GT-mc^2: Minha Cloud Científica

Manual de usuário/desenvolvedor do PortEngin

Antônio Tadeu Azevedo Gomes

Bruno Correa e Vinicius de Macedo Moreira

16 de Outubro de 2012

Sumário

1 Configurações do CSGrid

Para acessar dados de um projeto CSGrid através de um portal são necessárias algumas configurações descritar no Wiki do SINAPAD. Estar configurações darão permissões para os portais executarem algoritmos e acessarem projetos através de um certificado gerado pela equipe e previamente cadastrado na ferramenta de governança do Openbus.

2 Sobre o Framework

O Framework de portais do SINAPAD tem como função oferecer uma interface web amigável e dinâmica para a submissão de algoritmos para os cluster disponíveis pelos CENAPADs em todo o Brasil. Essa interface é gerada dinamicamente através de arquivos de configuração (XML) que definem os atributos de entrada e saída dos algoritmos. Para garantir o funcionamento deste Framework também são necessárias configurações de outros arquivos XML que serão descritos nesta sessão.

Em sua infraestrutura, o Framework fornece uma área de projeto, onde os arquivos de entrada e saída ficam armazenados e a partir da qual se oferecem as opções de download, remoção, visualização e edição destes arquivos.

O Framework também oferece uma interface para o monitoramento de suas submissões, informando o estado das tarefas (aguardando, executando, erro ou executado), dados de entrada passados para os algoritmos (assim como os arquivos que foram passados como entrada para este) e os dados de saída das execuções. Esses dados são armazenados em um histórico de submissões e não serão alterados por nenhuma modificação na área de projeto do usuário.

O Framework do SINAPAD é dividido em quatro projetos (sinapad-framework-csgrid, sinapad-framework-parameters, sinapad-framework-core e sinapad-framework-web) , dos quais apenas o projeto sinapad-framework.web é editável pelo usuário final do portal.

2.1 sinapad-framework-csgrid

O Projeto sinapad-framework-csgrid tem como função oferecer uma API amigável para o acesso ao serviço de Autenticação e aos componentes Openbus, Opendreams e ProjectService. Estes componentes são responsáveis por fazer as submissões para o CSGrid e as consultas à sua área de projeto.

Um Javadoc dessa API pode ser encontrado em http://www.lncc.br/sinapad/csgrid-api

Os métodos dessa API devem ser utilizados apenas se a implementação core (sinapad-framework-core) não oferecer o método buscado, pois algumas rotinas implementadas pelo SINAPAD são oferecidas apenas pelo core do Framework.

2.1.1 Serviço de Autenticação

O serviço de autenticação oferecido pelo CSGrid acessa um servidor OpenLDAP. Esse serviço requer informações do host, porta e DN do servidor para ser inicializado. Esse serviço é utilizado para validar os usuários que podem acessar um determinado portal.

2.1.2 Componente Openbus

Para inicializar o componente Openbus, as informações de host e porta onde o barramento encontra-se ativo devem ser informados. Quando o serviço Openbus está inicializado é necessário uma autenticação que recebe o nome da entidade, uma chave privada e um certificado, esses dados são cadastrados na ferramenta de governança oferecida pelo Tecgraf para o controle de acesso ao Openbus. A gerência desta ferramenta é feita pela equipe do SINAPAD.

2.1.3 Componente Opendreams

Para inicializar um componente Opendreams é necessário que uma instância do Openbus já tenha sido inicializada e esteja autenticada. Esse serviço é inicializado através do nome do componente e sua versão.

2.1.4 Componente ProjectService (HDataService)

Esse serviço é inicializado como o Opendreams, necessitando também de uma instância Openbus já conectada e passando o nome e versão do componente.

2.2 sinapad-framework-core

O projeto sinapad-framework-core oferece todos os serviços necessários para o funcionamento correto de um portal e é responsável por fazer a interface entre os dados disponíveis no CSGrid e a interface web deste portal. É nessa camada que serviços adicionais oferecidos pelo SINAPAD estão adicionados.

