

Spring Migration Guidelines for WEB Application

Table of Contents

Overview	4
Why to Migrate	5
High Level Requirements	6
Current Component	7
Proposed Component.....	8
Internal Web Structure.....	9
System Dependencies	10
Standard Project Structure.....	11
Before Spring Migration please complete.....	12
Migration Flow.....	13
1. Convert JSP/HTML/Tiles into Freemarker Template	14
HTML/JSP Migration Code Sample.....	17
2. Change Form bean to POJO	25
3. Change Action classes to Controller	29
4. Accessing account info	31
5. Form Validation.....	31
6. Controller level exception handling	31
7. Change from Struts Configuration to Spring Configuration.....	32
8. Removing EJB.....	35
9. Spring JDBC Template and Dao	36
10. Spring Security	42
11. Spring Security Setup in Local Machine	44
12. Common Header/Footer Setup.....	46
13. Common Menu Filter Setup.....	50
14. Intercept Filter Setup	52
15. Spring Security - Cache Control Header Setup	53
Glossary	54
Appendix A: Checklist.....	55
Appendix B: Code Samples	56

Natural Work Group

Name	Team	Ext
Micheal Reynolds	BT Internet	51896
Mohammed Ehfaj Khan	BT Internet	54497

Overview

This document provides layout and guidelines dealing with how web applications based on the Struts Framework can be migrated to use Spring MVC/Web Flow & FreeMarker.

Below are the few key features regarding spring framework:

1. Spring MVC provides business objects instead of form beans.
2. Spring MVC Supports built in validation and error handling very well, thereby reducing the job of the developer.
3. Flexible mapping to controllers as single entry point. (not just to URL patterns).
4. It allows you to build application on top of Spring, letting you use one Framework/configuration scheme in all tiers.
5. It is easy to use Aspect Oriented Programming (AOP), IOC, and other Spring features in web tier.
6. It is Compatible with newer technologies and tools. (like Freemarker).
7. Spring is view agnostic at its core. You can use a variety of tools to render the view layer once you establish the back-end as a Spring application.

Why to Migrate

1. **Better Inheritance Structure** - The Struts design is based on concrete inheritance, so that each custom action has to be in an inheritance hierarchy of the Struts Action component. On the other hand Spring controllers are interfaces, any component can play the role of the controller. This gives application designers more flexibility in the design of components.
2. **Flexible** - At the framework component level, Struts requires use of Struts-specific objects, such as Form Beans (static or dynamic), Actions, Action Mappings, Action Forwards, and Request Processors. Spring MVC is more flexible, as all its major components are defined as interfaces.
3. **Struts is a Web Framework Only** - Struts addresses only the presentation aspects of application development. On the other hand, Spring MVC is an integral part of the Spring framework, which fully integrates Spring with the rest of the frameworks managing business components as well as other aspects of Spring enterprise development.
4. **No action Forms** - One of the biggest and the most positive differences in the Spring framework is that it has no specialized ActionForm objects. The framework supports binding of HTTP form values directly into POJOs (Plain Old Java Objects). This feature greatly simplifies application maintenance by limiting the number of classes to create and maintain.
5. **Autowiring** – To piggy-back on the form discussion, Spring MVC also contains built-in support for autowiring POJOs to incoming request parameters on a POST so that the mapping of form value to Java object value is simply a matter of making sure the names match up. No specialized tags are required in the HTML to make this work.

High Level Requirements

1. Remove Struts web framework, JSP, and Tiles application by application.
2. Remove Struts manual coding, replacing it with Spring MVC/Web Flow/Security.
3. Create a Spring layered architecture that is lightweight.
4. Remove EJB where possible to reduce the application complexity and overhead.
5. Implement FreeMarker which is lightweight replacement of Tagslibs/JSP/Tiles and makes it easy to make sweeping changes to the layout without deploying an EAR app.

Current Component

The existing architecture includes a mixture of Struts 1.1, EJBs, and a combination of JSP/Tags/PCC in order to render the front-end.

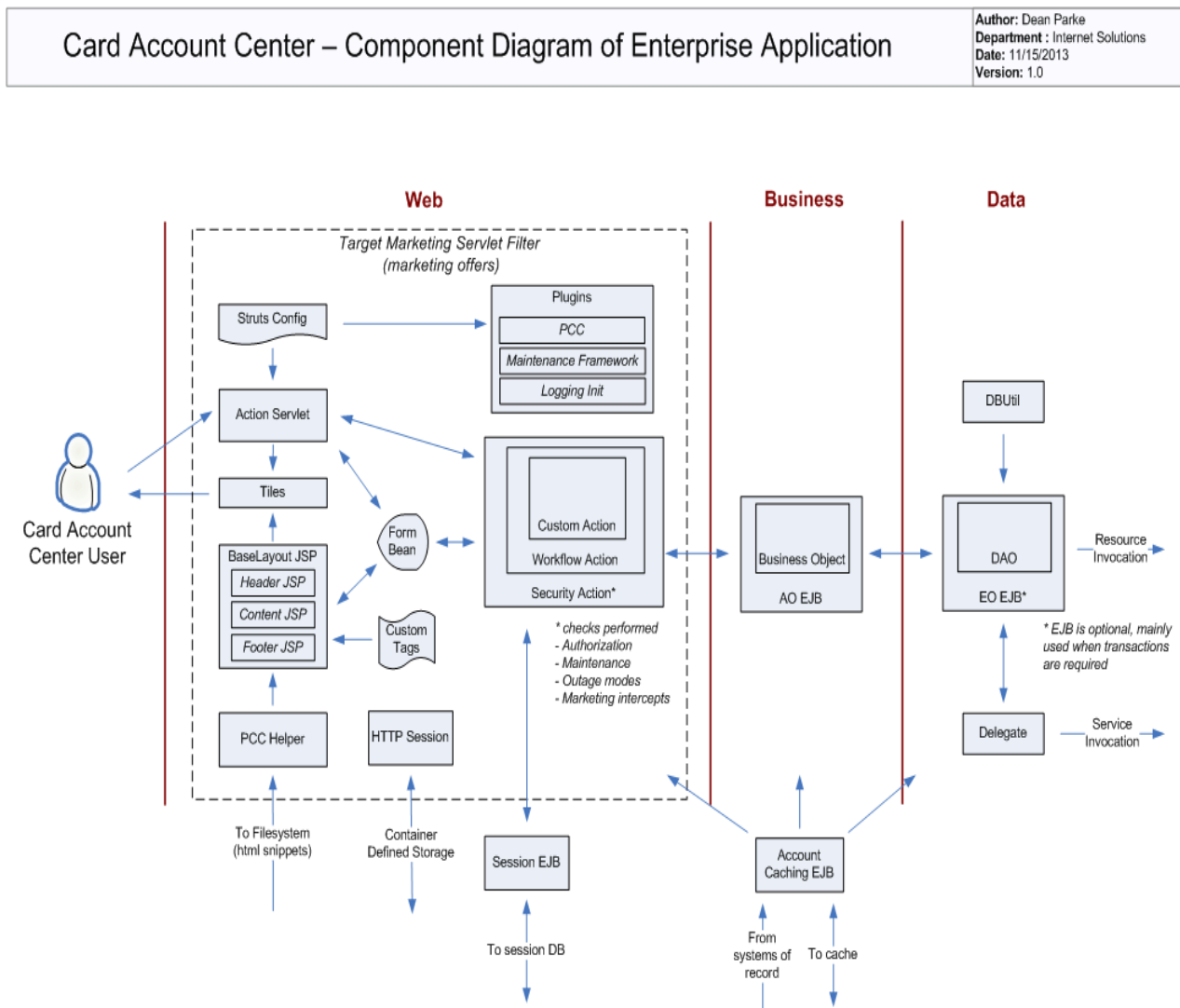


Figure 1 - High Level Application Flow

Proposed Component

As can be seen in the following diagrams, the existing Struts/PCC architecture will completely revamped and updated to include a flexible Spring architecture with an external template system driving look and feel instead of the PCC/JSP hybrid that exists in the Account Center today.

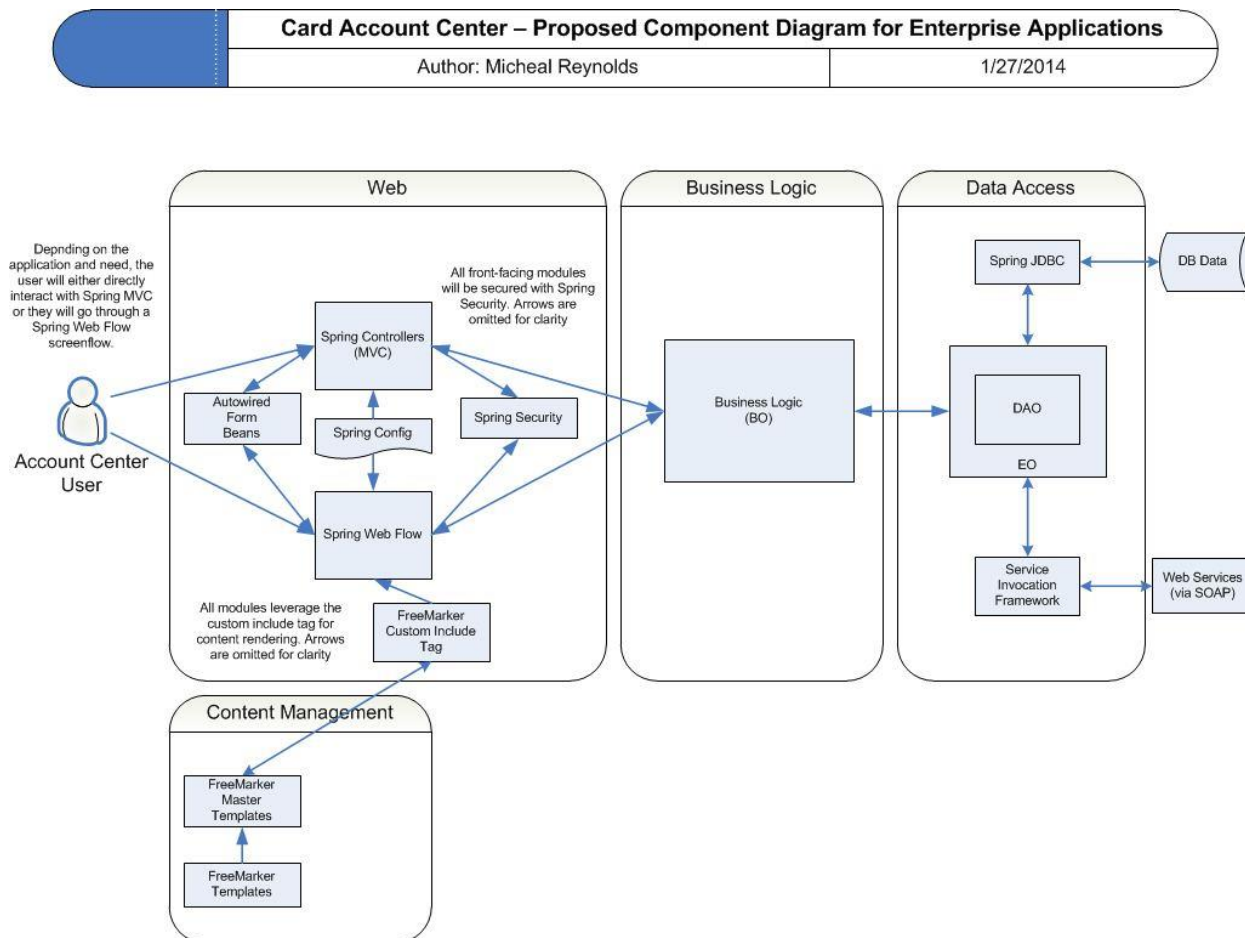


Figure 2 - High Level Application Flow

Internal Web Structure

The internal component structure of the actual web application consists of three segments: Spring Core, Spring MVC/Webflow, and the FreeMarker library.

Application will be built using above three new frameworks. Spring MVC will be used to leverage the basic premises of Inversion of Control and Dependency Injection. Spring Web Flow will be used to control workflow in the application, and FreeMarker will be used to create a template structure that can live in a directory outside of the deployed EAR. The following diagram gives a generic breakdown of those components.

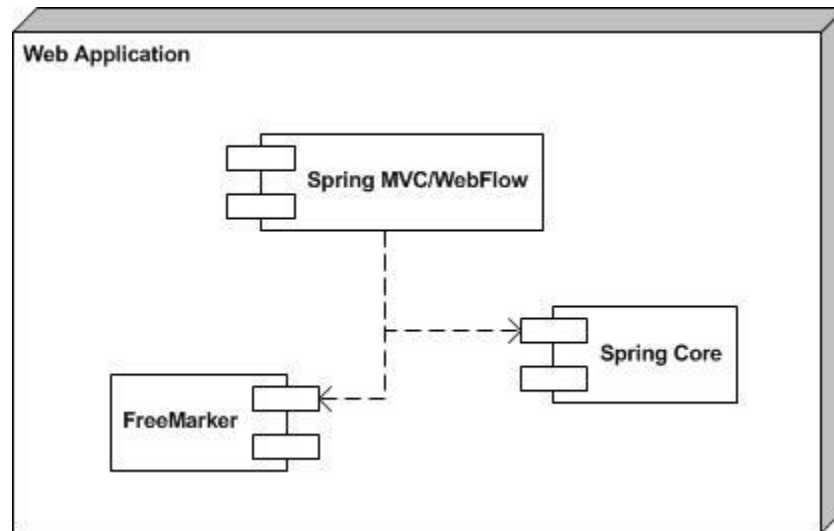


Figure 3 - Web Application Internal Structure

System Dependencies

Account Center Common

During Spring Migration CMS Common (CMSCommon.jar) will be replaced with Account Center Common (AccountCenterCommon.jar).

