HF MVC Day -2

Controller, AbstractAction, ParamEnum

Why HF MVC?

- In good old days, any web application had few pieces, and up to the engineer to do the right thing.
  - You write a dynamic JSP - you can add all your code here and post the form to the JSP or post to a Servlet.
  - Your Servlet can do the right thing by having only request mapping and rest the model does or you can do the crazy way do from JDBC connection to HTML building inside a servlet and call yourself OO!
  - Your model can do whatever you like from processing request to building HTML!
- Current HF MVC is the result of all the above(not that bad) and its side effects which led us to the present day EHRController and AbstractAction way.
- We could have used struts or any other readily available MVC framework and added that keyword to our resumes - but HF web applications do not have lots of bells and wishtles and a generic framework needed a framework to make it HF Way - so our EHRController and AbstractAction - is our way ot limiting HF Engineer to think HF Way, get on board quickly.

Why move away?

- Why we moved away from creating individual servlets:
  - The servlet - can never have instance variable - its not thread safe. We had instances where we accidentally did and realized the hard way we violated PHI since some one else is seeing a garbage data - tracing we realized all because of instance variables.
  - This is not too crazy but added a deployment item to check - web.xml - every time you added a servlet - you need to remember to put the web.xml entry in the Deployment Machine. With one servlet - we never have to change any more unless specifically we are doing a servlet for a different application. Note - we do occasionally write Servlet but only either quick and dirty ccare application or no one cares application. Rest must follow protocol.
  - Real-Time application, Claim Application - relied on Servlets - PM where we started to play around with this concept of Controller and Actions - which we really made the HF way when we revamped it in EHR.

EHRController - Main Servlet

```
public class EHRController extends BaseServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doAction(HttpServletRequest _request,
            HttpServletResponse _response) throws IOException, ServletException {
        //leave the following debugging info for now
        StringBuffer sb = new StringBuffer();
        ServletUtil.printRequest(_request);
        HFLogger.getLogger().info(sb.toString());
        EHRActionEnum action = null;
        String contentType = _request.getContentType();
        ActionInterface actionClass = null;
        FileUploadUtil fileUpload = null;
        String referrer = _request.getHeader("Referer");
        if (contentType != null && contentType.indexOf("multipart/form-data") > -1) {
            fileUpload = new FileUploadUtil();
            fileUpload.processRequest(_request);
            fileUpload.printRequestParameters();
            action = EHRActionEnum.valueOf((String) fileUpload.getParameter(CommonFormParamEnum.ACTION_NAME.toString()));
        } else {
            String actionName = _request.getParameter(CommonFormParamEnum.ACTION_NAME.toString());
            if (TextUtil.isEmpty(actionName, true)) {
                throw new InfrastructureException(" Action name is empty!!" + referrer);
            }
            action = EHRActionEnum.valueOf(actionName);
        }
        actionClass = action.getAction();
        actionClass.setMultiPartFormData(fileUpload);
        String url = actionClass.execute(_request, _response);
        if (!TextUtil.isEmpty(url, true)) {
            forward(url, _request, _response);
        }
    }
}
```

EHRController - contd..

- Things to remember:
  - CommonFormParamEnum.ACTION_NAME.name()
  - All HTML Form Action, jquery Ajax Action, must have the above form parameter defined - as hidden parameter, pass the parameter while building the JSON parameter. This is the key to instantiating the action class.
  - If your form has Multipart-form - you cannot map the traditional way of HTTP request parameter mapping. Use the FileUpload to get the HTTP Request parameter instead if request.getParameter() way.
  - Your Action Class must be mapped in the EHRActionEnum Enum.
- The EHRController is the pass through service with the major function being - identify the Action Class who is mainly responsible for the next step.
- Once Action Class finishes - find the URL where it goes next.
- Other than these 2 task the Controller - does nothing.

AbstractAction - Template Pattern

- All HF Web Application has this pattern which is identified by the following:
-

```
public String execute(HttpServletRequest _request, HttpServletResponse _response) {
    this.request = _request;
    this.response = _response;
    if (isValidAction()) {
        if (enableRequestDump())
            ServletUtil.printRequest(_request);
        extractRtUserFromSession(_request);
        if( isPatientPortalPremium() ){
            mapHttpRequestParams(_request, _response);
            performAction();
        }
        prepareResponseParams(_request, _response);
        return getURLToForward();
    } else {
        // session has expired.
        return "/electronic/logout.jsp";
    }
}
```

AbstractAction - contd..

- So far all the HF web application you wrote revolves around the same thing:
  - Map the HTTP Request
  - Perform some Action - invoke the BL
  - Prepare Response Parameter
  - Identify the next URL you will be heading to.
  - All the above only if the login is still valid
  - If invalid login - then go to the logout page!
- AbstractAction encapsulates the above steps in a tiny template the execute method - which then the individual Concrete Action are free to override (must override - the methods are mostly empty place holders.)
- AbstractAction - Has two flavors -
  - EHRAbstractAction - all EHR Related minor methods included and it implies you are going to URL next.
  - AbstractJSONAction - Which extends from AbstractEHRAction - but does not result URL instead the concrete class must override buildJSONArray - which will be returned to the client.

AbstractJSONAction - One more Template

```
 */
public abstract class AbstractJSONAction extends EHRAbstractAction {
    protected JSONArray array = new JSONArray();

    protected abstract void buildJSONArray() throws JSONException;

    protected void addToJSONArray(JSONObject _obj) {
        array.put(_obj);
    }

    protected void setJSONArray(JSONArray _array) {
        array = _array;
    }

    @Override
    protected void prepareResponseParams(HttpServletRequest _request, HttpServletResponse _response) {
        try {
            buildJSONArray();
        } catch (JSONException e1) {
            e1.printStackTrace();
            throw new InfrastructureException(e1);
        }
        try {
            PrintWriter writer = _response.getWriter();
            String jsonArr = array.toString();
            writer.write(jsonArr);
            if (jsonArr.indexOf("},") > -1) {
                jsonArr = jsonArr.replaceAll("},", "},\n");
            }
            HFLogger.getLogger().info(jsonArr);

        } catch (IOException e) {
            e.printStackTrace();
            throw new InfrastructureException(e);
        }
    }
```

AbstractJSONAction - contd..

- The abstract method buildJSONArray() is to be overriden.
- The template method in this case is prepareresponsepatameter.
- Notice getURL returns null - since you wrote the JSON string in the outputstream.

To Follow/Not to Follow Examples

- Review some examples - good practices, bad practices.
  - GrandCentral Actions -All of the grand central actions have a common theme - so create an AbstractGCAction - which encapsulates the majority of request mapping.
  - All the subsequent concrete classes are doing nothing more than call the BL.
- Review SendFaxAction - should ideally be split into as many case statement as clicks present on the screen. I believe the original thinking was one JSP one action - but having all the code in one action class and not in individual classes - we are not bound to test the entire page and every click instead of one action.
  - Note: If you extract a AbstractAction for SendFaxAction - anytime you changes the Abstract class you should ideally test every concrete class if something common was touched.
- PLEASE DO NOT BE LAZY AND USE STRING REQUEST PARAMETER NAME - JUST TAKE A MINUTE TO DEFINE A PARAM ENUM OR IF YOU ARE DEALING WITH PROVIDER_ID, PRACTICE_ID CHANCES ARE THEY ARE ALREADY PRESENT IN EHRPARAMENUM!