop
                i=l;
            }
        }
    }

    // only do the submit if all the beforeSubmit functions returned "true"
    if (state_beforeSubmit){
        var form = document.getElementById('htmlform');
        form.submit();
    }
    tryingToSubmit = false;
}

function handleDeleteButton() {
    $j('#confirmDeleteFormPopup').show();
}
```

```
    function cancelDeleteForm() {
        $j('#confirmDeleteFormPopup').hide();
    }


</script>

<%-- yes you see, these are not nessasary!..
<div id="htmlFormEntryBanner">--%>
    <%--<spring:message var="backMessage" code="htmlformentry.goBack"/>--%>
    <%--<c:if test="${!inPopup && (command.context.mode == 'ENTER' || command.context.mode == 'EDIT')}">--%>
        <%--<spring:message var="backMessage" code="htmlformentry.discard"/>--%>
    <%--</c:if>--%>
    <%--<div style="float: left" id="discardAndPrintDiv">--%>
        <%--<c:if test="${!inPopup}">--%>
            <%--<span id="discardLinkSpan"><a href="<c:choose><c:when test="${not empty command.
returnUrlWithParameters}">${command.returnUrlWithParameters}</c:when><c:otherwise>${pageContext.request.
contextPath}/patientDashboard.form?patientId=${command.patient.patientId}</c:otherwise></c:choose>" class="html-
form-entry-discard-changes">${backMessage}</a></span> | --%>
        <%--</c:if>--%>
        <%--<span id="printLinkSpan"><a href="javascript:window.print();"><spring:message code="htmlformentry.
print"/></a></span>  <br/>--%>
    <%--</div>--%>
    <%--<div style="float:right">--%>
        <%--<c:if test="${command.context.mode == 'VIEW'}">--%>
            <%--<c:if test="${!inPopup}">--%>
                <%--<openmrs:hasPrivilege privilege="Edit Encounters,Edit Observations">--%>
                    <%--<c:url var="editUrl" value="/module/htmlformentry/htmlFormEntry.form">--%>
                        <%--<c:forEach var="p" items="${param}">--%>
                            <%--<c:if test="${p.key != 'mode'}">--%>
                                <%--<c:param name="${p.key}" value="${p.value}"/>--%>
                            <%--</c:if>--%>
                        <%--</c:forEach>--%>
                        <%--<c:param name="mode" value="EDIT"/>--%>
                    <%--</c:url>--%>
                    <%--<a href="${editUrl}"><spring:message code="general.edit"/></a> |--%>
                <%--</openmrs:hasPrivilege>--%>
            <%--</c:if>--%>
            <%--<openmrs:hasPrivilege privilege="Delete Encounters,Delete Observations">--%>
                <%--<a onClick="handleDeleteButton()"><spring:message code="general.delete"/></a>--%>
                <%--<div id="confirmDeleteFormPopup" style="position: absolute; z-axis: 1; right: 0px;
background-color: #ffff00; border: 2px black solid; display: none; padding: 10px">--%>
                    <%--<center>--%>
                        <%--<spring:message code="htmlformentry.deleteReason"/>--%>
                        <%--<br/>--%>
                        <%--<textarea name="reason" id="deleteReason"></textarea>--%>
                        <%--<br/><br/>--%>
                        <%--<input type="button" value="<spring:message code="general.cancel"/>" onClick="
cancelDeleteForm()"/>--%>
                        <%--    --%>
                        <%--<input type="button" value="<spring:message code="general.delete"/>" id="
deleteButton"/>--%>
                    <%--</center>--%>
                <%--</div>--%>
            <%--</openmrs:hasPrivilege>--%>
        <%--</c:if>--%>
    <%--</div>--%>
    <%--<c:if test="${!inPopup}">--%>
        <%--<b>--%>
            <%--${command.patient.personName} |--%>
            <%--<c:choose>--%>
                <%--<c:when test="${not empty command.form}">--%>
                    <%--${command.form.name} (${command.form.encounterType.name})--%>
                <%--</c:when>--%>
                <%--<c:otherwise>--%>
                    <%--<c:if test="${not empty command.encounter}">--%>
                        <%--${command.encounter.form.name} (${command.encounter.encounterType.name})--%>
                    <%--</c:if>--%>
                <%--</c:otherwise> --%>
            <%--</c:choose>--%>
            <%---%>
```

```
            <%--|--%>
            <%--<c:if test="${not empty command.encounter}">--%>
                <%--<openmrs:formatDate date="${command.encounter.encounterDatetime}"/> | ${command.encounter.
location.name} --%>
            <%--</c:if>--%>
            <%--<c:if test="${empty command.encounter}">--%>
                <%--<spring:message code="htmlformentry.newForm"/>--%>
            <%--</c:if>--%>
        <%--</b>--%>
    <%--</c:if>--%>
<%--</div>--%>

<c:if test="${command.context.mode != 'VIEW'}">
    <spring:hasBindErrors name="command">
        <spring:message code="fix.error"/>
        <div class="error">
            <c:forEach items="${errors.allErrors}" var="error">
                <spring:message code="${error.code}" text="${error.code}"/><br/>
            </c:forEach>
        </div>
        <br />
    </spring:hasBindErrors>
</c:if>