Clear Case location – **inet_common_pov** → **AccountCenterCommon_SpringMigration**

Account Center Common Features

1. Spring Security & Global Exceptions Handling
2. Strong Auth Security
3. FreeMarker file template loader
4. Property file loader
5. Include template directive
6. Handling Outage/Access mode
7. Authorization/Authentication
8. Keep Session Alive
9. Session Cookie
10. Shared link configuration
11. Maintenance page handling

Third party Jars

Clear Case location - **inet_thirdparty**



Third Party Jars		
commons-lang.jar	org.springframework.aop-3.1.1.RELEASE.jar	org.springframework.js.resources-2.3.0.RELEASE.jar
commons-logging-1.1.3.jar	org.springframework.asm-3.1.1.RELEASE.jar	org.springframework.js-2.3.0.RELEASE.jar
freemarker.jar	org.springframework.aspects-3.1.1.RELEASE.jar	org.springframework.orm-3.1.1.RELEASE.jar
hamcrest-all-1.3.jar (need to be added)	org.springframework.beans-3.1.1.RELEASE.jar	org.springframework.oym-3.1.1.RELEASE.jar
hamcrest-core-1.3.jar (need to be added)	org.springframework.binding-2.3.0.RELEASE.jar	org.springframework.spring-library-3.1.1.RELEASE.libd
hibernate-validator-4.2.0.Final.jar	org.springframework.context.support-3.1.1.RELEASE.jar	org.springframework.test-3.1.1.RELEASE.jar
jackson-all-1.8.1.jar	org.springframework.core-3.1.1.RELEASE.jar	org.springframework.transaction-3.1.1.RELEASE.jar
jai_codec.jar	org.springframework.expression-3.1.1.RELEASE.jar	org.springframework.web.portlet-3.1.1.RELEASE.jar
jai_core.jar	org.springframework.faces-2.3.0.RELEASE.jar	org.springframework.web.servlet-3.1.1.RELEASE.jar
jaxb-api-2.2.jar	org.springframework.instrument.tomcat-3.1.1.RELEASE.jar	org.springframework.web.struts-3.1.1.RELEASE.jar
junit-4.8.jar	org.springframework.instrument-3.1.1.RELEASE.jar	org.springframework.web-3.1.1.RELEASE.jar
log4j.jar	org.springframework.jdbc-3.1.1.RELEASE.jar	org.springframework.webflow-2.3.0.RELEASE.jar
slf4j-api-1.6.4.jar	org.springframework.jms-3.1.1.RELEASE.jar	spring-mock-2.0.8.jar
slf4j-log4j12-1.6.4.jar		spring-test-3.2.6.RELEASE.jar

Standard Project Structure

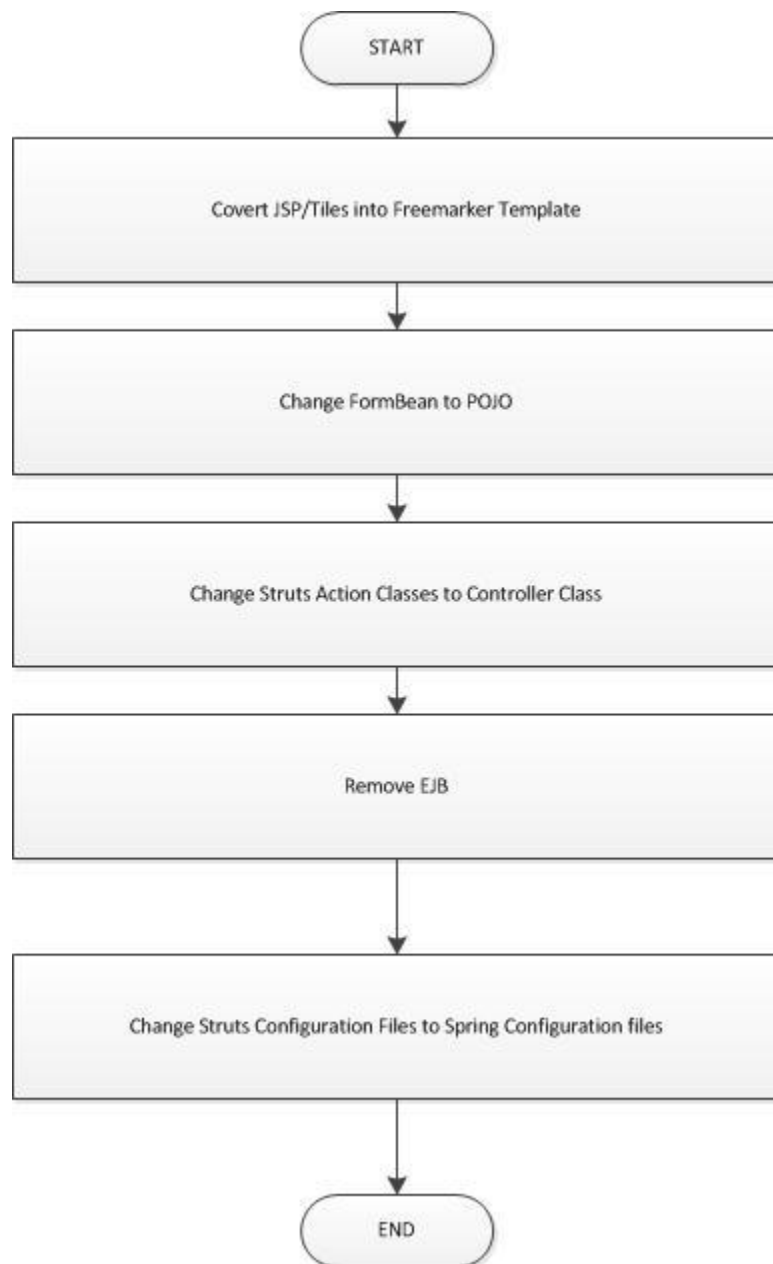
Packages

com...bo – **Stores BO classes**
com...common – **Common classes**
com...controller – **Application's controller**
com...dao – **DAO classes (If any)**
com...eo – **Business delegates initialization/calls**
com...exception – **Custom exceptions**
com...form – **Application's forms**
com...vo – **VO classes**
com...util – **Utility Classes**
com...validator – **Form Validators**

Before Spring Migration please complete

-  Spring Migration Guideline - Best Practices.pdf
-  Appendix A: Checklist

Migration Flow



1. Convert JSP/HTML/Tiles into Freemarker Template

In order to ease maintenance and increase flexibility inside the application, every page will have a dynamically driven set of templates controlled by the FreeMarker template engine. In order to support quick and broad changes without a code release, Application should establish a master template for each page. The master templates will then include or exclude nested sub-templates based on the data passed to the application in order to build custom pages or permutations of pages based on the data being presented to the end user. In order to ease the number of templates created, there will be a custom FreeMarker directive created for generating the name of the file it needs to load on an include. This will allow the separation of files by things such as source code for specific needs but still allow for the default case if there isn't a specialized file for the particular data set.

For example, if there is a header section that is customized by source code, and the app is trying to load the look and feel for source code '3T4E', then the custom directive will look for a template for the header section with 3T4E incorporated into the filename. If there is no such file, the system will assume there is no header customization for that particular source code and load the default header section. In this way, it works very similarly to the PCC loading system used by the DiscoverCard Account Center. This will drastically cut down on the number of files required to maintain the various looks for each source code as the system will not need a full set of files for each source code as the DynaView system does today. It will instead only have file variations for those segments which NEED to be differentiated somehow.

Freemarker template (.FTL) will be stored in the same PCC location (create new folder "freemarker" within PCC folder)

Example: /sys_data/websphere/inetcmspcc/content/<app folder>/freemarker

a. Change HTML file into FTL file

Example:

File Name: eligibleBanks.shtml



```
<div class="formLeft">
  <input name="bankKey" type="radio" value="<%bankKey%>" <%checked%>/>
</div>
<div class="formRight">
  <strong><%bankName%></strong>&mdash;Routing Number
  <%bankRoutingNumber%>&mdash;Account Number <%bankAccountNumber%>
</div>
```

b. Freemarker Template

File Name: eligibleBanks.ftl



```
<div class="formLeft">
  <input name="bankKey" type="radio" value="${bankKey}" ${checked}/>
</div>
<div class="formRight">
  <strong>${bankName}</strong>&mdash;Routing Number
  ${bankRoutingNumber}&mdash;Account Number ${bankAccountNumber}
</div>
```

c. Creating MASTER TEMPLATE

Master template is the template placed in your web application's freemarker folder. This will be common template across the application; other template will share this template. Master template will decide which page/template to load based on passing values/parameters in runtime.

Parameters will be passed by java code (i.e Controller class)

Sample layout:



File Name: masterTemplate.ftl

```
<!-- Custom directive to include templates -->
<#assign includeTemplateFile =
"com.discovercard.cardmembersvcs.common.freemarker.directive.IncludeTemplateFile"?new
()>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html:html xmlns="http://www.w3.org/1999/xhtml">

<!-- Include application specific css/js (If any) -->
<@includeTemplateFile page="<path>" component="" name="css.ftl"/>

<!-- Include <head> element -->
<@includeTemplateFile page="header" component="" name="head.ftl"/>

<body >
    <div id="container">

        <!-- Include common header & menu both -->

        <@includeTemplateFile page="header" component="" name="header.ftl"/>

        <!-- main content -->
        <!-- below <div> is opened by header.ftl (AC nav code) -->
        <!-- <div class="content"> -->

        <!-- Application specific page -->
        <@includeTemplateFile page="{pageName}" component="main"
name="mainPage.ftl"/>
        </div>

        <link rel="stylesheet" type="text/css" href="/css/oo5_style.css" />
        <div class="clear">&nbsp;</div>

        <!-- Include Common Footer -->
        <@includeTemplateFile page="footer" component="" name="footerData.ftl"/>

        <!-- Include Application specific overlays (If any) -->
        <@includeTemplateFile page="{pageName}" component="overlays"
name="overlays.ftl"/>

    </div>
</body>
</html>
```

d. Explaining Master Template



→ include custom directive

File Name: IncludeTemplateFile.java

Location: AccountCenterCommon

FreeMarker user-defined directive for including a template. This is used by the Account Center in order to make it possible to load a version of a template driven by a variety of parameters including page, component, etc.

Since there can be many different data values by which we build filenames, we will not name the incoming parameters. We will call them pOne, pTwo, pThree, and pFour. We will not account for the passing of more than four random values.

Parameters:

pOne: The first parameter used to build the file name.

pTwo: The second parameter used to build the file name.

pThree: The third parameter used to build the file name.

pFour: The fourth parameter used to build the file name.

dir: The directory where we search for files.

name: The actual base filename with extension of the file.

page: The name of the overall page being rendered (ie., confirmation, error, etc.)

component: The component of the overall page being rendered (ie., header, footer, main)

parse: Whether or not the included template is parsed as a FreeMarker template.

→ Include .css/java script

Java script can be included either putting together in one template file or using <script> in standard fashion.

→ Include common header/Menu

Stores CM's name, e-mail address, last four of account number, last login and menu.

→ main content

Main content section will decide which page/template to load based on parameters.

```
<@includeTemplateFile page="{pageName}" component="main" name="mainPage.ftl"/>
```

Assuming pageName="esign" directs masterTemplate to esign/main folder and pick the mainPage.ftl template to load esign page. (Note: try to keep every page name as mainPage.ftl so in runtime you don't need to define file name.)

```
<@includeTemplateFile page="{pageName}" component="main" name="mainPage.ftl" pOne="{mainPageName}"/>
```

Assuming pageName="rewards" and mainPageName="CBB" directs masterTemplate to rewards/main folder and pick the mainPage_CBB.ftl template to load rewards page for CBB card.

→ Include common footer – Common footer across the application.

HTML/JSP Migration Code Sample

Old Struts Code



Struts-config.xml

Config file has ProgramTermsAction with two forwards “cbbTerms” & “milesTerms”. It has two tiles by Card Type CBB and Miles.



“/programTerms” will go to spring’s controller class

```
<action
  className="com.discovercard.internet.common.action.InetActionMapping"
  path="/programTerms"
  type="com.discovercard.cardmembersvcs.rewards.webclient.action.ProgramTermsActi
on"
  scope="request" validate="false">

  <forward

    className="com.discovercard.internet.common.action.InetActionForward"
    name="cbbTerms"
    path="com.discovercard.cardmembersvcs.rewards.tiles.CBBTermsPageTile" />

    <forward
    className="com.discovercard.internet.common.action.InetActionForward"
    name="milesTerms"
    path="com.discovercard.cardmembersvcs.rewards.tiles.MilesTermsPageTile" />

  </action>
```

tiles-def.xml



CBBTermsPageTile has ProgramTermsBodyTile.jsp to render page content



No spring migration for tiles

```
<definition
name="com.discovercard.cardmembersvcs.rewards.tiles.CBBTermsPageTile"
extends="com.discovercard.cardmembersvcs.rewards.tiles.RedeemBaseTile">
  <put name="titleText" value="Discover Card: Cashback Bonus Program Terms
and Conditions" />
  <put name="BodyTile" value="/web/rrm/ProgramTermsBodyTile.jsp" />
</definition>

<definition
name="com.discovercard.cardmembersvcs.rewards.tiles.MilesTermsPageTile"
extends="com.discovercard.cardmembersvcs.rewards.tiles.CBBTermsPageTile">
  <put name="titleText" value="Discover Card: Miles Program Terms and
Conditions" />
  <put name="highlightedNavBarLink" value="milesHighlightedNavBarLink" />
</definition>
```

**ProgramTermsAction.java**

Action code has default forward as techdiff error page, making AO call to populate program terms. Once populates program term, set inti request. Based on the card type got to specific forwards (cbbTerms/ milesTerms)

**Action classes go to Spring's Controller with URI mapping (i.e / programTerms)**

```
public class ProgramTermsAction {

    public InetActionForward executeRewardsAction(
        ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response,
        HeaderNavInfo navInfo,
        HeaderInfo hdrInfo) throws Exception {

        ProgramTerms progTerms;

        InetActionForward inetForward = (
            InetActionForward)mapping.findForward("techdiff");

        try {
            RewardsAODelegate rewardInterface = new RewardsAODelegate();
            progTerms =
            rewardInterface.getProgramTerms(acctInfo.getIncentiveTypeCode(),
            acctInfo.getIncentiveCode());

            // Make the list available in the request context for page rendering.
            request.setAttribute("programTerms", progTerms);

            if(RewardsUtil.accountEarnsCBB(acctInfo.getIncentiveTypeCode())) {
                inetForward = (InetActionForward)mapping.findForward("cbbTerms");
            } else if(RewardsUtil.accountEarnsMiles(acctInfo.getIncentiveTypeCode())) {
                inetForward = (InetActionForward)mapping.findForward("milesTerms");
            }
        } catch (Exception e) {
            // Log uncaught exception and forward to default techdiff tile.
        }
        return inetForward;
    }
}
```



