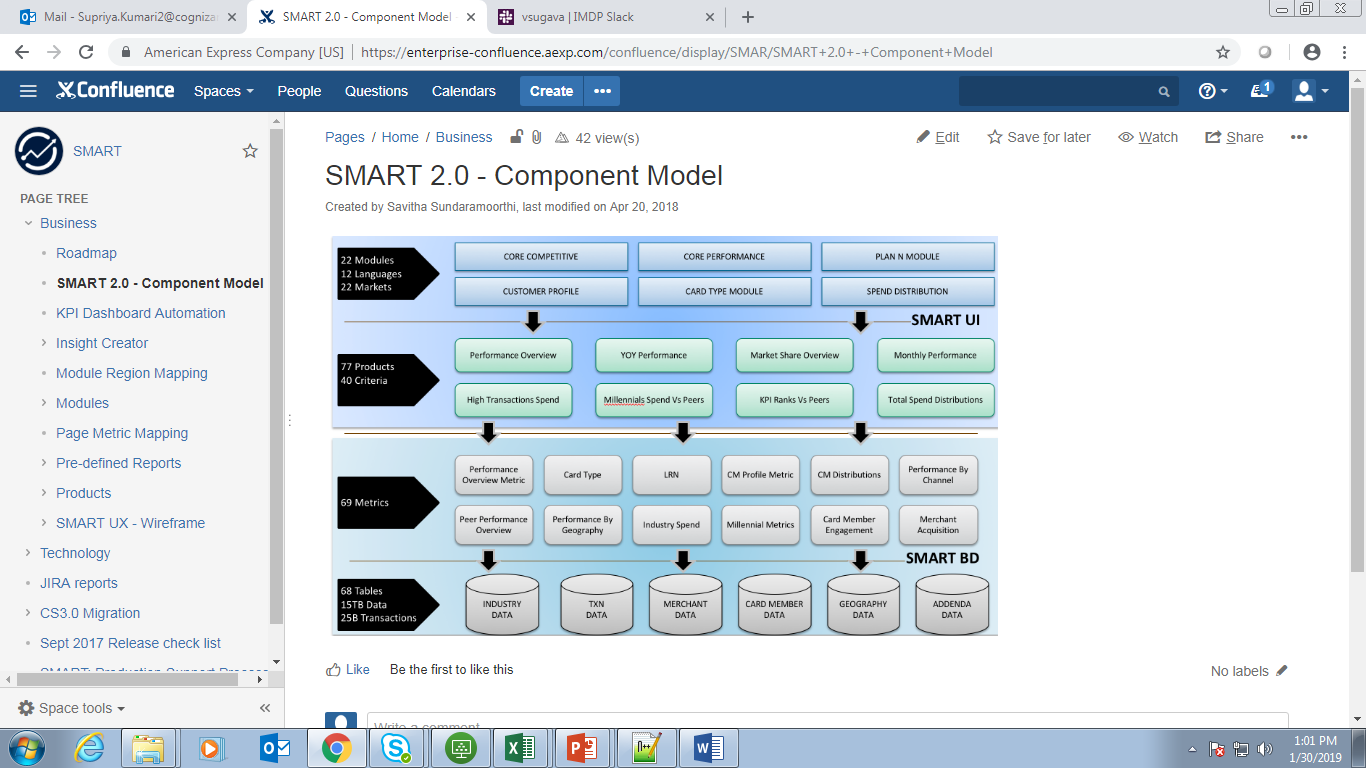
**Self Managed Analytic and Reporting Tool :**  SMART is a business intelligence reporting tool that provides standardized reporting and analytic through an easy to access, self service interface.

SMART is a self service global insights platform that offers merchant ready content with in-depth analysis and compelling insights into business performance, marketplace competition and customers.



Smart CMU translation utility is an interface for the Admin for enabling reports in newer language across Markets and regions.   
The purpose of the document is to make the User well aware of the intricacies of the Translation Module allowing optimum use and ease in the translation process.

