# Ícone Descrição gerada automaticamenteUma imagem contendo Ícone Descrição gerada automaticamente

# Integração Seam 2.1.2 e Jax-rs com Resteasy 2.0.1.GA no SICOP

A integração surge da necessidade de resolver um problema existente entre a comunicação dos sistemas SICAF X ORQUESTRADOR X SICOP, que antes realizada no padrão SOAP/RPC. Como solução, foi proposto a implementação do padrão Rest.

Alguns serviços afetados com essa integração e alteração de padrão arquitetural são:

* PVW
* Envio de circuito ao NV3
* Pedido de Viabilidade técnica

# Funcionamento Resteasy dentro do SICOP

Para que o Resteasy/JAX-RS funcione perfeitamente, foi necessário adicionar algumas configurações no arquivo **web.xml**. Todas as configurações adicionadas referente a especificação está devidamente identificada com o nome Resteasy. Outra configuração necessária para que os recursos se tornem acessíveis e visíveis ao Resteasy, é a inclusão do recurso criado dentro da lista de singletons do JAX-RS na classe **SicopJaxRsApplication**.

**Obs:**

Existe uma configuração para que recursos e provedores sejam registrados automaticamente, porém, em nenhum momento está configuração funcionou corretamente. Segundo a documentação, adicionando o parâmetro **resteasy.scan** como true dentro do arquivo web.xml, o Resteasy varreria o projeto e localizará todos os recursos e provedores.

# Utilização do JAXB

Definimos a utilização do json como protocolo de dados, visto que a transmissão tem menor latência e é mais leve se comparado ao xml, entretanto, durante os testes dos recursos encontramos algumas dificuldades para transmissão dos dados nesse formato, o resteasy não estava conseguindo converter o json recebido mesmo utilizando **@Produces**, **@Consumes** e content-type como application/json. Procurando uma solução que se enquadrasse em nossa situação, encontramos a anotação **@BadgerFish**, que simplesmente faz conversão de um xml para json. Para que a conversão do BadgerFish funcione, é necessário que os requests e responses dos recursos sejam configurados como xml, e fizemos isso utilizando a biblioteca Jettison JAXB.

Obs:

Houve um problema ao trabalhar com listas de objeto nesse formato, irei descrever como fizemos para solucionar.

1. Adicionar a anotação **@XmlAccessorType(XmlAccessType.*FIELD*)** na classe do objeto que é uma lista.
2. Na classe que contém a lista, será necessário configurar conforme imagem.Texto

   Descrição gerada automaticamente
3. Exemplo de um Json esperado para esse request. (vide imagem passo 2)

Linha do tempo

Descrição gerada automaticamente

# A URI padrão para o acesso aos recursos criados no sicop

Ambiente local

**http://localhost:8180/sicop/sicop/rest/{recurso}**

Ambiente de produção, homologação e desenvolvimento

**https://host/sicop/rest/{recurso}**

# Dependências adicionadas

* activation-1.1.jar
* commons-codec-1.2.jar
* commons-httpclient-3.1.jar
* httpclient-4.1.1.jar
* httpcore-4.0.1.jar
* jar.ws.rs-api-2.0.1.jar
* javax.json-api-1.0.jar
* jaxrs-api-2.0.1.GA.jar
* jboss-seam-resteasy-2.1.0.GA.jar
* jcip-annotations-1.0.jar
* jettison-1.2.jar
* lombok-0.11.6.jar
* resteasy-client-3.0.0.Final.jar
* resteasy-jaxb-provider-2.1.0.GA.jar
* resteasy-jaxrs-2.0.1.GA.jar
* resteasy-jettison-provider-2.0.1.GA.jar
* resteasy-json-p-provider-3.0.9.Final.jar
* scannotation-1.0.2.jar

Referências

<http://www.mastertheboss.com/jboss-frameworks/resteasy/resteasy-tutorial>

<https://docs.jboss.org/seam/2.1.2/reference/en-US/html/>

<https://docs.jboss.org/resteasy/docs/2.0.0.GA/userguide/html/index.html>