ProgramTermsBodyTile.jsp



Migrate JSP/BodyTag/HTML/SHTML to Freemarker

```
<%% page session="false"%>
<%% page import="com.discovercard.cardmembersvcs.common.security.*" %>
<%% taglib uri="struts-bean" prefix="bean" %>
<%% taglib uri="inettags" prefix="dfshtml" %>
<%% taglib uri="rewards" prefix="rewards" %>

<bean:define id="cardType"
    name="<%=AccountCenterAccessControl.HEADER_INFO_ID%>"
    property="cardType"
    type="java.lang.String"/>
<bean:define id="optionCode"
    name="<%=AccountCenterAccessControl.HEADER_INFO_ID%>"
    property="optionCode"
    type="java.lang.String"/>
<bean:define
    id="incentiveTypeCode"
    name="<%=AccountCenterAccessControl.HEADER_INFO_ID%>"
    property="accountInfo.incentiveTypeCode"
    type="java.lang.String" />

<bean:define
    id="incentiveCode"
    name="<%=AccountCenterAccessControl.HEADER_INFO_ID%>"
    property="accountInfo.incentiveCode"
    type="java.lang.String" />

<dfshtml:IncludeTag file="/rewards/rrm/common/BaseTacPage.shtml"
    cardtype="<%=cardType%>" optioncode="<%=optionCode%>"
    incentivetype="<%=incentiveTypeCode%>"
    incentivecode="<%=incentiveCode%>" >

    <rewards:includeParamBody name="content">
        <rewards:programTermsBodyTag/>
    </rewards:includeParamBody>
```



BaseTacPage.shtml

**Migrate HTML/SHTML to Freemarker**

```
<!-- BEGIN: Content Wrapper -->
<div id="content-wrapper">

    <!-- BEGIN: Main Content -->
    <div id="main-content">

        <link href="/css/sub-styles.css" rel="stylesheet" type="text/css" />

        <div id="middle-content">
            <style type="text/css">
                .bodySubtitle {
                    color:#000000;
                    font-family:arial,helvetica,verdana;
                    font-size:12px;
                    font-weight:bold;
                }
            </style>
            <%content%>
        </div>

    </div>
<!-- END: Content Wrapper -->
```



ProgramTermsBodyTag.java



Migrate BodyTag/HTML/SHTML to Freemarker

```

public class ProgramTermsBodyTag extends BodyTagSupport {
    protected StringBuffer tmpBuf = new StringBuffer(10240);
    private static final String TERMS_PATH = "/rewards/termsConditions/";
    protected HttpServletRequest request = null;
    protected java.util.HashMap parmsMap = new java.util.HashMap();

    public int doStartTag() throws JspException {
        String statecode = "";
        String cardType = "";
        String optionCode = "";
        String incentiveType = "";
        String incentiveCode = "";

        try {
            request = (HttpServletRequest) pageContext.getRequest();
            HeaderNavInfo navInfo = (HeaderNavInfo)
request.getAttribute(AccountCenterAccessControl.HEADER_INFO_ID);
            if(navInfo != null) {
                statecode = navInfo.getState();
                cardType = navInfo.getCardType();
                optionCode = navInfo.getOptionCode();
                AccountInfo acctInfo = navInfo.getAccountInfo();
                if(acctInfo != null) {
                    incentiveType = acctInfo.getIncentiveTypeCode();
                    incentiveCode = acctInfo.getIncentiveCode();
                }
            }

            parmsMap.clear();
            ProgramTerms progTerms = (ProgramTerms)
request.getAttribute("programTerms");
            String programTerms = progTerms.getHtmlText();

            parmsMap.put("partnersLink",
RewardsPCCHelper.getContentByStateExclusions(TERMS_PATH + "partnersLink.shtml",
cardType, optionCode, incentiveType, incentiveCode, statecode));
            parmsMap.put("progTerms", programTerms);
            tmpBuf.append(PCCHelper.getAndReplaceFile(TERMS_PATH +
"programTermsAndConditions.shtml", cardType, optionCode, incentiveType,
incentiveCode, parmsMap));

            pageContext.getOut().println(tmpBuf);
        } catch (Exception e) {
        }

        return SKIP_BODY;
    }
}

```

New Spring Code



ProgramTermsController.java



Controller file can be any generic file name and multiple URI's can be placed in one controller by functionality.

```
@RequestMapping(value = "/programTerms", method = RequestMethod.GET)
protected ModelAndView programTerms(HttpServletRequest request,
HttpServletResponse response) throws Exception {

    final AccountInfo acctInfo = (AccountInfo)
request.getAttribute("acctInfo");

    if (acctInfo != null) {
        final HeaderInfoData navInfo = new HeaderInfoData
(acctInfo);
        final String stateCode = (navInfo != null)?
navInfo.getState(): "";
        try {

            final ProgramTerms progTerms =
getRewardsBO(request).getProgramTerms(acctInfo.getIncentiveTypeCode(),
acctInfo.getIncentiveCode());

            final String programTerms = progTerms != null ?
progTerms.getHtmlText() : "";

            // Make the list available in the request context for
page rendering.
            request.setAttribute("progTerms", programTerms);

        } catch (Exception e) {
            // Log uncaught exception and forward to default
techdiff tile.
            logger.warn("Uncaught exception", e);
            throw new TechnicalDifficultiesException("Uncaught
exception");
        }
    } else {
        // Log it and move on to techdiff page
        logger.error(" couldn't fetch account information from the
header.");
        throw new TechnicalDifficultiesException("couldn't fetch
account information from the header.");
    }

    ModelAndView mav = new ModelAndView(MASTER_TEMPLATE);
    mav.addObject("pageName", "termsConditions");

    return mav;
}
```



BaseTacPage.ftl (Freemarker code)s



Freemarker files will be placed in path: C:/sys_data/websphere/inetcmsspcc/content/<app folder>/freemarker/<page folder>/ (i.e. termsConditions)

```
<!-- BEGIN: Content Wrapper -->
<div id="content-wrapper">

    <!-- BEGIN: Main Content -->
    <div id="main-content">

        <link href="/css/sub-styles.css" rel="stylesheet" type="text/css" />

        <div id="middle-content">
            <style type="text/css">
                .bodySubtitle {
                    color:#000000;
                    font-family:arial,Helvetica,verdana;
                    font-size:12px;
                    font-weight:bold;
                }
            </style>
            <@includeTemplateFile page="termsConditions" component="include"
name="programTermsAndConditions.ftl"/>
        </div>

    </div>
<!-- END: Content Wrapper -->
```



programTermsAndConditions.ftl (Freemarker code)



Freemarker files will be placed in path: C:/sys_data/websphere/inetcmspcc/content/<app folder>/freemarker/<page folder>/ (i.e. termsConditions)

```
<table border="0" cellpadding="0" cellspacing="0" width="567">
<tr>
<td>&nbsp;</td>
<td>
    
    <br>

    <@includeTemplateFile page="termsConditions" component="include"
    name="partnersLink.ftl" />

    <br><br><span class = "bodyText">
    If you would like to print this page, select &quot;Print&quot; under your
    browser's &quot;File&quot; menu.</span>
    </td>
</tr>
<tr>
<td>&nbsp;</td>
<td class = "bodyText">
    <br>
    ${Request.progTerms}
</td>
<td>&nbsp;</td>
</tr>
<tr>
<td>&nbsp;</td>
<td class = "bodyText">
    <br>
    <strong>Please use your browser's "Back" button to return to the
    previous page.</strong>
    </td>
<td>&nbsp;</td>
</tr>
</table>
```

partnersLink.ftl (Freemarker code)

```
<span class = "bodyText">
<a href = "/cardmembersvcs/rewards/app/showAllPartnersTerms">View Cashback
Bonus Partners Terms and Conditions</a>
```


2. Change Form bean to POJO



CVVForm.java

Struts Form bean

```
public class CVVForm extends ValidatorForm {
    private static final long serialVersionUID = 1L;
    private String cvv;
    private String expMonth;
    private String expYear;

    public ActionErrors validate(ActionMapping mapping, HttpServletRequest
request) {
        ActionErrors errors = new ActionErrors();

        if(GenericValidator.isBlankOrNull(cvv)) {
            errors.add("cvvNumber", new
ActionError("error.request.cvv.required"));
        }

        if(GenericValidator.isBlankOrNull(expMonth) ||
expMonth.equalsIgnoreCase("Month") ||
GenericValidator.isBlankOrNull(expYear) ||
expYear.equalsIgnoreCase("Year")) {
            errors.add("expDate", new
ActionError("error.request.expDate.required"));
        }

        return errors;
    }

    public String getCvv() {
        return cvv;
    }
    public void setCvv(String cvv) {
        this.cvv = cvv;
    }
    public String getExpMonth() {
        return expMonth;
    }
    public void setExpMonth(String expMonth) {
        this.expMonth = expMonth;
    }
    public String getExpYear() {
        return expYear;
    }
    public void setExpYear(String expYear) {
        this.expYear = expYear;
    }
}
```



Spring Code:



Steps:

1. Remove struts soecific for validator (if any i. e. `ValidatorForm`) and create pojo class.
2. Create form specific Validator extending with `DFSSpringValidator` Class (`InternetCommon.jar`)
3. Bind validator with form using Spring's binder. (`@InitBinder`)

```
//Form Class
public class CVVForm implements Serializable {
    private static final long serialVersionUID = 1L;
    private String cvv;
    private String expMonth;
    private String expYear;

    public String getCvv() {
        return cvv;
    }
    public void setCvv(String cvv) {
        this.cvv = cvv;
    }
    public String getExpMonth() {
        return expMonth;
    }
    public void setExpMonth(String expMonth) {
        this.expMonth = expMonth;
    }
    public String getExpYear() {
        return expYear;
    }
    public void setExpYear(String expYear) {
        this.expYear = expYear;
    }
}

//Create Validator class exteding DFSSpringValidator.
/* DFSSpringValidator has common util and security check methods to perform input
validation.
Common Methods:

noSQLInjection, noXSSAttack, hasForbiddenCharacters, htmlEncodeString,
isAlphaNumeric, isAlphabetic, isNumeric, isAlphaNumericUnderscore,
containScriptTag, isValidLength.

*/
@Component
public class CVVFormValidator extends DFSSpringValidator {

    private final Logger logger = LoggerFactory.getLogger(getClass());
    private boolean isDebug = logger.isDebugEnabled();

    private CVVForm cvvForm;

    //Continue.....
```

```

@Override
    public boolean supports(Class<?> clazz) {
        return CVVForm.class.isAssignableFrom(clazz);
    }

/**
 * Validate input info
 */
@Override
    public void validate(Object obj, Errors errors) {

        cvvForm = (CVVForm) obj;

        //Validate cvv and Check security Attack
        if(!DFSSpringValidator.isNumeric(cvvForm.getCvv()) ||
            !DFSSpringValidator.isValidLength(cvvForm.getCvv(), 3)) {
            messageContext.addMessage(new
MessageBuilder().error().source("cvvNumber").defaultText(getMessage("error.re
quest.cvv.required")).build());
        } else if(!noSQLInjection(cvvForm.getCvv()) ||
            !noXSSAttack(cvvForm.getCvv()) ||
            hasForbiddenCharacters(cvvForm.getCvv())) {
            messageContext.addMessage(new
MessageBuilder().error().source("cvvNumber").defaultText(getMessage("error.re
quest.cvv.invalid")).build());
        }

        //validate Month/Year
        if(!isNumeric(cvvForm.getExpMonth()) ||
            !isNumeric(cvvForm.getExpYear()) ||
            !isValidLength(cvvForm.getExpMonth(), 2) ||
            !isValidLength(cvvForm.getExpYear(), 2)) {
            messageContext.addMessage(new
MessageBuilder().error().source("expDate").defaultText(getMessage("error.requ
est.expDate.required")).build());
        }

        //Bind validator with form.
        @Controller
        public class <Any>Controller {

            @Autowired
            private CVVFormValidator cvvFormValidator;
            @InitBinder("cvvForm")

            private void initCVVFormBinder(WebDataBinder binder) {
                binder.setValidator(cvvFormValidator);
            }
            .
            .
            .

            //Continue.....

```

```
//Associating form with request mapping using @Validated
/*
As soon as request will be triggered, validator is called before executing
any statement from defined method.
In this example CVVFormValidator will be called. Once validator completes
execution BindingResult will contain error status. Use hasErrors() method to
check the error status.
*/

@RequestMapping(value = "/verifyCVV", method = RequestMethod.GET)
public ModelAndView verifyCVV(HttpServletRequest request, HttpServletResponse
response, @ModelAttribute("cvvForm") @Validated CVVForm cartForm,
BindingResult result) {

    if(result.hasErrors()){
        //Perform some error handling

    }
}
}
```

3. Change Action classes to Controller

Struts Code:

```

Public class DashboardAction {

    public InetActionForward executeRewardsAction(ActionMapping
mapping,ActionForm form,HttpServletRequest request,
    HttpServletResponse response, HeaderNavInfo navInfo, HeaderInfo hdrInfo)
    throws Exception {

        // Make a PartnerKey and set the data on it
        PartnerKey theKey = new PartnerKey();
        theKey.setActSourceCode(ACT_SOURCE_CODE);
        theKey.setIncentiveCode(incentiveCode);
        theKey.setIncentiveType(incentiveType);
        theKey.setStateCode(stateCode);
        theKey.setCategoryCode("UDC");

        RewardsAODelegate rewardInterface = new RewardsAODelegate();
        PartnerList charityList = rewardInterface.getPartnerList(theKey);

        //Default Inet Forwards
        InetActionForward inetForward =
(InetActionForward)mapping.findForward("techdiff");

        if(<any logic using charityList> ) {
            inetForward = (InetActionForward)mapping.findForward("cbbFwd");
        } else if(<any logic charityList>) {
            inetForward = (InetActionForward)mapping.findForward("milesFwd");
        }

        return inetForward;
    }
}
//struts-config.xml code
<action
    className="com.discovercard.internet.common.action.InetActionMapping"
    path="/dashboard"
    type="com.discovercard.cardmembersvcs.rewards.webclient.action.Dashboar
dAction" scope="request" validate="false">
    <forward
        className="com.discovercard.internet.common.action.InetActionForward"
        name="cbbFwd"
        path="com.discovercard.cardmembersvcs.rewards.tiles.CBBDashboardPageTil
e" />
    <forward
        className="com.discovercard.internet.common.action.InetActionForward"
        name="milesFwd"
        path="com.discovercard.cardmembersvcs.rewards.tiles.MilesDashboardPageT
ile" />
</action>

```

**Spring Code:****Steps:**

1. Remove action class and create controller.
2. Gather data using BO call.
3. Add data to Spring's ModelAndView Object
4. Move presentation logic to freemarker

```
@Controller
public class DashboardController {

    @RequestMapping(value = "/dashboard ", method = RequestMethod.GET)
    public ModelAndView dashboard(HttpServletRequest request
        throws Exception {

        // Make a PartnerKey and set the data on it
        PartnerKey theKey = new PartnerKey();
        theKey.setActSourceCode(ACT_SOURCE_CODE);
        theKey.setIncentiveCode(incentiveCode);
        theKey.setIncentiveType(incentiveType);
        theKey.setStateCode(stateCode);
        theKey.setCategoryCode("UDC");

        PartnerList charityList = getRewardsBo().getPartnerList(theKey);

        ModelAndView mav = new ModelAndView(MASTER_TEMPLATE);

        //Go to dashboard folder in file system
        mav.addObject("pageName", "dashboard");

        //Take data to freemarker to perform presentation logic
        mav.addObject("charityList", charityList);

        return mav;
    }
}
```

4. Accessing account info



Accountinfo is configured in acSecurity-context.xml file
`<property name="accountInfoLocation" value="acctInfo"/>`

DFSRequestAuthenticationFilter.java from Account Center Common code sets in request object during each request.