Java Methods:

equalsIgnoreCase()

Spring Annotation used in project:

@Controller

@SessionAttributes(value = {SmartConstants.***SMART\_USER\_FORM***,"clientsForm"})

@Resource

@RequestMapping(value = "/welcome", method = {RequestMethod.***POST***, RequestMethod.***GET***})

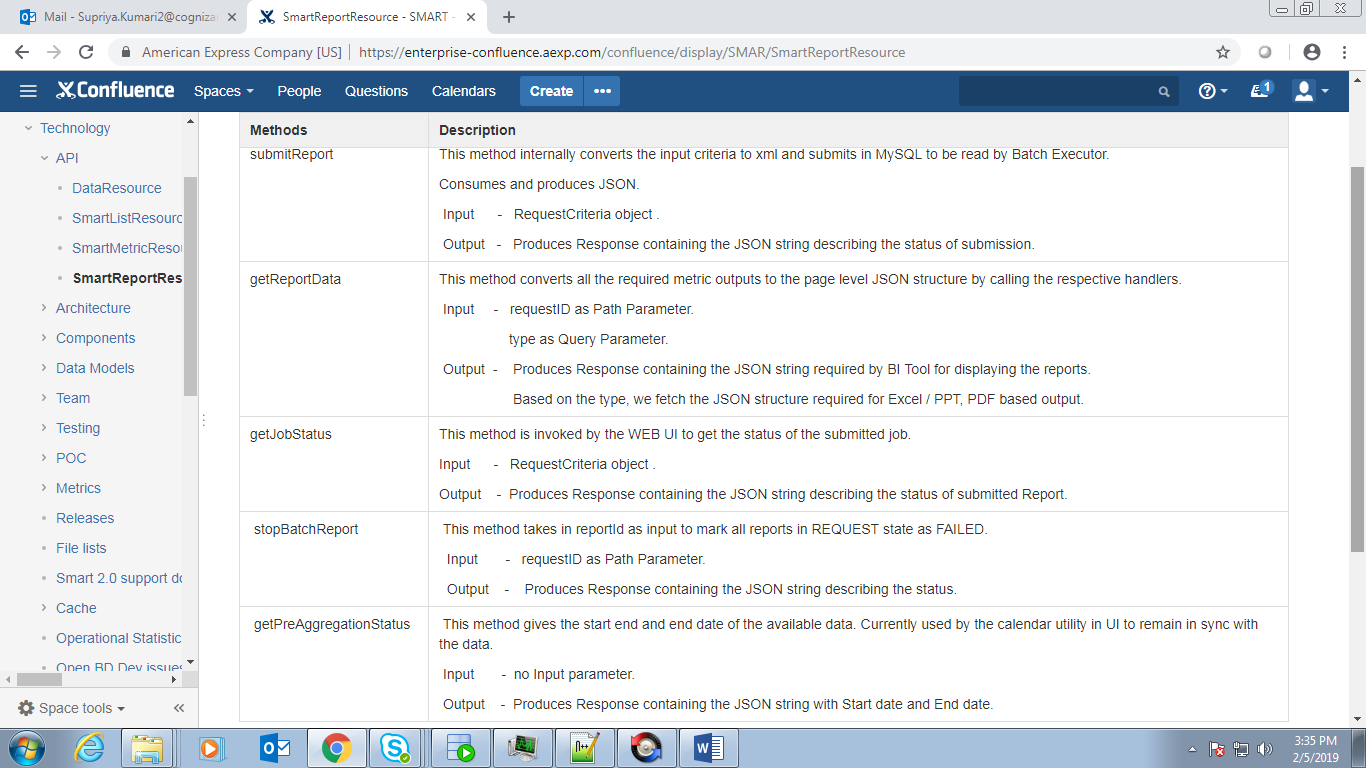
**Where u have used REST in ur project:**

To display the content in report, Jasper server needs input from controller(ICController).To send this json input to jasper we were used REST Call.

**REST Method wrote in smart:**

**A> SmartReportResource:** This class has methods that provides report level rest services for submitting, fetching, validating and checking status of a report.

Methods under this class:

**B> SmartMetricResource:** This class has methods that provides metric level rest services for submitting, fetching ,validating  and checking status of metrics.

