Desenvolvedor

De PJe

O documento de arquitetura orienta o desenvolvimento no padrão do PJe: Acessar

O documento de gerenciamento de configuração traz orientações para a produção e guarda de artefatos e liberação de versões: Acessar

Alguns procedimentos necessários para o desenvolvimento no PJe:

Conteúdo

Pré-requisitos para ser desenvolvedor no PJe

Os pré-requisitos exigidos são:

- Conhecimento da linguagem de programação Java (nível de conhecimento mínimo: de intermediário a avançado).
- Conhecimento de persistência com JPA e Hibernate (nível de conhecimento mínimo: básico).
- Conhecimento da linguagem SQL do banco de dados PostgreSQL (nível de conhecimento mínimo: básico).
- Conhecimento de JSF (nível de conhecimento mínimo: básico).
- Conhecimento de JBoss Seam 2.2 (nível de conhecimento mínimo: básico).
- Habilidade de uso com a ferramenta Eclipse (nível de conhecimento mínimo: nível básico).

Os pré-requisitos desejáveis são:

- Habilidade de uso com a ferramenta Maven (nível de conhecimento mínimo: nível básico).
- Habilidade de uso com as ferramentas Git e GitLab (nível de conhecimento mínimo: noções gerais).
- Conhecimento do servidor de aplicação JBoss 5.1 (nível de conhecimento mínimo: básico).
- Conhecimento de padrões de projeto (design patterns) (nível de conhecimento mínimo: básico).
- Conhecimento de jBPM (nível de conhecimento mínimo: noções gerais).
- Habilidade de uso com a ferramenta de controle de demandas JIRA (nível de conhecimento mínimo: noções gerais).
- Ter tido algum contato de uso com o sistema PJe.

Configuração do ambiente de desenvolvimento

A seguir, algumas instruções para instalação do ambiente de desenvolvimento.

Roteiro para Instalação e Configuração do Sistema Operacional

- Recomenda-se o uso do Ubuntu para as máquinas de desenvolvimento, visto que seu desempenho é bastante superior ao do MS-Windows
- Procure usar uma interface leve para o ambiente de trabalho. Uma alternativa interessante pode ser obtida em:

```
$ sudo apt-get install gnome-session-fallback
```

- Recomenda-se a atualização do driver gráfico a partir de https://01.org/linuxgraphics/downloads/2013/intelr-linux-graphics-installer-version-1.0.1

Roteiro para Instalação do Java (Sistema Linux 64 bits)

Oracle JDK

- Recomenda-se a instalação do Oracle JDK em função de padronização com o JDK para sistema operacional MS-Windows e com o procedimento demonstrado em http://www.cnj.jus.br/wikipje/images/e/ec/CorrecaoCertificadoCNJ.mp4
- Certifique-se que sua versão é de 64 bits

```
$ file /sbin/init
```

- Verifique se há alguma versão do Java instalada

```
$ java -version
```

- Se houver um Open JDK instalado, remova-o:

```
$ sudo apt-get purge openjdk-\*
```

- Crie uma pasta para instalação do Java

```
$ sudo mkdir -p /usr/local/java
```

- Faça o download do JRE e do JDK, ambos compatíveis com Linux 64 bits
- Copie os arquivos baixados para a pasta criada

```
$ cd ~/Downloads
$ sudo -s cp -r arquivoJDK.tar.gz /usr/local/java
$ sudo -s cp -r arquivoJRE.tar.gz /usr/local/java
```

- Ajuste as permissões e descompacte os arquivos

```
$ cd /usr/local/java
$ sudo -s chmod a+x  arquivoJDK.tar.gz
$ sudo -s chmod a+x  arquivoJRE.tar.gz
$ sudo -s tar -xvzf arquivoJDK.tar.gz
$ sudo -s tar -xvzf arquivoJRE.tar.gz
```

- Edite e acescente as informações sobre o Java no arquivo profile

```
$ sudo gedit /etc/profile
```

```
Trecho a ser inserido no arquivo profile

JAVA_HOME=/usr/local/java/seuJDK
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
JRE_HOME=/usr/local/java/seuJRE
PATH=$PATH:$HOME/bin:$JRE_HOME/bin
export JAVA_HOME
export JRE_HOME
export PATH
```