Todos os serviços extras que atualmente não são oferecidos diretamente pelo CSGrid ou seus componentes que foram implementados pela equipe do SINAPAD estão disponibilizdos nesta API de maneira transparente como, por exemplo, o histório de arquivos. A API sinapad-framework-csgrid é um requisito da API core e é utilizada internamente para o acesso aos componentes e serviços oferecidos pelo CSGrid.

Um Javadoc desta API pode ser encontrado em http://www.lncc.br/sinapad/core-api/.

2.3 sinapad-framework-web

O projeto sinapad-framewor-web é o único projeto que deve ser modificado pelo usuário final. Esse projeto possui os arquivos JSP, JSPF, XML, HTML, CSS e JS que podem, ou devem, ser editados pelo usuário para modificar a interface web do portal ou o comportamento deste.

2.3.1 Configurando Ações

É possível configurar novas ações no Framework e adicioná-las ao fluxo de uma aplicação. As ações devem estender a classe BaseAction e implementar um método que retorne uma String e não receba nenhum parâmetro, como o exemplo:

```
public String act() {
 //do something
 return SUCCESS;
}
```

Também é necessária a criação de uma página de retorno (JSP) que mostrará o conteúdo retornado por uma ação e será indicada no arquivo struts.xml como um retorno válido para esta ação, como:

```
<action name="act"
        class="br.lncc.sinapad.action.act.ActAction"
        method="act">
  <result name="success">act.jsp</result>
</action>
```

No arquivo struts.xml encontram-se todas as ações executadas pelo Framework. Novsa ações podem ser chamadas através dos valores do atributo "name" como alvo do formulário.

Essa configuração pode ser feita também via anotações no código Java. O mesmo método anterior utilizando anotações ficaria:

```
@Action(value = "/act",
        results = { @Result(name = "success", location = "/act.jsp") })
public String act() {
  //do something
  return SUCCESS;
}
```

Para entender melhor sobre outras opções de configuração das ações ou como montar o JSP, é necessário estudar o manual do Framework Struts 2 que é utilizado pelo Framework do SINAPAD para controlar o fluxo da aplicação.

Os arquivos XML de configuração são validados através de arquivos XSD. A seguir são apresentadas descrições e exemplos desses arquivos.

2.3.2 authentication.xml

Nesse arquivo o usuário deve configurar a conexão com servidor LDAP (mesmo utilizado pelo CSGrid) que será usado pelo portal. É possível fazer a configuração utilizando SSL através da porta 636 e com um keystore válido. Caso a senha desse keystore não seja informada esse campo será ignorado.

```
<authentication-config>
  <ldap-ssl-enabled>false</ldap-ssl-enabled>
  <keystore-path>path-to-keystore</keystore-path>
  <keystore-password>keystore-password</keystore-password>
  <ldap-host>hostname.lncc.br</ldap-host>
  <ldap-port>389</ldap-port>
  <ldap-dn>dc=sinapad,dc=lncc,dc=br</ldap-dn>
</authentication-config>
```

A tag $< auth-type >$ possui os valores válidos: required, optional e none. Caso o SSL seja habilitado para a autenticação LDAP será necessário seguir os passos descrito no wiki do projeto, com informações de como validar o certificado utilizado por sua aplicação.

2.3.3 email.xml

O Framework fornece o envio de e-mails caso algum erro aconteça. Para enviar esse e-mail o responsável pelo portal deve configurar os dados apresentados abaixo.

```
<email-config>
  <smtp-host>smtp-host.lncc.br</smtp-host>
  <email-from>sinapad@lncc.br</email-from>
  <email-to>sinapad@lncc.br</email-to>
</email-config>
```