**Exception**: We have created Project specific Parent Exception class by extending exception.

Now every exception is extending to SmartException only.

**public** **abstract** **class** SmartException **extends** Exception {

**private** String errorCode;

**private** SmartException( **final** String message )

{

**super**( message );

}

**public** SmartException( **final** String code, **final** String message )

{

**this**( message );

**this**.errorCode = code;

}

**public** SmartException( **final** String code, **final** String message, **final** Throwable cause )

{

**super**( message, cause );

**this**.errorCode = code;

}

**public** String getErrorCode()

{

**return** errorCode;

}

}

SmartIOException

SmartNoDataFoundException:

**public** **class** SmartNoDataFoundException **extends** SmartException {

**private** **static** **final** String ***ERR\_CODE*** = "";

**public** SmartNoDataFoundException(**final** String message )

{

**super**(***ERR\_CODE***, message );

}

**public** SmartNoDataFoundException(**final** String message, **final** Throwable cause )

{

**super**( ***ERR\_CODE***, message, cause );

}

}

**SmartRuntimeException:**

**public** **class** SmartRuntimeException **extends** RuntimeException {

**private** String errorCode;

**private** SmartRuntimeException( **final** String message )

{

**super**( message );

}

**public** SmartRuntimeException( **final** String code, **final** String message )

{

**this**( message );

**this**.errorCode = code;

}

**public** SmartRuntimeException( **final** String code, **final** String message, **final** Throwable cause )

{

**super**( message, cause );

**this**.errorCode = code;

}

**public** String getErrorCode()

{

**return** errorCode;

}

}

**SmartValidationException:**

**public** **class** SmartValidationException **extends** SmartException {

**private** **static** **final** String ***ERR\_CODE*** = "";

**public** SmartValidationException(**final** String message )

{

**super**(***ERR\_CODE***, message );

}

**public** SmartValidationException(**final** String message, **final** Throwable cause )

{

**super**( ***ERR\_CODE***, message, cause );

}

}

**ICController:**

@Controller

@RequestMapping(value = "/IC")

**public** **class** ICController {

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(ICController.**class**);

@Autowired

**private** IResourceBundleService resourceBundleService;

@Autowired

**private** IPageDesignService pageDesignService;

@Resource

**private** ITranslateMarkerService iTranslateMarkerService;

@Autowired

**private** ICarouselService iCarouselService;

@Autowired

**private** IMarkerService markerService;

@RequestMapping(value = "/getResourceBundles/{pageId}/{applicableRun}/{sourceLanguage}/{translationLanguage}", method = RequestMethod.***GET***)

**public** @ResponseBody

String getResourceBundle(ResourceBundleForm resourceBundleForm,

@PathVariable Map<String, String> pathVariables) {

***logger***.info("Inside getResourceBundle in ICController");

resourceBundleForm.setApplicableRun((pathVariables.get("applicableRun")));

resourceBundleForm.setSourceLanguage(pathVariables.get("sourceLanguage"));

resourceBundleForm.setDestinationLanguage(pathVariables.get("translationLanguage"));

resourceBundleForm.setPageId(pathVariables.get("pageId"));

resourceBundleService.populateResourceBundleDetails(resourceBundleForm);

**return** resourceBundleForm.getListResBndlPageData().toString();

}

@RequestMapping(value = "/getPageImage/{pageId}/{imageName}", method = RequestMethod.***GET***)

@SuppressWarnings("unchecked")

**public** **void** getImage(HttpServletResponse response, @PathVariable Map<String, String> pathVariables) {}

**For creation of json that needs to send to jasper:**

@RequestMapping(value = "/getPageJson/{pageId}/{applicableRun}/{sourceLanguage}/{translationLanguage}", method = {RequestMethod.***POST***, RequestMethod.***GET***})

**public** @ResponseBody

String getPageJson(ResourceBundleForm resourceBundleForm,

@PathVariable Map<String, String> pathVariables, JsonObject jsonObject) {

}

**private** Map<String, Object> getJasperPresentation(CreateReportBean crtRprtBean, JsonObject jsonObject){}