Java Code to get Account Info:

```
AccountInfo acctInfo = (AccountInfo) request.getAttribute("acctInfo");
```

5. Form Validation



Please refer “2. Change Form bean to POJO”

6. Controller level exception handling

Developer can define controller level exception (Note: global exception will be handled by “Account center Common” as a part of Spring Security). As soon as exception is thrown, ExceptionHandler will catch and perform defined action.

Define Exception handler in controller

```
@Controller
public class <Any>Controller {

    @ExceptionHandler({NoRedemptionException.class})
    public ModelAndView handle NoRedemptionException (NoRedemptionException ex) {

        ModelAndView mav = new ModelAndView(MASTER_TEMPLATE);
        mav.addObject("pageName", "noRedemption");
        return mav;
    }
}
```

Throwing Exception anywhere in the code

```
@RequestMapping(value = "/dashboard ", method = RequestMethod.GET)
public ModelAndView dashboard(HttpServletRequest request
    throws Exception {

    if (<any Logic>) {
        throw new NoRedemptionException()
    }

    ModelAndView mav = new ModelAndView(MASTER_TEMPLATE);

    //Go to dashboard folder in file system
    mav.addObject("pageName", "dashboard");

    //Take data to freemarker to perform presentation logic
    mav.addObject("charityList", charityList);
    return mav;
}
}
```

7. Change from Struts Configuration to Spring Configuration

a. Spring/Controller Configuration: (Web.xml)

```
<servlet>
  <servlet-name>rewards</servlet-name>
  <servlet-class>
    org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring-config/spring-context.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

b. KeepSessionAlive Configuration (web.xml)

```
<servlet>
  <servlet-name>KeepSessionAlive</servlet-name>
  <servlet-
class>com.discovercard.cardmembersvcs.common.security.KeepSessionAlive</servl
et-class>
  <init-param>
    <param-name>authorizedSAStatus</param-name>
    <param-value>REGULAR</param-value>
  </init-param>
</servlet>

<servlet-mapping>
  <servlet-name>rewards</servlet-name>
  <url-pattern>/app/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>KeepSessionAlive</servlet-name>
  <url-pattern>/keepalive</url-pattern>
</servlet-mapping>
```

c. Maintenance Page Configuration (web.xml)

```
<servlet>
  <servlet-name>MaintenanceInitServlet</servlet-name>
  <servlet-
class>com.discovercard.internet.common.maintenance.MaintenanceInitServlet</servlet-class>
  <init-param>
    <param-name>configFile</param-name>
    <param-value>/sys_data/websphere/inetcmppcc/layout-
config/maintenance-config.xml</param-value>
  </init-param>
  <init-param>
    <param-name>refreshInterval</param-name>
    <param-value>60000</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
//Continue.....
```



```
<servlet-mapping>
    <servlet-name>MaintenanceInitServlet</servlet-name>
    <url-pattern>/maintenanceservlet</url-pattern>
</servlet-mapping>
```

d. Freemarker Configuration (Spring-context.xml)

```
<bean id="fmXmlEscape" class="freemarker.template.utility.XmlEscape"/>

<bean id="freemarkerConfig"
class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">
    <property name="preTemplateLoaders">
        <list>
            <bean id="rewardsTemplateLoader"
class="com.discovercard.cardmembersvcs.common.freemarker.AccountCenterFileTem
plateLoader">
                <constructor-arg type="java.lang.String"
value="/sys_data/websphere/inetcmspcc/content/rewards/freemarker"/>
            </bean>
            <bean id="commonTemplateLoader"
class="com.discovercard.cardmembersvcs.common.freemarker.Ac
countCenterFileTemplateLoader">
                <constructor-arg type="java.lang.String"
value="/sys_data/websphere/inetcmspcc/content/common/freemarker"/>
            </bean>
        </list>
    </property>
    <property name="freemarkerVariables">
        <map>
            <entry key="xml_escape" value-ref="fmXmlEscape"/>
        </map>
    </property>
    <property name="freemarkerSettings">
        <props>
            <prop key="template_update_delay">300</prop> <!-- 5 Min --
        >
    </props>
</property>
</bean>

//Continue....
```

```

<bean id="contentViewResolver"
class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
  <property name="mediaTypes">
    <map>
      <entry key="html" value="text/html"/>
      <entry key="ftl" value="text/html"/>
      <entry key="xml" value="application/xml"/>
      <entry key="json" value="application/json"/>
    </map>
  </property>
  <property name="favorPathExtension" value="true"/>
  <property name="defaultViews">
    <list>
      <bean
class="org.springframework.web.servlet.view.json.MappingJacksonJsonView">
        <property name="prefixJson" value="true"/>
      </bean>
    </list>
  </property>
  <property name="viewResolvers">
    <list>
      <bean
class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver
">
        <property name="cache" value="true"/>
        <property name="order" value="1"/>
        <property name="prefix" value=""/>
        <property name="suffix" value=".ftl"/>
        <property name="contentType" value="text/html;charset=UTF-8"/>
        <property name="exposeSpringMacroHelpers" value="true"/>
        <property name="requestContextAttribute" value="rc"/>
        <property name="exposeSessionAttributes" value="true"/>
      </bean>
    </list>
  </property>
</bean>

```

e. Loading Property File (spring-context.xml)

```

<bean id="messageSource"
class="org.springframework.context.support.ReloadableResourceBundleMessageSou
rce">
  <property name="basenames">
    <list>

<value>file:/sys_data/websphere/inetcmsspcc/content/rewards/spring/properties/
InappropriateWords</value>

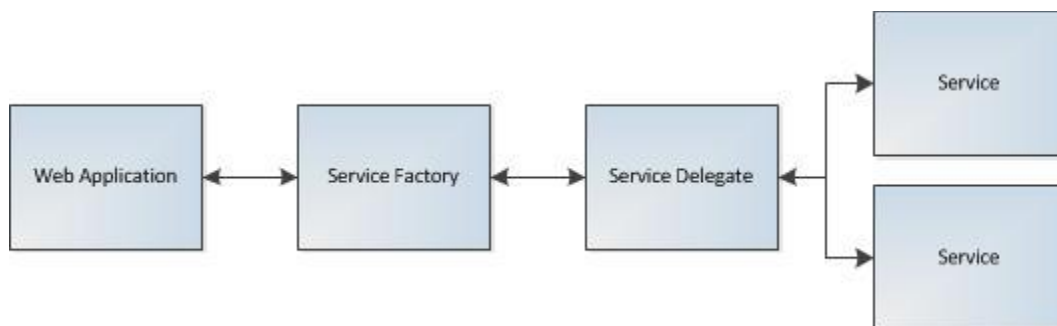
<value>file:/sys_data/websphere/inetcmsspcc/content/rewards/spring/properties/
GroupingDef</value>
    </list>
  </property>
  <property name="cacheSeconds" value=" 86400000"/> <!--24 hrs. -->
</bean>

```

8. Removing EJB



EJB module will be removed from Web Application. There will be no AO layer. Service call will be made by BO layer via EO.



a. Initializing Business Delegates (BD)

Spring Config: (spring-context.xml)

BD config:

```

<bean id="dataServiceFactory"
class="com.discovercard.cardmembersvcs.dataservice.bd.DataServiceFactory"
factory-method="getInstance"/>
  
```

EO Config:

```

<bean id="dataServiceEO"
class="com.discovercard.cardmembersvcs.rewards.eo.DataServiceEO">
  <constructor-arg ref="dataServiceFactory"/>
</bean>
  
```

BO Config:

```


<bean id="rewardsBO"
class="com.discovercard.cardmembersvcs.rewards.bo.RewardsBO">
  <property name="dataServiceEO" ref="dataServiceEO"/>
</bean>
  
```

b. Java side changes

- i. Remove XxxAO.., XxxAoBean.., and XxxAoHome classes.
- ii. Declare EO classes inside BO
- iii. Initiate service call in EO.
- iv. Get VO object in EO class from service class.
- v. Sent back VO to BO and let BO perform other business logic (If applicable).

Note: No business logic in Web Layer (Controller/Form-POJO/View-Freemarker)

9. Spring JDBC Template and Dao

 The JdbcTemplate class is the central class in the JDBC core package. It handles the creation and release of resources, which helps you avoid common errors such as forgetting to close the connection. It performs the basic tasks of the core JDBC workflow such as statement creation and execution, leaving application code to provide SQL and extract results. The JdbcTemplate class executes SQL queries, update statements and stored procedure calls, performs iteration over ResultSets and extraction of returned parameter values. It also catches JDBC exceptions and translates them to the generic, more informative, exception hierarchy defined in the org.springframework.dao package.

When you use the JdbcTemplate for your code, you only need to implement callback interfaces, giving them a clearly defined contract. The PreparedStatementCreator callback interface creates a prepared statement given a Connection provided by this class, providing SQL and any necessary parameters. The same is true for the CallableStatementCreator interface, which creates callable statements. The RowCallbackHandler interface extracts values from each row of a ResultSet.

The JdbcTemplate can be used within a DAO implementation through direct instantiation with a DataSource reference, or be configured in a Spring IoC container and given to DAOs as a bean reference.

To support basic JDBC operations **abstract class** DFSSpringDao is created which is wrapper around Spring JDBC class, this class is located inside Intercommon.jar

```
@Repository
public abstract class DFSSpringDao {

    //Thread safe jdbcTemplate
    private JdbcTemplate jdbcTemplate;

    /**
     * The DataSource to be used by this instance
     */
    protected DataSource dataSource;

    /**
     * The {@link ThrowableProcessor ThrowableProcessor} to be used by this
     instance.
     * The default is {@link DefaultThrowableProcessor
     DefaultThrowableProcessor}.
     */
    protected ThrowableProcessor m_throwableProcessor =
        DefaultThrowableProcessor.getInstance();

    public DFSSpringDao() {

    }

    //Continue...
```

```

/**
 * Constructor. Initializing DFSSpringDao instance with a DataSource.
 * If DFSSpringDao is initialized this way, it will close the
connection after
 * each query or update. This constructor is the recommended
constructor
 * to be used for applications that will be deployed in an application
server.
 *
 * @param dataSource The DataSource to obtain database connections
 */
public DFSSpringDao(DataSource dataSource) {
    this.dataSource = dataSource;
    setDataSource(dataSource);
}

/**
 * This method checks if DataSource is being used.
 *
 * @return true if DataSource is used, false otherwise.
 */
protected boolean isUsingDataSource() {
    return this.dataSource != null;
}

/**
 * Initialize JdbcTemplate
 * @param dataSource
 */
private void setDataSource(DataSource dataSource) {

    this.jdbcTemplate = new JdbcTemplate(dataSource);
}

/**
 * Issue a single SQL update operation (such as an insert, update or
delete statement) via a prepared statement, binding the given arguments. The
database connection will be closed by this method.
 * @param sql - SQL containing bind parameters
 * @param params - arguments to bind to the query (example - new
Object[] { arg1, arg2, ...})
 * @return - Return the number of rows affected
 * @throws DataAccessException - if there is any problem issuing the
update
 */
protected int executeUpdate(String sql, Object[] params) throws
DBUtilException {

}

//Continue...

```

```
    /**
     * Executing query against a database. The database connection will be
     closed by this method.
     * @param sql - SQL command to be executed
     * @param params - Data criteria bind to query
     * @param rowMapper - Type of objects to be returned as result
     * @return - Return collection of objects for specific type specified
     by rowMapper
     * @throws DBUtilException
     */
    protected Collection<?> executeQuery(String sql, Object[] params,
    RowMapper<?> rowMapper) throws DBUtilException {

        }

    /**
     * Executing query against a database. The database connection will be
     closed by this method.
     * @param sql - SQL command to be executed
     * @param rowMapper - Type of objects to be returned as result
     * @return - Return collection of objects for specific type specified
     by rowMapper
     * @throws DBUtilException
     */
    protected Collection<?> executeQuery(String sql, RowMapper<?>
    rowMapper) throws DBUtilException {

        }

    /**
     * Executing query against a database. The database connection will be
     closed by this method.
     * @param sql - SQL command to be executed
     * @param elementType - Type of objects to be returned as result (like
     Integer.class, Double.class)
     * @return - Return List of objects for specific type specified by
     elementType
     * @throws DBUtilException
     */
    protected List<?> executeQuery(String sql, Object[] params, Class<?>
    elementType) throws DBUtilException {

    }

    //Continue...
```

```

    /**
     * Executing query against a database. The database connection will be
     closed by this method.
     * @param sql - SQL command to be executed
     * @return - Return List of objects for specific type specified by
     elementType
     * @throws DBUtilException
     */
    protected List<?> executeQuery(String sql, Class<?> elementType) throws
    DBUtilException {
        }

    /**
     * Executing a stored procedure in the database. This method will
     close the connections that it creates.
     * @param sql - Procedure Name (like PACKAGENAME.PROCDURENAME)
     * @param argument - Data criteria bind to query (like SqlParameter,
     SqlOutParameter)
     * @param isFunction - Indicates whether procedure (false) or function
     (true)
     * @param dbHandler - Customize DBHandler
     * @return
     * @throws DBUtilException
     */
    protected Object executeStoredProcedure(String sql, Object argument,
    boolean isFunction, IDBHandler dbHandler, RowMapper<?> rowMapper)
    throws DBUtilException {
        }

    /**
     * Initialize Simple Jdbc to call stored procedure.
     * @param paramArray - SqlParameter mapping with input (IN)/output (OUT)
     param name and data type from Stored Procedure.
     * @param sql - Procedure Name (like PACKAGENAME.PROCDURENAME)
     * @param isFunction - Indicates whether procedure (false) or function
     (true)
     * @param rowMapper - the RowMapper implementation that will map the
     data returned for each row
     * @return SimpleJdbcCall
     */
    private SimpleJdbcCall initSimpleJdbcCall(SqlParameter[] paramArray,
    String sql, boolean isFunction, RowMapper<?> rowMapper) {
        }

    public JdbcTemplate getJdbcTemplate() {
        return jdbcTemplate;
    }
}

```



Below is the example to use **abstract class** DFSSpringDao