2.3.4 file-extension.xml

Para configurar os arquivos que o Framework deve conhecer é preciso informar os tipos de arquivos que são permitidos em seu portal, caso estes tipos estejam viculados a seus arquivos de entrada uma validação de extensão e, opcionalmente, uma validação personalizada será executada antes que o arquivo seja enviado para sua área de projetos. As extensões serão utilizadas para validar se o arquivo escolhido pelo usuário pode ser carregado dentro do portal e também se ele pode ser visualizado e editado. Para oferecer uma validação personalizada é necessário a implementação de um validador. Este validador deve implementar a interface br.lncc.sinapad.validator.FileValidator e implementar seus métodos. Quando um usuário tentar visualizar um arquivo, sua extensão será verificada, e caso sua visualização não seja permitida esse arquivo não será aberto. O motivo dessa verificação é que arquivos muito grandes podem travar seu navegador web, e deve ser aberto apenas localmente após o download. Essa opção pode ser desabilitada, liberando a visualização de qualquer arquivo da área de projetos.

Um exemplo de configuração deste XML pode ser visto a seguir.

```
<file-extension-config>
  <enabled>true</enabled>
  <file-type-config name="TEXT" can-view="true">
    <file-extension>txt</file-extension>
    <file-extension>log</file-extension>
   </file-type-config>
  <file-type-config name="LUA" can-view="true">
    <file-extension>lua</file-extension>
    <file-validator>
      br.lncc.sinapad.validator.impl.LuaFileValidator
    </file-validator>
  </file-type-config>
</file-extension-config>
```

2.3.5 logos.xml

Para facilitar e dinamizar a apresentação dos logos de cada portal, o Framework oferece um arquivo de configuração que é responsável por organizar os logotipos na interface web do portal. Estes logos são compostos da URL da instituição a qual o logo pertence e do nome da imagem deste logo. Para que o portal encontre essas imagens, elas devem ser colocadas em uma pasta, já existente no Framework, chamada logos.

Um exemplo de configuração deste XML pode ser visto a seguir.

```
<logos-config>
  <logo-config position="bottom">
    <logo-url>http://www.lncc.br/sinapad/</logo-url>
    <logo-img>sinapad.png</logo-img>
  </logo-config>
  <logo-config position="right">
    <logo-url>http://www.lncc.br/sinapad/</logo-url>
    <logo-img>sinapad.png</logo-img>
  </logo-config>
</logos-config>
```

2.3.6 openbus.xml

Para configurar a conexão com o servidor de execuções de algoritmo é necessário uma credencial fornecida pela equipe do SINAPAD. Esta credencial então, deve ser informada para o portal que realizará a conexão e validação desta para poder submeter seus algoritmos. Todos os dados para a configuração deste arquivo serão oferecidos pela própria equipe, basta informar o caminho onde seu portal encontrará estes arquivos no sistema de arquivos.

Um exemplo de configuração deste XML pode ser visto a seguir.

```
<open-bus-config>
  <open-bus-host>host.lncc.br</open-bus-host>
  <open-bus-port>2500</open-bus-port>
  <open-bus-entity>Entity</open-bus-entity>
  <open-bus-key>path/entity.key</open-bus-key>
  <open-bus-cert>path/AccessControlService.crt</open-bus-cert>
</open-bus-config>
```

2.3.7 portal.xml

Para configurar dados sobre o portal, o Framework oferece um arquivo que deve conter todos os dados do portal, tais como seu nome e abreviação, qual algoritmo deve ser utilizado e quais centros são habilitados para a execução.

Nesse arquivo é também configurada a localização do arquivo config.xml do CSGri em questão. Para garantir que o portal esteja de acordo com o algoritmo cadastrado no CSGrid, a estratégia é utilizar um link simbólico dentro do próprio servidor web (Apache Tomcat) do CSGrid para o config.xml real localizado na pasta "algorithms" do CSGrid. No arquivo portal.xml as tags $< config - xml - server >$ e $< config - xml - port >$ serão responsáveis por indicar onde o portal deve procurar por este arquivo.