<c:if test="${command.context.mode != 'VIEW'}">
    <form id="htmlform" method="post" onSubmit="submitHtmlForm(); return false;">
        <input type="hidden" name="personId" value="${ command.patient.personId }"/>
        <input type="hidden" name="htmlFormId" value="${ command.htmlFormId }"/>
        <input type="hidden" name="formModifiedTimestamp" value="${ command.formModifiedTimestamp }"/>
        <input type="hidden" name="encounterModifiedTimestamp" value="${ command.encounterModifiedTimestamp }"/>
        <c:if test="${ not empty command.encounter }">
            <input type="hidden" name="encounterId" value="${ command.encounter.encounterId }"/>
        </c:if>
        <input type="hidden" name="closeAfterSubmission" value="${param.closeAfterSubmission}"/>
        <input type="hidden" name="hasChangedInd" class="has-changed-ind" value="${ command.hasChangedInd }" />
</c:if>

<c:if test="${command.context.guessingInd == 'true'}">
    <div class="error">
        <spring:message code="htmlformentry.form.reconstruct.warning" />
    </div>
</c:if>



<div id="main-wrap">

    <div id="sidebar">
        <div>

            ${command.htmlToDisplay}  <%-- THIS is where the Magic happens: Rendering the HTML form and
displaying on this particular Div --%>
        </div>

    </div>
    <div id="content-wrap">
        <div id="info-wrap">
            <center>
                <canvas id="myCanvas" width="400" height="200" style="border:1px solid #000000;">
                    Your browser does not support the HTML5 canvas tag.
                </canvas>
            </center>

        </div>
        <div id="info-wrap">
            </br></br><span>Patient Narrative</span>
            <textarea rows="4" cols="50">
                Describe your narrative here.
            </textarea>
        </div>
        <div id="info-wrap">
```

```
            </br></br><span>Upload file (X-ray, reports, etc)</span>
            <input type="file" name="file" id="file" size="40"/>
        </div>
        <div id="info-wrap">
            </br></br>
            <%--<form action="" method="post">--%>
            <%
                ReCaptcha c = ReCaptchaFactory.newReCaptcha("6LdAWuMSAAAAAD3RQXMNBKgI9-1OiYjDx_sl0xYy",
"6LdAWuMSAAAAALxWgnM5yRj_tGVRQCk4lit8rLHb", false);
                out.print(c.createRecaptchaHtml(null, null));
            %>

            </br>
            <input id="submit" type="button" value="Submit" />
            <%--</form>--%>
        </div>
    </div>


</div>



<c:if test="${command.context.mode != 'VIEW'}">
    <div id="passwordPopup" style="position: absolute; z-axis: 1; bottom: 25px; background-color: #ffff00;
border: 2px black solid; display: none; padding: 10px">
        <center>
            <table>
                <tr>
                    <td colspan="2"><b><spring:message code="htmlformentry.loginAgainMessage"/></b></td>
                </tr>
                <tr>
                    <td align="right"><b>Username:</b></td>
                    <td><input type="text" id="passwordPopupUsername"/></td>
                </tr>
                <tr>
                    <td align="right"><b>Password:</b></td>
                    <td><input type="password" id="passwordPopupPassword"/></td>
                </tr>
                <tr>
                    <td colspan="2" align="center"><input type="button" value="Submit" onClick="
loginThenSubmitHtmlForm()"/></td>
                </tr>
            </table>
        </center>
    </div>
</form>
</c:if>

<c:if test="${not empty command.fieldAccessorJavascript}">
    <script type="text/javascript">
        ${command.fieldAccessorJavascript}
    </script>
</c:if>
<c:if test="${not empty command.setLastSubmissionFieldsJavascript || not empty command.
lastSubmissionErrorJavascript}">
    <script type="text/javascript">
        $j(document).ready( function() {
            ${command.setLastSubmissionFieldsJavascript}
            ${command.lastSubmissionErrorJavascript}

            $j('input[toggleDim]:not(:checked)').each(function () {
                var target = $j(this).attr("toggleDim");
                $j("#" + target + " :input").attr('disabled', true);
                $j("#" + target).animate({opacity:0.5}, 100);
            });

            $j('input[toggleDim]:checked').each(function () {
                var target = $j(this).attr("toggleDim");
                $j("#" + target + " :input").removeAttr('disabled');
                $j("#" + target).animate({opacity:1.0}, 0);
```

```
            });

            $j('input[toggleHide]:not(:checked)').each(function () {
                var target = $j(this).attr("toggleHide");
                $j("#" + target).hide();
            });

            $j('input[toggleHide]:checked').each(function () {
                var target = $j(this).attr("toggleHide");
                $j("#" + target).fadeIn();
            });

        });
    </script>
</c:if>

<c:if test="${!pageFragment}">
    <c:choose>
        <c:when test="${inPopup}">
            <%@ include file="/WEB-INF/template/footerMinimal.jsp" %>
        </c:when>
        <c:otherwise>
            <%@ include file="/WEB-INF/template/footer.jsp" %>
        </c:otherwise>
    </c:choose>
</c:if>
```