```
public class RewardsCacheDaoImpl extends DFSSpringDao implements
RewardsCacheDao {
    public static String ACCRDFLNEW_SQL = "update ac_crd_fl_new set UPD_DT
= sysdate where CRDMBR_ACCT_NBR = ?";
    public static String INETRWRD_SQL = "update inet_rwr set UPD_TMS =
sysdate where CRDMBR_ACCNT_NBR = ?";

    /**
     * Datasource is passed using configuration xml
     * @param dataSource
     */
    public RewardsCacheDaoImpl(DataSource dataSource) {
        super(dataSource);
    }

    public int updateCacheTableTimeStamp(String accountNumber, String
tableName) {

        Object[] params = {accountNumber};
        int rowsUpdated = 0;

        if (RewardsUtil.AC_CRD_FL_NEW.equalsIgnoreCase(tableName)) {
            rowsUpdated =
executeUpdate(UpdateCacheTimeStampHandler.ACCRDFLNEW_SQL, params);
        } else if (RewardsUtil.INETRWRD.equalsIgnoreCase(tableName)) {
            rowsUpdated =
executeUpdate(UpdateCacheTimeStampHandler.INETRWRD_SQL, params);
        }

        return rowsUpdated;
    }
}

spring-context.xml

<!-- Data Source -->
<jee:jndi-lookup id="cacheDataSource" jndi-
name="jdbc/inetcms/CacheOracleDataSource" expected-
type="javax.sql.DataSource"/>

<!-- Websphere Transaction -->
<bean id="txManager"
class="org.springframework.transaction.jta.WebSphereUowTransactionManager"/>

<!-- Spring Advice -->
<tx:advice id="txAdvice" transaction-manager="txManager">
    <tx:attributes>
        <tx:method name="*" read-only="true"
propagation="NOT_SUPPORTED"/></tx:method>
    </tx:attributes>
</tx:advice>
//Continue...
```



```

<!-- Define the AOP for transactions -->
<aop:config>
    <aop:pointcut id="rewardsDBEOTx" expression="execution(*
com.discovercard.cardmembersvcs.rewards.eo.RewardsDBEO.*(..))"/>

    <aop:advisor advice-ref="txAdvice" pointcut-ref="rewardsDBEOTx"/>
</aop:config>

```



Calling Stored Procedure using **abstract class** DFSSpringDao.

Use executeStoredProcedure () method, and pass customize rowMapper class, based on required returned object

```

public class RewardsCacheDaoImpl extends DFSSpringDao implements
RewardsCacheDao {
    /**
     * Datasource is passed using configuration xml
     * @param dataSource
     */
    public RewardsCacheDaoImpl(DataSource dataSource) {
        super(dataSource);
    }

    public RewardsAndBilling getRewardsAndBilling(String accountNumber)
    throws RewardsException {
        return (RewardsAndBilling)
executeStoredProcedure(GetRewardsAndBillingHandler.GET_REWARD_PROC,
accountNumber, false, new GetRewardsAndBillingHandler(), new
RewardsAndBillingMapper());
    }
    /**
     * Row Mapper for RewardsAndBilling
     */
    private class RewardsAndBillingMapper implements
RowMapper<RewardsAndBilling> {

        public RewardsAndBilling mapRow(ResultSet rs, int rowNum) throws
SQLException {
            RewardsAndBilling rewardsAndBilling = new
RewardsAndBilling();

            try {
                rewardsAndBilling.setAccountNumber(rs.getString(1));
                rewardsAndBilling.setEarnedRewardAmount(rs.getString(2));
                rewardsAndBilling.setAvailableRewardAmount(rs.getString(3));
            } catch (Exception e) {
                throw new DBUtilException(e.getMessage());
            }
            return rewardsAndBilling;
        }
    }
}

```

10. Spring Security

Since the application is being migrated to the Spring/Spring MVC framework, it will also leverage Spring Security in order to verify that the session is valid for each incoming request. Because DFS is not yet migrated to Spring Authentication, we will leverage Spring's PreAuthorization pattern in order to verify session data and load the required datasets into the application space prior to the execution of any bound controller endpoint.

In this model, the DFS Request Authentication Filter will look up the session identifier from the 'dcsession' cookie the current Account Center uses to control access. If it can't find the cookie or the cookie has an invalid value, it will throw a RuntimeException that is classed to match the application condition. If it locates a valid cookie, it will fetch the account information from the cache based on the account number retrieved from the session database and then push it into the application space for use by downstream classes/the view layer.

A long term solution is being developed Security in spring that will use something more elegant for aggregation and modularity. Because the app realizes the Spring pattern, however, it will be a simple update when that solution is fully designed.

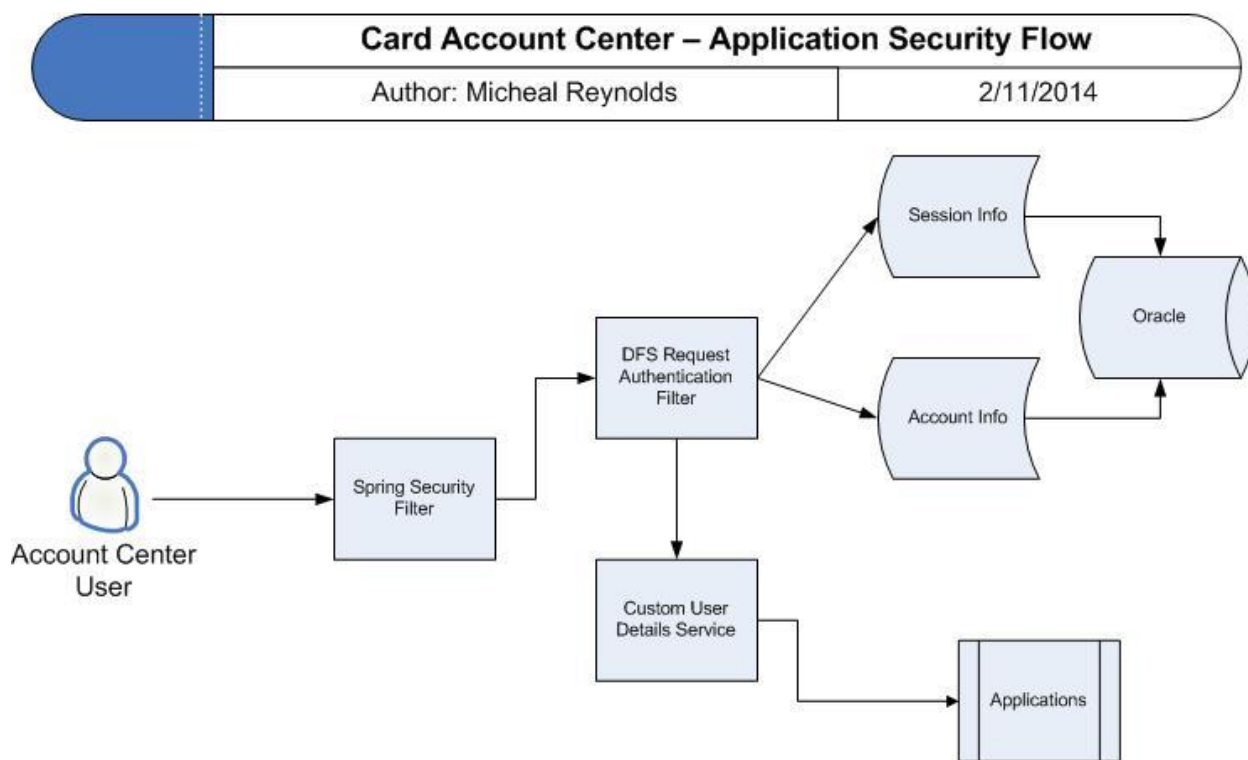


Figure 5 - Spring Security Workflow

Spring Security Configuration (acSecurity-context.xml)

```

<!-- Example of a high security http element and filter configuration
  <security:http use-expressions="true" auto-config="false" entry-point-ref="http403EntryPoint" create-
  session="stateless">
    <security:intercept-url pattern="/highSecRequiredURI" access="hasRole('ROLE_USER')"/>
    <security:custom-filter position="PRE_AUTH_FILTER" ref="dfsSecurityFilterHigh"/>
  </security:http>

  <bean id="dfsSecurityFilter"
  class="com.discovercard.cardmembersvcs.common.security.DFSRequestAuthenticationFilter">
    <property name="cookieName" value="dcsession"/>
    <property name="authorizedSAStatus" value="HIGH"/>
    <property name="accountInfoLocation" value="acctInfo"/>
    <property name="authenticationManager" ref="authenticationManager"/>
  </bean>
  -->

  <!-- This is the default security filter for normal security. In the case of a HIGH security need, another filter
  calling
  the exact same class but with the HIGH value for authorizedStatus would be defined. Then, a new http
  element would
  need to be created above the default to intercept those high need URLs. See above example. -->
  <security:http use-expressions="true" auto-config="false" entry-point-ref="http403EntryPoint" create-
  session="stateless">
    <security:intercept-url pattern="/*" access="hasRole('ROLE_USER')"/>
    <security:custom-filter before="PRE_AUTH_FILTER" ref="dfsSecurityExceptionFilter"/>
    <security:custom-filter position="PRE_AUTH_FILTER" ref="dfsSecurityFilter"/>
  </security:http>

  <bean id="dfsSecurityFilter"
  class="com.discovercard.cardmembersvcs.common.security.DFSRequestAuthenticationFilter">
    <property name="cookieName" value="dcsession"/>
    <property name="authorizedSAStatus" value="REGULAR"/>
    <property name="accountInfoLocation" value="acctInfo"/>
    <property name="authenticationManager" ref="authenticationManager"/>
  </bean>

  <bean id="preauthAuthProvider"
  class="org.springframework.security.web.authentication.preauth.PreAuthenticatedAuthenticationProvider">
    <property name="preAuthenticatedUserDetailsService">
      <bean id="userDetailsServiceWrapper"
      class="org.springframework.security.core.userdetails.UserDetailsServiceByNameServiceWrapper">
        <property name="userDetailsService" ref="customUserDetailsService"/>
      </bean>
    </property>
  </bean>

  <security:authentication-manager alias="authenticationManager">
    <security:authentication-provider ref="preauthAuthProvider"/>
  </security:authentication-manager>

  <bean id="dfsSecurityExceptionFilter"
  class="com.discovercard.cardmembersvcs.common.security.DFSSecurityExceptionFilter"/>
  <bean id="customUserDetailsService"
  class="com.discovercard.cardmembersvcs.common.security.CustomUserDetailsService"/>
  <bean id="http403EntryPoint"
  class="org.springframework.security.web.authentication.Http403ForbiddenEntryPoint"/>

```

11. Spring Security Setup in Local Machine

- a. **Setup Fakelogin** (Check Dlife group - "Discover Internet Developer Handbook group", RAD Development
<https://dlife.discoverfinancial.com/docs/DOC-27074>

- b. **Setup Proxy Settings & Edit hosts** (Check Dlife group - "Discover Internet Developer Handbook group":
<https://dlife.discoverfinancial.com/docs/DOC-27683>

<https://dlife.discoverfinancial.com/docs/DOC-27681>

NOTE: Fake login should be like **local.discovercard.com:<port>/url**

- c. **Add acSecurity-context.xml to project path “/WebContent/WEB-INF/spring-config/”**
(Refer Appendix C: Code Samples)
- d. **Add spring-context.xml to project path “/WebContent/WEB-INF/spring-config/”**
(Refer Appendix C: Code Samples)
- e. **Create folder “/sys_data/websphere/inetcmppcc/layout-config/” in local and copy outage-config.xml & accessConfig.xml** (Refer Appendix C: Code Samples)
- f. **Add below content in spring-context.xml**

```
<context:component-scan base-package="<package path>"/>

<mvc:annotation-driven/>

<!-- Outage Utility Bootstrap Class -->
<bean id="outageBootstrap"
class="com.discovercard.cardmembersvcs.common.outage.util.OutageUtilBootstrap">
    <constructor-arg index="0" type="String"
value="/sys_data/websphere/inetcmppcc/layout-config/outage-config.xml"/>
    <constructor-arg index="1">
        <value>60000</value>
    </constructor-arg>
</bean>

<!-- Access Level Bootstrap Class -->
<bean id="accessBootstrap"
class="com.discovercard.cardmembersvcs.common.access.AccessBootstrap">
    <constructor-arg index="0" type="String"
value="/sys_data/websphere/inetcmppcc/layout-config/accessConfig.xml"/>
    <constructor-arg index="1">
        <value>60000</value>
    </constructor-arg>
</bean>
```

- g. **Add security context and filter in web.xml**

```
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring-config/acSecurity-context.xml</param-value>
</context-param>

<!-- Spring security filter -->
```

```
<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

h. Add Session info and account info in web.xml

```
<ejb-ref id="EjbRef_1400085211639">
  <description>
  </description>
  <ejb-ref-name>ejb/SessionInfoAO</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.discovercard.cardmembersvcs.session.ao.SessionInfoAOHome</home>
  <remote>com.discovercard.cardmembersvcs.session.ao.SessionInfoAO</remote>
</ejb-ref>
<ejb-ref id="EjbRef_1400096386620">
  <description>
  </description>
  <ejb-ref-name>ejb/AccountDatabaseEO</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.discovercard.cardmembersvcs.data.account.eo.AccountDatabaseEOHome</home>
  <remote>com.discovercard.cardmembersvcs.data.account.eo.AccountDatabaseEO</remote>
</ejb-ref>
```

NOTE: update ibm-web-bnd.xmi with above reference accordingly

12. Common Header/Footer Setup

I. Overview – Freemarker Template

In order to ease maintenance and increase flexibility inside the application, every page will have a dynamically driven set of templates controlled by the FreeMarker template engine. In order to support quick and broad changes without a code release, Application should establish a master template for each page. The master templates will then include or exclude nested sub-templates based on the data passed to the application in order to build custom pages or permutations of pages based on the data being presented to the end user. In order to ease the number of templates created, there will be a custom FreeMarker directive created for generating the name of the file it needs to load on an include. This will allow the separation of files by things such as source code for specific needs but still allow for the default case if there isn't a specialized file for the particular data set.

Application specific freemarker template (.FTL) will be stored in the same PCC location

Example: /sys_data/websphere/inetcmsspcc/content/<app folder>/freemarker

NOTE: If there is any modification done regarding header/footer/menu items on PCC files, then Freemarker template should also be modified for that application with freemarker's common header/footer/menu item.