Caso seu algorítmos possua mais do que uma versão, o atributo "allow-multiple-versions" deve receber o valor "true" e as tags "$< component - config >$" devem ser adicionadas para cada versão, assim como os componentes que tem acesso a essa versão.

Um exemplo de configuração deste XML pode ser visto a seguir.

```
<portal-config>
  <portal-acronym-name>Portal</portal-acronym-name>
  <portal-full-name>SINAPAD Portal Framework</portal-full-name>
  <portal-redmine-link>
    http://www.lncc.br/sinapad/redmine
  </portal-redmine-link>
  <algorithm-config allow-multiple-versions="false">
    <project-name>Portal</project-name>
    <algorithm-name>Algorithm</algorithm-name>
    <algorithm-default-version>1.0.0</algorithm-default-version>
    <algorithm-host>host.lncc.br</algorithm-host>
    <algorithm-port>8080</algorithm-port>
    <components-config>
      <component-config>
        <algorithm-version>1.0.0</algorithm-version>
        <open-dreams-config>
          <component-name>OpenDreams</component-name>
          <component-version>1.1.0</component-version>
        </open-dreams-config>
        <data-service-config>
          <component-name>ProjectService</component-name>
          <component-version>1.1.0</component-version>
        </data-service-config>
        <meta-scheduler-config>
          <component-name>MetaScheduler</component-name>
          <component-version>1.0.0</component-version>
        </meta-scheduler-config>
      </component-config>
    </components-config>
  </algorithm-config>
```

```
</portal-config>
```

2.3.8 modules.xml

Esse arquivo habilita os módulos de arquivos compartilhados, arquivos públicos, histórico de arquivos e usuário convidado disponíveis pelo portal. O módulo de arquivos compartilhados (Shared Files Module) e arquivos públicos (Public Files Module) não possuem nenhuma informação adicional, eles são apenas habilitado ou desabilitado. O módulo de histórico de algoritmos (Algorithm History Module) por sua vez, possui uma informação adicional que é o tempo (em dias) que os arquivos do histório ficarão armazenados no servidor. O valor em dias é um número inteiro positivo, caso seja -1 (ou qualquer outro número negativo) o histórico será mantido para sempre, caso o número de dias seja 0 então não haverá histórico da execução. O módulo para usuários convidados ativa o modo "guest" nos portais, habilitando qualquer usuário a se autenticar sem a necessidade de usuário e senha. Esse usuário pode ter algumas limitações descritas nos seus atributos de validação e só poderão submeter algoritmos após informar um e-mail em que receberão a informação do término de sua execução. Além disso uma validação "captcha" será requerida. Esses usuários não terão acesso à uma área de projetos e suas informações serão armazenadas temporariamente no histórico de submissões. Suas tarefas não estarão protegidas, sendo assim qualquer pessoa que possua o identificador de sua tarefa submetida terá acesso aos seus dados e poderá baixar os arquivos de entrada e saída dessa submissão. Os tipos de atributos disponíveis para validação de usuários convidados são "TextInteger", "TextDouble", "ListBoxInteger" e "ListBoxDouble".

```
<modules-config>
  <shared-files-module-config>
    <enabled>false</enabled>
  </shared-files-module-config>
  <public-files-module-config>
    <enabled>false</enabled>
  </public-files-module-config>
  <algorithm-history-module-config>
    <enabled>true</enabled>
    <history-time-life>10</history-time-life>
  </algorithm-history-module-config>
  <guest-user-module-config>
    <enabled>true</enabled>
    <submission-per-ip-config>
      <quantity>-1</quantity>
      <life-time-in-minutes>-1</life-time-in-minutes>
    </submission-per-ip-config>
    <attributes-config>
      <attribute-config>
```

```
        <name>LENGTHS</name>
        <type>ListBoxInteger</type>
        <error-message>Fragment lengths must be between 1 and 5 and list maximum size is
        <min>1</min>
        <max>5</max>
        <length>2</length>
      </attribute-config>
    </guest-user-module-config>
</modules-config>
```