- Informe a localização do Java para o sistema

```
$ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/local/java/seuJRE/bin/java" 1
$ sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/local/java/seuJDK/bin/javac" 1
$ sudo update-alternatives --install "/usr/bin/javaws" "javaws" "/usr/local/java/seuJRE/bin/javaws" 1
```

- Informe o Java padrão para o sistema

```
$ sudo update-alternatives --set java /usr/local/java/seuJRE/bin/java
$ sudo update-alternatives --set javac /usr/local/java/seuJDK/bin/javac
$ sudo update-alternatives --set javaws /usr/local/java/seuJRE/bin/javaws
```

- Atualize o path

```
$ . /etc/profile
```

- Verifique a instalação

```
$ java -version
$ javac -version
```

- Reinicialize o sistema e a instalação estará concluída

OpenJDK

```
$ sudo apt-get purge openjdk* icedtea*
$ sudo apt-get install openjdk-7-demo openjdk-7-jdk openjdk-7-jre openjdk-7-jre-headless icedtea-7-jre-jamvm icedtea-7-plugin
$ update-java-alternatives -l
$ sudo update-java-alternatives -s java-1.7.0-openjdk-amd64
```

Roteiro para Configuração do Navegador Firefox

- Crie um link simbólico para no diretório de plugins do Firefox /usr/lib/firefox-addons/plugins para /usr/local/java/seuJRE/lib/amd64/libnpjp2.so

```
$ sudo ln -sv /usr/local/java/seuJRE/lib/amd64/libnpjp2.so /usr/lib/firefox-addons/plugins
```

- No menu Ferramentas, selecione complementos (add-ons) > plugins > Java (TM) plugin > sempre ativar

Roteiro para Instalação e Configuração do Cliente para Certificados Digitais

Ubuntu 64 bits

- Para todos os modelos de eToken, baixe o arquivo em http://vqv.com.br/cnj/libhal1_0.5.14-8_amd64.deb e instale-o

```
$ cd ~/Downloads
$ sudo dpkg -i libhal1_0.5.14-8_amd64.deb
```

- Há vários modelos de eToken. Neste roteiro são tratados o eToken Pro Aladdin e o eToken GD Starsign. Para verificar o modelo de seu eToken, conecte-o a uma USB e use o comando:

```
$ lsusb
```

Para eToken Pro Aladdin:

- Baixe http://vqv.com.br/cnj/cliente-safenet.tar.gz
- Descompacte, ajuste as permissões e execute o script de instalação

```
$ cd ~/Downloads
$ tar -zvxf cliente-safenet.tar.gz
$ chmod 700 install_SafenetAuthenticationClient-8.1.0-4_amd64.deb.sh
```

```
$ sudo ./install_SafenetAuthenticationClient-8.1.0-4_amd64.deb.sh
$ sudo apt-get -f install
```

- Crie os links simbólicos para as bibliotecas necessárias

```
$ cd /usr/lib
$ sudo ln -sv /usr/lib64/libeToken.so /usr/lib/libeToken.so.8
$ sudo ln -sv /usr/lib64/libeTokenUI.so /usr/lib/libeTokenUI.so.8
```

- Acesse uma instalação do PJe.
- Na primeira tentativa, você será solicitado a selecionar o driver. Escolha:

```
/usr/lib64/libeToken.so
```

Para eToken Giesecke & Devrient Starsign:

- Baixe http://vqv.com.br/cnj/safesignidentityclient_3.0.77-Ubuntu_amd64.deb
- Instale o cliente

```
$ cd ~/Downloads
$ sudo dpkg -i SafenetAuthenticationClient-8.1.0-4_amd64.deb
$ sudo apt-get -f install
```

- Na primeira vez que acessar o PJe, selecionar o driver

```
/usr/lib/libaetpkss.so.3.0.2528
```

Para alternar entre diferentes tipos de eTokens

- Renomeie o driver, insira o próximo eToken e carregue o PJe.
- Ao acessar o PJe, você será solicitado a escolher um novo driver.
- Selecione conforme o tipo de eToken, conforme explicado anteriormente.