II. Account Center Common Freemarker Template

In order to maintain consistency across the web applications, freemarker template has common **header, footer, and menu item**, which can be used by every web application during spring migration.

Note:

Copy "common\freemarker" folder from

Location: please pull from any ENV (Prod/PA) : sys_data/websphere/inetcmsspcc/content/common/freemarker

Local folder structure: C:\sys_data\websphere\inetcmsspcc\content\common\freemarker

Common Header:

File Path: /sys_data/websphere/inetcmsspcc/content/common/freemarker/header

Files List:

1. **header.ftl** – Master file for header. Contains business name, Account Home, Account Profile, Manage Alerts links & Menu items populated by Menu We Service. Please refer chapter 14 for more info.
2. **head.ftl** – Master file for meta-refresh & session timeout configuration.

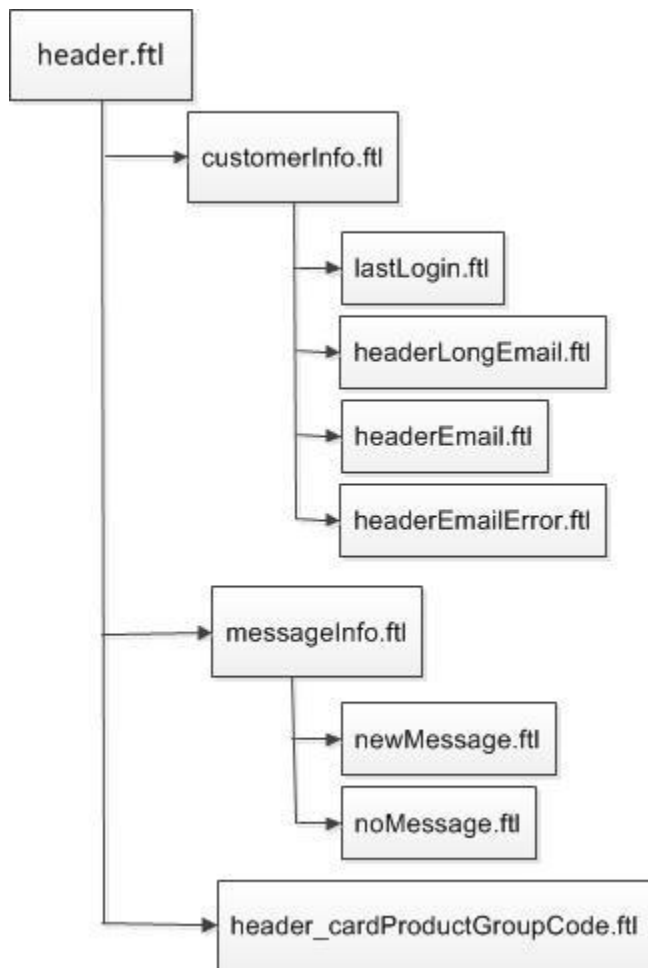
III. Common Footer

File Path: /sys_data/websphere/inetcmspcc/content/common/freemarker/footer

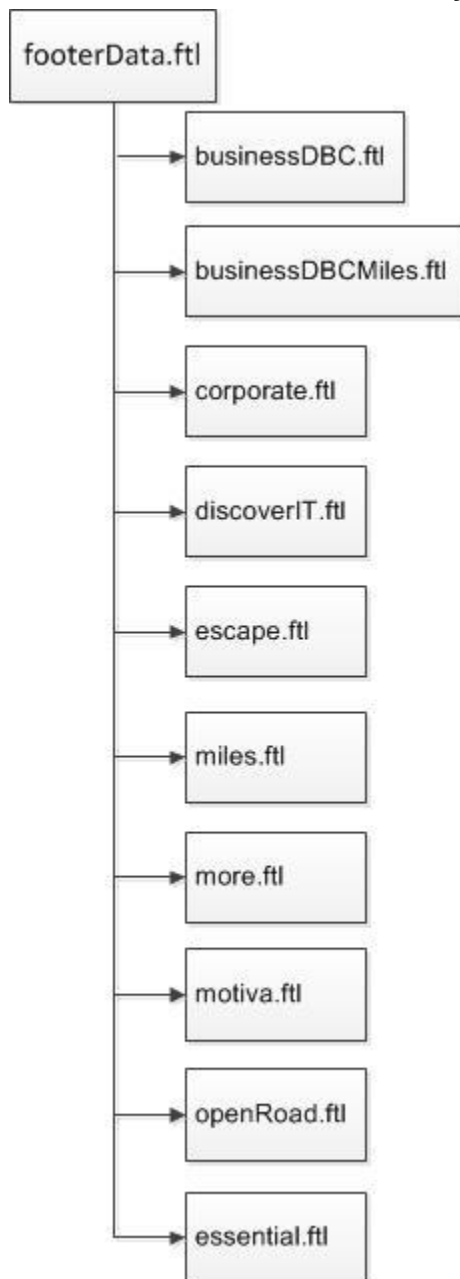
Files List:

1. **footerData.ftl** – Master file for footer. This file includes below .ftl files:
 - a. **businessDBC.ftl**
 - b. **businessDBCMiles.ftl**
 - c. **corporate.ftl**
 - d. **discoverIT.ftl**
 - e. **escape.ftl**
 - f. **miles.ftl**
 - g. **more.ftl**
 - h. **motiva.ftl**
 - i. **openRoad.ftl**
 - j. **essential.ftl**

IV. File Hierarchy – Header



V. File Hierarchy – Footer



VI. Freemarker Setup (spring-context.xml)

```
<bean id="freemarkerConfig"
class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">
    <property name="preTemplateLoaders">
        <list>
            <bean id="rewardsTemplateLoader"
class="com.discovercard.cardmembersvcs.common.freemarker.AccountCenterFileTemplateLoader">
                <constructor-arg type="java.lang.String"
value="/sys_data/websphere/inetcmpcc/content/rewards/freemarker"/>
            </bean>
        </list>
    </property>
    <!--Add Common header/footer -->
    <bean id="commonTemplateLoader"
class="com.discovercard.cardmembersvcs.common.freemarker.AccountCenterFileTemplateLoader">
        <constructor-arg type="java.lang.String"
value="/sys_data/websphere/inetcmpcc/content/common/freemarker"/>
    </bean>
    </list>
</property>
<property name="freemarkerVariables">
    <map>
        <entry key="xml_escape" value-ref="fmXmlEscape"/>
    </map>
</property>
<property name="freemarkerSettings">
    <props>
        <prop key="template_update_delay">0</prop>
    </props>
</property>
</bean>
```

VII. Adding header/footer in MasterTemplate

Sample Code:

```
<#assign includeTemplateFile = "com.discovercard.cardmembersvcs.common.freemarker.directive.IncludeTemplateFile"?new()>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html:html xmlns="http://www.w3.org/1999/xhtml">
    <@includeTemplateFile page="header" component="" name="head.ftl"/>
</body>
<div id="container">
    <@includeTemplateFile page="header" component="" name="header.ftl"/>
    <@includeTemplateFile page="${pageName}" component="main" name="mainPage.ftl"/>
</div>
<link rel="stylesheet" type="text/css" href="/css/oo5_style.css" />
<div class="clear"><div>
    <@includeTemplateFile page="footer" component="" name="footerData.ftl"/>
</div>
</body>
</html>
```

13. Common Menu Filter Setup

All the consumer applications that display Header & Menu item (altogether refer as Header) will be using Menu Web Service in order to populate Header.

a. Account Center Common Code:

Menu population and we service call is done under the Account Center common, inside DFSRequestAuthenticationFilter.

DFS Filter tries to get menu using account info and if available sets it in the request attribute

```
CacheDataInfo cacheDataInfo = acctBO.getAccountInfoWithMenu
(clearAccountNumber);
        if (cacheDataInfo.getFullMenuTxt() != null) {
            aRequest.setAttribute(MenuUtil.MENU_INFO,
cacheDataInfo.getFullMenuTxt());
        }
```

There is another Filter called MenuFilter has the logic for calling menu service in case menu info was not retrieved by DFS Filter as explained above.

Once menu is set in Request attribute, common header.ftl will have reference to populate this menu.

Note: Please refer chapter 13 for more info regarding common header/footer.

b. Consumer Application level configuration:

- i. **MenuWebServiceDelegate:** This will be MenuWebServiceWAS6Delegate.jar or MenuWebServiceClient.jar depending upon on which version of websphere your application is running. **Clear Case: inet_pov → MenuWebService**
- ii. **Menu Service SIF file:** This contains the fixed name, timeout and other values required for our ECC frameworks to invoke the service in this case Menu Service. **menuWSInfo.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceInfo name="menuWebServiceInfo" defaultProtocol="WS" overrideParam="testOverride"
defaultConfigFile="defaultServiceConf.xml">
    <protocol name="WS">
        <serviceAttribute name="jndiName"
value="cell/persistent/service/CardmemberInternetMenuWSFN1"/>
        <serviceAttribute name="url" value=""/>
        <serviceAttribute name="appName" value="Account Center - CMSCommon" />
        <serviceAttribute name="appShortName" value="AC- CMSCommon" />
        <serviceAttribute name="appGroup" value="www.discovercard.com - Acct Ctr" />
        <serviceAttribute name="appVersion" value="1.0.0" />
        <serviceAttribute name="serviceBusinessDelegateVersion" value="1.0.0" />
        <serviceAttribute name="sifPolicyRef" value="#policy"/>
    </protocol>
    <sifPolicy ID="policy">
        <operation name="getMenu">
            <operationAttribute name="timeout" value="4000"/>
        </operation>
    </sifPolicy>
</serviceInfo>
```

- iii. **AccountManagerAggregate jar:** Use Was6/WAS8 based on the application's websphere.

iv. **Menu Filter Configuration:** Configure Menu filter in **acSecurity-context.xml**

```

<security:http use-expressions="true" auto-config="false" entry-point-
ref="http403EntryPoint" create-session="stateless">
    <security:intercept-url pattern="/*" access="hasRole('ROLE_USER') "
/>
    <security:custom-filter before="PRE_AUTH_FILTER"
ref="dfsSecurityExceptionHandler" />
    <security:custom-filter position="PRE_AUTH_FILTER"
ref="dfsSecurityFilter" />
    <security:custom-filter after="PRE_AUTH_FILTER" ref="menuFilter" />
</security:http>

```

v. **Menu Layout configuration:** Configure in **acSecurity-context.xml**

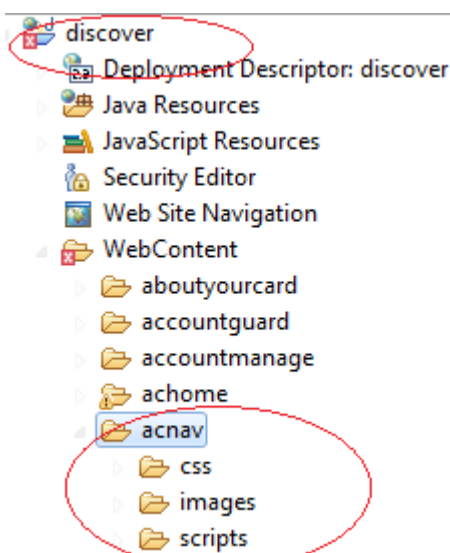
```

<bean id="menuFilter" class="com.discovercard.cardmembersvcs.common.security.MenuFilter">
    <property name="accountInfoLocation" value="#{dfsSecurityFilter.accountInfoLocation}"/>
    <property name="commonFTLPath"
value="/sys_data/websphere/inetcmsspcc/content/common/freemarker/menu/" />
</bean>

```

Note: Make sure you have acnav folder with css, images, and scripts.

Location: J:\Internet\Strut to Spring Migration\Common Header-Footer\<latest date folder>

vi. **jackson-all-1.8.1.jar:** Add Jackson jar. **Clear Case: inet_thirdparty**

14. Intercept Filter Setup

Every Web consumer application should have interceptable capability. Below is the configuration:

STEP 1: Update latest AccountCenterCommon.jar

STEP 2: Add intercept filter in acSecurity-context.xml

```
<security:http use-expressions="true" auto-config="false" entry-point-
ref="http403EntryPoint" create-session="stateless">
    <security:intercept-url pattern="/*" access="hasRole('ROLE_USER') " />
    <security:custom-filter before="PRE_AUTH_FILTER"
ref="dfsSecurityExceptionHandler" />
    <security:custom-filter position="PRE_AUTH_FILTER" ref="dfsSecurityFilter" />
    <security:custom-filter after="PRE_AUTH_FILTER" ref="compositeFilter" />
</security:http>

<bean id="compositeFilter" class="org.springframework.web.filter.CompositeFilter">
    <property name="filters">
        <list>
            <ref bean="outageFilter"/>
            <ref bean="interceptFilter"/>
            <ref bean="menuFilter"/>
        </list>
    </property>
</bean>

<bean id="interceptFilter"
class="com.discovercard.cardmembersvcs.common.intercept.InterceptFilter"/>
```

Note: in case of single filter, InterceptFilter can be added directly to security custom filter: (like below)

```
<security:http use-expressions="true" auto-config="false" entry-point-
ref="http403EntryPoint" create-session="stateless">
    <security:intercept-url pattern="/*" access="hasRole('ROLE_USER') " />
    <security:custom-filter before="PRE_AUTH_FILTER"
ref="dfsSecurityExceptionHandler" />
    <security:custom-filter position="PRE_AUTH_FILTER" ref="dfsSecurityFilter" />
    <security:custom-filter after="PRE_AUTH_FILTER" ref="interceptFilter" />
</security:http>

<bean id="interceptFilter"
class="com.discovercard.cardmembersvcs.common.intercept.InterceptFilter"/>
```

STEP 3: WSCR Change: Add URL in interceptconfig.cfg.properties (if needed for specific url, should be determined by the business)

File Name: interceptconfig.cfg.properties

File Path: /sys_data/websphere/inetcmsspcc/content/common/config

MAX_INTERCEPTS=1

/cardmembersvcs/achome/homepage=593142908,511490508

15. Spring Security - Cache Control Header Setup

Spring Security allows users to easily inject the default security headers to assist in protecting their application. This Cache Control header is one of them, considered best practice in DFS applications.

As users login, browser caches rendered pages, have evolved to include caches for secure connections as well. This means that a user may view an authenticated page, log out, and then a malicious user can use the browser history to view the cached page.

To help mitigate this Spring Security has added cache control support which will insert the following headers into you response.

```
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
```

Simply adding `<cache-control>` in `acSecurity-context.xml` will resolve browser's cache issue.