2.3.9 Configurando a Aparência de um Portal

Para configurar a aparência de um portal, o Framework oferece arquivos CSS para todos os elementos que compõem a aplicação Web. Esses arquivos podem ser encontrados na raiz do do portal em uma pasta chamada "css". O arquivo sinapad.css é responsável pela configuração do layout de todos os elementos HTML deste. Esse layout especifica cores de links, cores de fundo e tamanho de fontes. O arquivo algorithm.css define as propriedades dos parâmetros do algoritmo, como tamanho de caixas de texto, espaçamento entre os parâmetros e suas cores. São esses dois arquivos que refletem as maiores mudanças no layout de um portal, e suas modificações não afetam seu comportamento. Os demais arquivos: lightbox.css, tree.css e rightcontext.css refletem mudanças no gerenciamento de arquivos, menu e visualização de imagens em um portal.

Arquivos JSP também podem ser modificadores porém, essas modificações, devem ser feitas por um designer que conheça a arquitetura de um portal, para que seu funcionamento seja garantido.

2.4 sinapad-framework-parameters

O projeto sinapad-framework-parameters oferece parâmetros estendidos para o configurador do CSGrid. Estes parâmetros tem como função passar algumas informações necessárias para que a interface web seja montada corretamente. Um conhecimento prévio do Manual do Configurador, oferecido pelo Tecgraf é requerido para o entendimento da utilização dos parâmetros estendidos utilizados pelo Framework de portais.

Foram adicionados alguns atributos aos seguintes tipos de parâmetros: "arquivo_de_entrada", "arquivo_de_saida", "lista_de_inteiros", "lista_de_reais" e "enumeracao".

2.4.1 arquivo_de_entrada

O atributo "base" foi adicionado a tag "arquivo_de_entrada" com a intenção de mostrar todos os elementos que encontram-se dentro desse diretório em uma visualização de arquivos de entrada. Ao contrário do CSGrid, na interface do portal, o usuário só pode visualizar os arquivos a partir do diretório "base" informado.

Os diretórios base devem ser informados no seguinte formato: "DIR_1/DIR_2/.../DIR_N", a partir do diretório do Projeto.

O atributo base pode receber seu valor dinamicamente através de qualquer valor selecionado na interface posteriormente. Para isso é necessário a sintaxe: #{NOME}, onde "NOME" é o nome do elemento que você deseja obter o valor selecionado, no momento que a opção de escolha de arquivos é selecionada.

2.4.2 arquivo_de_saida

Assim como nos arquivos de entrada, os arquivos de saída também possuem o atributo "base" com a mesma função.

2.4.3 lista_de_inteiros

Os atributos "tamanho" e "padrao" foram adicionador ao atributo "lista_de_inteiros".

O atributo "tamanho" tem a função de limitar superiormente o número de inteiros que pode ser adicionado à essa lista. Caso não seja especificado, essa lista permitirá qualquer quantidade de inteiros.

O atributo "padrao" tem a função de definir os valores já preenchidos desta lista por padrão. Esses valores devem ser informados no seguinte formato: "{NUM_1,NUM_2,...,NUM_N}".

2.4.4 lista_de_reais

Assim como a lista_de_inteiros essa lista funciona com números reais.

2.4.5 enumeracao

Os atributos "tipo" e "orientacao" foram adicionados à tag "enumeracao" com a intenção de exibir na interface web uma listbox ou um radio buttons. O atributo "tipo" pode possuir dois valores String: "selecao" ou "radio". O valor "selecao" criará uma listbox na interface web, enquanto o valor "radio" criará um conjunto de radio buttons. Por padrão, o valor "selecao" é configurado automaticamente caso nenhum valor seja passado. O atributo "orientacao" é utilizado em conjunto com o atributo "tipo" quando este possui o valor "radio". Ele pode possuir dois valores : "horizontal" e "vertical". Esse atributo definirá a orientação que os radio buttons serão mostrados na interface web.