Microsoft Windows 7 64 bits

- Neste roteiro são tratados o eToken Pro Aladdin e o eToken GD Starsign. Instuções completas sobre esses e os demais modelos de eToken podem ser encontradas em http://www.certisign.com.br/atendimento-suporte/downloads
- O plugin Java deve estar instalado e ativo no navegador

Para eToken Pro Aladdin:

- Baixe e execute o driver em http://vqv.com.br/cnj/Safenet_Authentication_Client_8.1SP2-x64.msi
- Reinicie o navegador, insira o eToken e acesse o PJe

Para eToken Giesecke & Devrient Starsign:

- Baixe e execute o driver a partir de http://vqv.com.br/cnj/GDsetupStarsignCUTx64.exe
- Baixe e execute o cliente a partir de http://vqv.com.br/cnj/SafeSign_Identity_Client-Standard-3.0.87-general-x64-win-admin-std-vc8.exe
- Reinicie o navegador, insira o eToken e acesse o PJe

Roteiro para Instalação do JBoss 5.1

- Baixar e instalar (descompactar) o JBoss 5.1.0.GA (download em http://www.jboss.org/jbossas/downloads/).
- Alternativamente, usar o Jboss pré-configurado fornecido pela equipe de desenvolvimento do PJe ...
- Na pasta server\pje\lib do Jboss, copie o driver jdbc do Postgres, e do Oracle, se for o caso
- Verifique mais adiante no roteiro de instalação do Eclipse as configurações necessárias para uso do Jboss no Eclipse
- Para usar o JDK7, é preciso configurar o Jboss, conforme trecho de código a seguir:

```
Na pasta \server\pje\conf\bootstrap, arquivo profile.xml (<parameter class="java.io.File">)

       <bean name="AttachmentStore" class="org.jboss.system.server.profileservice.repository.AbstractAttachmentStore">
          <constructor>
            <parameter class="java.io.File">
               <inject bean="BootstrapProfileFactory" property="attachmentStoreRoot" />
            </parameter>
          </constructor>
       </bean>
```

Roteiro para Instalação e Configuração do Sistema Gerenciador de Banco de Dados (desenvolvimento)

Instalação do SGBD postgres

- Execute o seguinte comando no terminal

```
$ sudo apt-get install postgresql pgadmin3
```

Geração de uma senha conhecida para o usuário padrão postgres

- Alterar a senha do usuário postgres no sistema operacional

```
$ sudo passwd postgres
```

- Alternar para o usuário postgres

```
$ su postgres
```

- Alterar a senha do usuário postgres no SGBD

```
$ psql -c "ALTER USER postgres WITH PASSWORD 'nova_senha'" -d template1
```

- Acessar o SGBD usando o cliente pgAdmin III para certificação de que o SGBD está acessível

Ajuste de max_prepared_transactions

Configure **max_prepared_transactions** no arquivo postgresql.conf valorado com 10

Atenção! Após a modificação, reinicialize o SGBD para que tenha efeito

```
$ locate postgresql.conf
$ cd /etc/postgresql/9.1/main
$ sudo gedit postgresql.conf

No MS-Windows,o arquivo fica em: C:\Program Files\PostgreSQL\9.3\data
```

Atenção! Retire o caractere de comentário # do início da linha

```
Substituir #max_prepared_transactions=0 por max_prepared_transactions=10
```

Criação da base de dados

Criar três bases de dados (pje, pje_bin e pje_log) com os seguintes parâmetros configurados, na aba Definition

```
encoding: LATIN1
template: template0
collation: C
character type: C
```

Restauração dos dumps

Restaurar os dumps das três bases com a seguinte instrução

```
$ pg_restore -U postgres -h localhost -v -Fc -d nome_base arquivo.dump
```

Criação dos Datasources

- Criar um arquivo xxxxxx-ds.xml para definição dos datasources, conforme exemplificado a seguir. Observe que cada datasource tem um jndi-name.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE datasources
    PUBLIC "-//JBoss//DTD JBOSS JCA Config 1.5//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-ds_1_5.dtd">

<datasources>

        <xa-datasource>
                <jndi-name>PJE_DESCANSO_DS</jndi-name>
                <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
                <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
                <xa-datasource-property name="PortNumber">5432</xa-datasource-property>
                <xa-datasource-property name="DatabaseName">pje</xa-datasource-property>
                <new-connection-sql>set search_path = public, acl, core, client, criminal, jt</new-connection-sql>
                <user-name>postgres</user-name>
                <password>postgres</password>
                <min-pool-size>3</min-pool-size>
                <max-pool-size>10</max-pool-size>
                <track-connection-by-tx />
                <metadata>
                        <type-mapping>PostgreSQL 8.0</type-mapping>
                </metadata>
        </xa-datasource>
```