```
<security:http use-expressions="true" auto-config="false" entry-point-
ref="http403EntryPoint" create-session="stateless">
    <security:intercept-url pattern="/**" access="hasRole('ROLE_USER') " />
    <security:custom-filter before="PRE_AUTH_FILTER"
ref="dfsSecurityExceptionHandler" />
    <security:custom-filter position="PRE_AUTH_FILTER"
ref="dfsSecurityFilter" />
    <security:custom-filter after="PRE_AUTH_FILTER" ref="compositeFilter"
/>
    <security:headers>
        <security:cache-control />
    </security:headers>
</security:http>
```

Click [DFS Glossary](#) to add a new term or to verify words and acronyms used at Discover Financial Services.

[illegible]

Appendix A: Checklist

1. Setup spring security in local? (Please refer: Chapter #11 Spring Security Setup in Local Machine)
2. Identified lasted baseline from Production?
3. Created/Updated project structure in Clear case?
<https://dlife.discoverfinancial.com/docs/DOC-27106>
4. Checked aggregated jars/third party jars in Clear Case or Created symlink for those?
5. All latest PCC (.html, .shtml) and static (.js, .css) files from production have been retrieved?
(K:\pa_staging\PROD\arwintcm)
6. Freemarker Folder structure is defined within app. (Please refer: Chapter #1 - Convert JSP/HTML/Tiles into Freemarker Template)
7. All common servlet/helper/util/shared link code from CMSCommon/InternetCommonStruts has been identified?
8. All common servlet/helper/util/shared link code has been moved to Account Center Common?
9. Checked common components for Header/Footer/menu? (Please refer: Chapter #12 Common Header/Footer Setup)
10. Checked all global exceptions from struts configuration?
11. Checked Outage, Access, and Maintenance mode?
12. Checked Account number encryption in log4j file?
13. Checked Keep Session Alive? (web.xml)
14. Checked Database/ref binder? (web.xml/was6-8 admin console – resource reference)
15. All service calls/versions/internal/external (WAS6/WAS8) have been identified?
16. All methods from each service have been listed?
17. Current WAS Tuning script and TPS doc have been retrieved?
18. Current test case has been retrieved?
19. Checked Current Webserver/JVM/Nodes? (Please refer was6/8 admin console)
20. Checked current known production issue?
21. Checked Tivoli Setup for new slf4j logger? (Check with production support team)

Appendix B: Code Samples

Spring Configuration: (spring-context.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:security="http://www.springframework.org/schema/security"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:flow="http://www.springframework.org/schema/webflow-config"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:util="http://www.springframework.org/schema/util"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd
                           http://www.springframework.org/schema/mvc
                           http://www.springframework.org/schema/mvc/spring-mvc.xsd
                           http://www.springframework.org/schema/webflow-config
                           http://www.springframework.org/schema/webflow-config/spring-webflow-config-2.3.xsd
                           http://www.springframework.org/schema/security
                           http://www.springframework.org/schema/security/spring-security-3.1.xsd
                           http://www.springframework.org/schema/util
                           http://www.springframework.org/schema/util/spring-util.xsd
                           http://www.springframework.org/schema/mvc
                           http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

    <context:component-scan base-package="com.discovercard.cardmembersvcs.cash2ck"/>

    <mvc:annotation-driven/>

    <!-- Outage Utility Bootstrap Class -->
    <bean id="outageBootstrap"
class="com.discovercard.cardmembersvcs.common.outage.util.OutageUtilBootstrap">
        <constructor-arg index="0" type="String"
value="/sys_data/websphere/inetcmppcc/layout-config/outage-config.xml"/>
        <constructor-arg index="1">
            <value>60000</value>
        </constructor-arg>
    </bean>

    <!-- Access Level Bootstrap Class -->
    <bean id="accessBootstrap"
class="com.discovercard.cardmembersvcs.common.access.AccessBootstrap">
        <constructor-arg index="0" type="String"
value="/sys_data/websphere/inetcmppcc/layout-config/accessConfig.xml"/>
        <constructor-arg index="1">
            <value>60000</value>
        </constructor-arg>
    </bean>

    <!-- Template File Loader -->
    <bean id="cashToCheckPropertyFileLoader"
class="com.discovercard.cardmembersvcs.common.freemarker.AccountCenterPropertyFileLoader">
        <constructor-arg type="java.lang.String"
value="/sys_data/websphere/inetcmppcc/content/cash2ck/freemarker"/>
    </bean>

    <!-- Message - Property File -->
    <bean id="messageSource"
class="org.springframework.context.support.ResourceBundleMessageSource">
        <property name="basename" value="WEB-INF/resource/ApplicationResources" />
    </bean>

    <!-- Cash To Check BO -->
    <bean id="cash2CkBo" class="com.discovercard.cardmembersvcs.cash2ck.bo.Cash2CkBo">
        <property name="cash2CkEO" ref="cash2CkEO"/>
        <property name="dataServiceEO" ref="dataServiceEO"/>
        <property name="balanceTransferServiceEO" ref="balanceTransferServiceEO"/>
        <property name="acctServiceEO" ref="accountServiceEO"/>
    </bean>
</beans>
```



```

        <property name="promoPricingServiceEO" ref="promotionPricingServiceEO"/>
        <property name="eCommProfileServiceEO" ref="eCommProfileServiceEO"/>
        <property name="ePayServiceEO" ref="ePayServiceEO"/>
        <property name="memoServiceEO" ref="memoServiceEO"/>
        <property name="paymentServiceEO" ref="paymentServiceEO"/>
    </bean>

    <!-- Cash To Check EO -->
    <bean id="cash2CkEO" class="com.discovercard.cardmembersvcs.cash2ck.eo.Cash2CkEO" />

    <!-- Strong Auth BO -->
    <bean id="strongAuthBo" class="com.discovercard.cardmembersvcs.cash2ck.bo.StrongAuthBo">
        <property name="strongAuthEO" ref="strongAuthEO"/>
        <property name="accountManagerEO" ref="accountManagerEO"/>
    </bean>

    <!-- Strong Auth EO -->
    <bean id="strongAuthEO" class="com.discovercard.cardmembersvcs.cash2ck.eo.StrongAuthEO"
/>

    <!-- START - Service Delegate -->
    <bean id="accountManagerEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.AccountManagerEO">
        <constructor-arg name="accountManagerBDFactory" ref="accountManagerBDFactory"/>
    </bean>
    <bean id="accountServiceEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.AccountServiceEO">
        <constructor-arg name="accountServiceDelegateFactory"
ref="accountServiceDelegateFactory"/>
    </bean>
    <bean id="dataServiceEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.DataServiceEO">
        <constructor-arg name="dataServiceFactory" ref="dataServiceFactory"/>
    </bean>
    <bean id="balanceTransferServiceEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.BalanceTransferServiceEO">
        <constructor-arg name="balanceTransferServiceFactory"
ref="balanceTransferServiceFactory"/>
    </bean>
    <bean id="paymentServiceEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.PaymentServiceEO">
        <constructor-arg name="paymentServiceFactory" ref="paymentServiceFactory"/>
    </bean>
    <bean id="memoServiceEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.MemoServiceEO">
        <constructor-arg name="memosServiceDelegateFactory"
ref="memosServiceDelegateFactory"/>
    </bean>
    <bean id="ePayServiceEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.EPayServiceEO">
        <constructor-arg name="ePayServiceFactory" ref="ePayServiceFactory"/>
    </bean>
    <bean id="promotionPricingServiceEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.PromotionPricingServiceEO">
        <constructor-arg name="promotionPricingServiceFactory"
ref="promotionPricingServiceFactory"/>
    </bean>
    <bean id="eCommProfileServiceEO"
class="com.discovercard.cardmembersvcs.cash2ck.eo.ECommProfileServiceEO">
        <constructor-arg name="eCommProfileServiceDelegateFactory"
ref="eCommProfileServiceDelegateFactory"/>
    </bean>
    <!-- END - Service Delegate -->

    <!-- START - Service Factory -->
    <bean id="accountManagerBDFactory"
class="com.discovercard.cardmembersvcs.accountmanager.bd.AccountManagerBDFactory" factory-
method="getInstance"/>
    <bean id="accountServiceDelegateFactory"
class="com.discoverfinancial.accountservice.delegates.AccountServiceDelegateFactory" factory-
method="getInstance"/>

```

```

        <bean id="dataServiceFactory"
class="com.discovercard.cardmembersvcs.dataservice.bd.DataServiceFactory" factory-
method="getInstance"/>
        <bean id="balanceTransferServiceFactory"
class="com.discoverfinancial.bt.service.bd.BalanceTransferServiceFactory" factory-
method="getInstance"/>
        <bean id="paymentServiceFactory"
class="com.discoverfinancial.paymentservice.bd.PaymentServiceFactory" factory-
method="getInstance"/>
        <bean id="memosServiceDelegateFactory"
class="com.discoverfinancial.memoservice.delegates.MemosServiceDelegateFactory" factory-
method="getInstance"/>
        <bean id="ePayServiceFactory"
class="com.discovercard.cardmembersvcs.epayws.bd.EPayServiceFactory" factory-
method="getInstance"/>
        <bean id="promotionPricingServiceFactory"
class="com.discoverfinancial.ecc.promopricing.bd.PromotionPricingServiceFactory" factory-
method="getInstance"/>
        <bean id="eCommProfileServiceDelegateFactory"
class="com.discover.ecommprofile.delegate.ECommProfileServiceDelegateFactory" factory-
method="getInstance"/>
        <!-- END - Service Factory -->

        <flow:flow-builder-services id="flowBuilderServices" view-factory-
creator="viewFactoryCreator"/>

        <bean id="viewFactoryCreator"
class="org.springframework.webflow.mvc.builder.MvcViewFactoryCreator">
            <property name="viewResolvers">
                <list>
                    <ref bean="contentViewResolver"/>
                </list>
            </property>
        </bean>

        <!-- freemarker config -->
        <bean id="fmXmlEscape" class="freemarker.template.utility.XmlEscape"/>

        <bean id="freemarkerConfig"
class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">
            <property name="preTemplateLoaders">
                <list>
                    <bean id="cashToCheckTemplateLoader"
class="com.discovercard.cardmembersvcs.common.freemarker.AccountCenterFileTemplateLoader">
                        <constructor-arg type="java.lang.String"
value="/sys_data/websphere/inetcmppcc/content/cash2ck/freemarker"/>
                    </bean>
                </list>
            </property>
            <property name="freemarkerVariables">
                <map>
                    <entry key="xml_escape" value-ref="fmXmlEscape"/>
                </map>
            </property>
            <property name="freemarkerSettings">
                <props>
                    <prop key="template_update_delay">0</prop>
                </props>
            </property>
        </bean>

        <bean id="contentViewResolver"
class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
            <property name="mediaTypes">
                <map>
                    <entry key="html" value="text/html"/>
                    <entry key="ftl" value="text/html"/>
                    <entry key="xml" value="application/xml"/>
                    <entry key="json" value="application/json"/>
                </map>
            </property>
        </bean>

```

```

    </map>
  </property>
  <property name="favorPathExtension" value="true"/>
  <property name="defaultViews">
    <list>
      <bean class="org.springframework.web.servlet.view.json.MappingJacksonJsonView">
        <property name="prefixJson" value="true"/>
      </bean>
    </list>
  </property>
  <property name="viewResolvers">
    <list>
      <bean
class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver">
        <property name="cache" value="true"/>
        <property name="order" value="1"/>
        <property name="prefix" value=""/>
        <property name="suffix" value=".ftl"/>
        <property name="contentType" value="text/html;charset=UTF-8"/>
        <property name="exposeSpringMacroHelpers" value="true"/>
        <property name="requestContextAttribute" value="rc"/>
        <property name="exposeSessionAttributes" value="true"/>
      </bean>
    </list>
  </property>
</bean>
</beans>

```

Spring Security context: (acSecurity-context.xml)

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:security="http://www.springframework.org/schema/security"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
    http://www.springframework.org/schema/security
    http://www.springframework.org/schema/security/spring-security-3.1.xsd">

  <!-- Example of a high security http element and filter configuration -->
  <!-- <security:http use-expressions="true" auto-config="false" entry-point-
ref="http403EntryPoint" create-session="stateless">
    <security:intercept-url pattern="/highSecRequiredURI"
access="hasRole('ROLE_USER')" />
    <security:custom-filter position="PRE_AUTH_FILTER" ref="dfsSecurityFilterHigh" />
  </security:http>

  <bean id="dfsSecurityFilter"
class="com.discovercard.cardmembersvcs.common.security.DFSRequestAuthenticationFilter">
    <property name="cookieName" value="dcsession"/>
    <property name="authorizedSAStatus" value="HIGH"/>
    <property name="accountInfoLocation" value="acctInfo"/>
    <property name="authenticationManager" ref="authenticationManager" />
    <property name="headerInfoLocation" value="headerInfo" /> <!-- Common
header/footer -->
  </bean> -->

  <!-- This is the default security filter for normal security. In the case of a HIGH
security need, another filter calling
  the exact same class but with the HIGH value for authorizedStatus would be defined.
Then, a new http element would
  need to be created above the default to intercept those high need URLs. See above
example. -->
  <security:http use-expressions="true" auto-config="false" entry-point-
ref="http403EntryPoint" create-session="stateless">
    <security:intercept-url pattern="/*" access="hasRole('ROLE_USER')" />
    <security:custom-filter before="PRE_AUTH_FILTER" ref="dfsSecurityExceptionFilter"
/>

    <security:custom-filter position="PRE_AUTH_FILTER" ref="dfsSecurityFilter" />

```

```

</security:http>

<bean id="dfsSecurityFilter"
class="com.discovercard.cardmembersvcs.common.security.DFSRequestAuthenticationFilter">
    <property name="cookieName" value="dcsession"/>
    <property name="authorizedSAstatus" value="REGULAR"/>
    <property name="accountInfoLocation" value="acctInfo"/>
    <property name="authenticationManager" ref="authenticationManager" />
</bean>

<bean id="preauthAuthProvider"
class="org.springframework.security.web.authentication.preauth.PreAuthenticatedAuthenticationProvider">
    <property name="preAuthenticatedUserDetailsService">
        <bean id="userDetailsServiceWrapper"
class="org.springframework.security.core.userdetails.UserDetailsServiceByNameServiceWrapper">
            <property name="userDetailsService"
ref="customUserDetailsService"/>
        </bean>
    </property>
</bean>

<security:authentication-manager alias="authenticationManager">
    <security:authentication-provider ref="preauthAuthProvider" />
</security:authentication-manager>

<bean id="dfsSecurityExceptionHandler"
class="com.discovercard.cardmembersvcs.common.security.DFSExceptionHandler"/>
<bean id="customUserDetailsService"
class="com.discovercard.cardmembersvcs.common.security.CustomUserDetailsService"/>
<bean id="http403EntryPoint"
class="org.springframework.security.web.authentication.Http403ForbiddenEntryPoint"/>
</beans>

```

Outage Configuration: (outage-config.xml)

```

<?xml version="1.0" encoding="UTF-8"?>
<outageModes>
    <mode name="aclite" defaultAction="deny" dbColumnName="ACL">
        <UriException>/cardmembersvcs/loginlogout/</UriException>
        <UriException>/cardmembersvcs/achome/</UriException>
        <UriException>/dfs/accounthome/summary/</UriException>
        <UriException>/dfs/accounthome/tobankac/</UriException>
        <UriException>/dfs/accounthome/tobankacalt/</UriException>
        <UriException>/cardmembersvcs/statements/</UriException>
        <UriException>/cardmembersvcs/shopdiscover/</UriException>
        <UriException>/cardmembersvcs/smc/</UriException>
        <UriException>/cardmembersvcs/spendalyzer/</UriException>
        <UriException>/cardmembersvcs/mobile/app/achome/</UriException>
        <UriException>/cardmembersvcs/mobile/app/ems/</UriException>
        <UriException>/cardmembersvcs/mobile/app/statements/</UriException>
        <UriException>/cardmembersvcs/mobile/app/strongauth/</UriException>
        <UriException>/cardmembersvcs/mobile/app/epay/</UriException>
        <UriException>/cardmembersvcs/mobile/app/p2p/p2planding/</UriException>
        <UriException>/cardmembersvcs/mobile/app/personalprofile/profilelanding/</UriException>
        <UriException>/cardmembersvcs/mobile/app/rewards/rewardslanding/</UriException>

        <UriException>/cardsvcs/acs/session/</UriException>
        <UriException>/cardsvcs/acs/acct/</UriException>
        <UriException>/cardsvcs/acs/pymt/</UriException>
        <UriException>/cardsvcs/acs/stmt/</UriException>
        <UriException>/cardsvcs/acs/strongauth/</UriException>
        <UriException>/cardsvcs/acs/contact/</UriException>
        <UriException>/cardsvcs/acs/msghist/</UriException>
        <UriException>/cardsvcs/acs/shopd/</UriException>

        <!-- Adding EPay URLs that will be available in ACLite mode-->
        <UriException>/cardmembersvcs/epay/app/actionSubmitted/</UriException>
        <UriException>/cardmembersvcs/epay/app/addBank/</UriException>
        <UriException>/cardmembersvcs/epay/app/bankInfo/</UriException>
        <UriException>/cardmembersvcs/epay/app/cancel/</UriException>
    </mode>
</outageModes>

```

```

        <UriException>/cardmembersvcs/epay/app/deleteBank</UriException>
        <UriException>/cardmembersvcs/epay/app/doubleClick</UriException>
        <UriException>/cardmembersvcs/epay/app/edit</UriException>
        <UriException>/cardmembersvcs/epay/app/ha</UriException>
        <UriException>/cardmembersvcs/epay/app/openDates</UriException>
        <UriException>/cardmembersvcs/epay/app/overlayError</UriException>
        <UriException>/cardmembersvcs/epay/app/payment</UriException>
        <UriException>/cardmembersvcs/epay/app/pending</UriException>
        <UriException>/cardmembersvcs/epay/app/printableEnrollmentCancelation</UriException>
        <UriException>/cardmembersvcs/epay/app/printableUpdateBankInfoConfirm</UriException>
        <UriException>/cardmembersvcs/epay/app/revise</UriException>
        <UriException>/cardmembersvcs/epay/app/setup</UriException>
        <UriException>/cardmembersvcs/epay/app/to</UriException>
        <UriException>/cardmembersvcs/epay/app/updateEmail</UriException>
        <UriException>/cardmembersvcs/epay/app/updateInfo</UriException>
        <UriException>/cardmembersvcs/epay/app/viewPending</UriException>

    </mode>
    <mode name="paperless" defaultAction="allow" dbColumnName="POM">
        <UriException>/cardmembersvcs/paperless</UriException>
    </mode>
    <mode name="cardProductGroupCode" defaultAction="allow" dbColumnName="CPG">
        <UriException>/cardmembersvcs/extwar/ExtendedWarranty</UriException>
    </mode>
    <!-- Since Rewards application is going to take care of handling the rewards outage on its own and as of now
    there are no other applications which should not be available during Rewards outage mode, so this section is
    commented -->
    <!-- <mode name="rewards" defaultAction="" dbColumnName="ROM">
        <UriException></UriException>
    </mode> -->
</outageModes>

```

Access Configuration: (accessConfig.xml)

```

<?xml version="1.0" encoding="UTF-8"?>
<secureAccess>
    <access level="F" defaultAction="allow">
        <UriException>/cardmembersvcs/fraud</UriException>
        <UriException>/cardmembersvcs/mobile/app/fraud </UriException>
    </access>

    <access level="R1" defaultAction="deny">
        <UriException>/cardmembersvcs/fraud</UriException>
        <UriException>/cardmembersvcs/acctxfer/app/createUser</UriException>
        <UriException>/cardmembersvcs/acctxfer/app/verifyUser</UriException>
        <UriException>/cardmembersvcs/acctxfer/app/createUserPwd</UriException>
        <UriException>/cardmembersvcs/acctxfer/app/verifyUserPwd</UriException>
        <UriException>/cardmembersvcs/acctxfer/app/accountTransfer</UriException>
        <UriException>/cardmembersvcs/acctxfer/app/accountChanged</UriException>
        <UriException>/cardmembersvcs/acctxfer/app/accountFraud</UriException>
        <UriException>/cardmembersvcs/acctxfer/app/accountFraudXfer</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/updateSSOPassword</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/updateSSOUserId</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/updateSSOUserIdAndPwd</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/updateSSOPasswordSubmit</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/updateSSOUserIdSubmit</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/updateSSOUidAndPwdSubmit</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/tempPassword</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/tempPasswordSubmit</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/createUid</UriException>
        <UriException>/cardmembersvcs/personalprofile/pp/createUidSubmit</UriException>
        <UriException>/dfs/accounthome/summary</UriException>
        <UriException>/dfs/accounthome/tobankacalt</UriException>
        <UriException>/dfs/accounthome/tobankac</UriException>
        <UriException>/cardmembersvcs/reg/saConfirm</UriException>
    </access>

```

```

        <UriException>/cardmembersvcs/reg/confirmRegistration</UriException>
        <UriException>/cardmembersvcs/reg/confirmPwdReset</UriException>
        <UriException>/cardmembersvcs/reg/confirmation</UriException>
        <UriException>/cardmembersvcs/reg/goto</UriException>
        <UriException>/cardmembersvcs/reg/sgoto</UriException>
        <UriException>/cardmembersvcs/registration/reg/sgoto</UriException>
        <UriException>/keepalive</UriException>
        <UriException>/cardmembersvcs/rewards/invalidAccess</UriException>
    </access>
</secureAccess>

```

Web Configuration: (web.xml)

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd">
    <display-name>Cash2Check</display-name>

    <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-config/acSecurity-context.xml</param-value>
    </context-param>

    <servlet>
        <servlet-name>cash2ck</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring-config/spring-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>KeepSessionAlive</servlet-name>
        <servlet-
class>com.discovercard.cardmembersvcs.common.security.KeepSessionAlive</servlet-class>
        <init-param>
            <param-name>authorizedSAStatus</param-name>
            <param-value>REGULAR</param-value>
        </init-param>
    </servlet>

    <servlet>
        <servlet-name>MaintenanceInitServlet</servlet-name>
        <servlet-
class>com.discovercard.internet.common.maintenance.MaintenanceInitServlet</servlet-class>
        <init-param>
            <param-name>configFile</param-name>
            <param-value>/sys_data/websphere/inetcmppcc/layout-config/maintenance-
config.xml</param-value>
        </init-param>
        <init-param>
            <param-name>refreshInterval</param-name>
            <param-value>60000</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>cash2ck</servlet-name>
        <url-pattern>/cshHome</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>cash2ck</servlet-name>
        <url-pattern>/optionNotAvailable</url-pattern>
    </servlet-mapping>

```

```

</servlet-mapping>
<servlet-mapping>
  <servlet-name>cash2ck</servlet-name>
  <url-pattern>/app/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>KeepSessionAlive</servlet-name>
  <url-pattern>/keepalive</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>MaintenanceInitServlet</servlet-name>
  <url-pattern>/maintenanceservlet</url-pattern>
</servlet-mapping>

<!-- Spring security filter -->
<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>

<ejb-ref id="EjbRef_1200612578493">
  <ejb-ref-name>ejb/SessionInfoAO</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.discovercard.cardmembersvcs.session.ao.SessionInfoAOHome</home>
  <remote>com.discovercard.cardmembersvcs.session.ao.SessionInfoAO</remote>
  <ejb-link>SessionInfoApp.jar#SessionInfoAO</ejb-link>
</ejb-ref>

<ejb-ref id="EjbRef_1200686933172">
  <ejb-ref-name>ejb/AccountDatabaseEO</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.discovercard.cardmembersvcs.data.account.eo.AccountDatabaseEOHome</home>
  <remote>com.discovercard.cardmembersvcs.data.account.eo.AccountDatabaseEO</remote>
  <ejb-link>AccountInfoApp.jar#AccountDatabaseEO</ejb-link>
</ejb-ref>

<!-- START - DataBase Connection resource -->
<!-- NCS -->
<env-entry>
  <description>The NCS Oracle data source</description>
  <env-entry-name>ncs.dataSource.name</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>NcsOracleDataSource</env-entry-value>
</env-entry>

<resource-ref id="ResourceRef_1201880344276">
  <description></description>
  <res-ref-name>NcsOracleDataSource</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>

<!-- Internet Oracle -->
<env-entry>
  <description></description>
  <env-entry-name>dataSource.name</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>InternetOracleDataSource</env-entry-value>
</env-entry>

```

```

    <resource-ref id="ResourceRef_1201885706613">
      <description>This entry is for the card inventory app jar
dependency.</description>
      <res-ref-name>InternetOracleDataSource</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
      <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>

    <!-- DB2DataSource -->

    <resource-ref id="ResourceRef_1202681064567">
      <description></description>
      <res-ref-name>DB2DataSource</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
      <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>

    <!-- CacheOracleDataSource -->
    <env-entry>
      <env-entry-name>cache.dataSource.name</env-entry-name>
      <env-entry-type>java.lang.String</env-entry-type>
      <env-entry-value>CacheOracleDataSource</env-entry-value>
    </env-entry>

    <resource-ref id="ResourceRef_1203545086493">
      <description></description>
      <res-ref-name>CacheOracleDataSource</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
      <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>

    <!-- InetCacheDataSource -->
    <resource-ref id="ResourceRef_1202770938216">
      <res-ref-name>InetCacheDataSource</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
      <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>

    <!-- cicseci -->
    <resource-ref id="ResourceRef_1201886848069">
      <description></description>
      <res-ref-name>cicseci</res-ref-name>
      <res-type>javax.resource.cci.ConnectionFactory</res-type>
      <res-auth>Application</res-auth>
      <res-sharing-scope>Shareable</res-sharing-scope>
    </resource-ref>
    <!-- END - DataBase Container resource -->

  </web-app>

```

Service call using BO:

1. Controller Class:

```

@RequestMapping(value = "/confirmInfo", method = RequestMethod.GET )
    protected ModelAndView confirmInfo( HttpServletRequest request,
    HttpServletResponse response, @ModelAttribute("cashAdvInputForm")
    CashAdvInputForm cashAdvInputForm) throws TechnicalDifficultiesException,
    Exception {
        // Establish the name of the workflow for logging
        MDC.put("workflow", "Confirmation Page");

```



```

        long start = System.currentTimeMillis();

        .
        .
        .

        //Submit Cash Advance using BT service call
        InvocationStatus invocationStatus = getBO(request).submitCashAdvance(
        cashAdvanceInfo );

        // Check if validation status is okay
        if( invocationStatus.getStatusCode().equals(InvocationStatus.STATUS_OK) ) {
        .
        .
        .
        }
    }

```

2. BO Class

```

public class Cash2CkBo {

    private BalanceTransferServiceEO balanceTransferServiceEO;

    public InvocationStatus submitCashAdvance(CashAdvanceInfo cashAdvanceInfo)
    {

        InvocationStatus status = new InvocationStatus();
        status = balanceTransferServiceEO.submitCashAdvance(cashAdvanceInfo);
    }
}

```

3. EO Class

```

public class BalanceTransferServiceEO {
    private final Logger logger = LoggerFactory.getLogger(getClass());
    private boolean isDebug = logger.isDebugEnabled();

    //-- public static final String AUTH_REASON_CODE_APPROVED = "A ";
    public static final String VENDOR_ID = "59"; //this was 49, BT team
    wants us to change to 59

    private IBalanceTransferServiceBD iBalanceTransferServiceBD;

    public BalanceTransferServiceEO(BalanceTransferServiceFactory
    balanceTransferServiceFactory) {

        iBalanceTransferServiceBD = (IBalanceTransferServiceBD)
        balanceTransferServiceFactory.getBDInterface();

    }

    public InvocationStatus submitCashAdvance(CashAdvanceInfo cashAdvInfo) throws
    Cash2CkException{

```

```

        iBalanceTransferServiceBD.initialize(Cash2CkConstants.CONSUMER_APP_NAME
, userVO);

        CashToCheckingInputVO inputVO = new
CashToCheckingInputVO();
        inputVO.setAccountNumber(
cashAdvInfo.getAccountNumber());

        // IMPORTANT bt-bt service uses an implied decimal
point e.g $53.45 is send as a string of 5345
        NumberFormat nf =
NumberFormat.getInstance(Locale.US);
        nf.setMinimumFractionDigits(2);
        nf.setMaximumFractionDigits(2);

        inputVO.setTransferredAmount(nf.format(cashAdvInfo.getAmount().doubleValue()
).replaceAll("\\.", "").replaceAll(",", ""));

        inputVO.setUserID(userVO);
        inputVO.setVendorNumber(VENDOR_ID);
        //-- new fields for ECC Authorizations

        inputVO.setMerchantName(Cash2CkConstants.MERCHANGE_NAME);
        inputVO.setTerminalID(Cash2CkConstants.TERMINAL_ID);
        inputVO.setSystemID(Cash2CkConstants.SYSTEM_ID);
        inputVO.setMCC(Cash2CkConstants.MCC);

        inputVO.setCustomerName(cashAdvInfo.getCustomerName());

        CashToCheckingOutputVO outputVO =
iBalanceTransferServiceBD.createCashToChecking( inputVO );
        .
        .
        .

    }

```