```
        <xa-datasource>
                <jndi-name>PJE_DESCANSO_BIN_DS</jndi-name>
                <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
                <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
                <xa-datasource-property name="PortNumber">5432</xa-datasource-property>
                <xa-datasource-property name="DatabaseName">pje_bin</xa-datasource-property>
                <new-connection-sql>set search_path = public, client, core</new-connection-sql>
                <user-name>postgres</user-name>
                <password>postgres</password>
                <min-pool-size>3</min-pool-size>
                <max-pool-size>10</max-pool-size>
                <track-connection-by-tx />
                <metadata>
                        <type-mapping>PostgreSQL 8.0</type-mapping>
                </metadata>
        </xa-datasource>

        <xa-datasource>
                <jndi-name>PJE_DESCANSO_LOG_DS</jndi-name>
                <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
                <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
                <xa-datasource-property name="PortNumber">5432</xa-datasource-property>
                <xa-datasource-property name="DatabaseName">pje_log</xa-datasource-property>
                <new-connection-sql>set search_path = public</new-connection-sql>
                <user-name>postgres</user-name>
                <password>postgres</password>
                <min-pool-size>3</min-pool-size>
                <max-pool-size>10</max-pool-size>
                <track-connection-by-tx />
                <metadata>
                        <type-mapping>PostgreSQL 8.0</type-mapping>
                </metadata>
        </xa-datasource>

</datasources>
```

Outras informações sobre a instalação e configuração do SGBD (ambiente de produção)

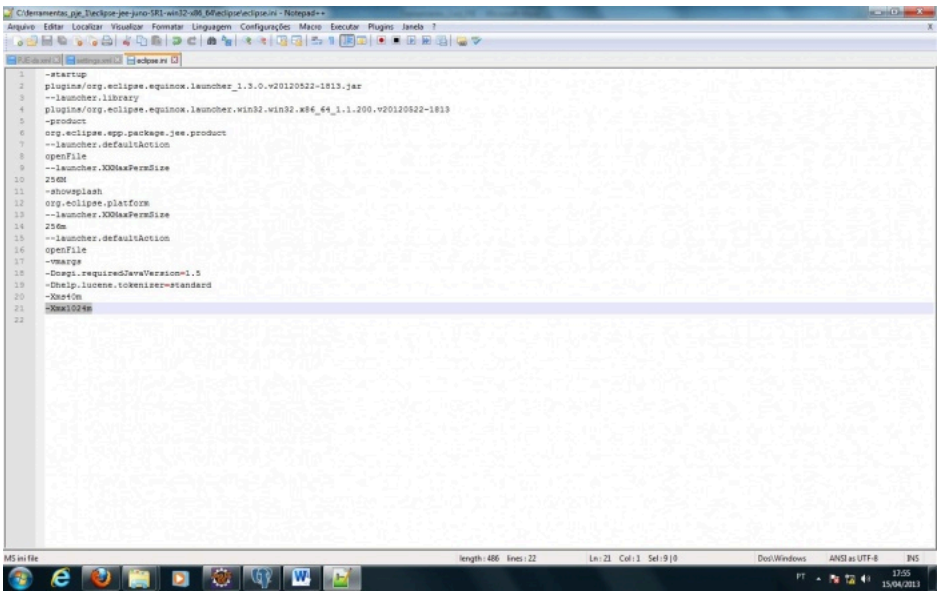
Instalação do Sistema gerenciador de banco de dados

Roteiro para Instalação do Eclipse e de seus plugins

- Baixar (http://www.eclipse.org/downloads/) e descompactar o Eclipse Kepler (Eclipse IDE for Java EE Developers)

```
Atenção! Ao instalar plugins ou gerar/configurar a chave para acesso ao GIT, sempre reinicialize a IDE, mesmo que não seja solicitado
```

- Ajustar as preferências do Eclipse conforme instruções em Preferências do Eclipse
- Aumentar o tamanho máximo do heap space para 1024 MB. Essa configuração pode ser feita no arquivo eclipse.ini com a alteração do valor -Xmx de 512m (padrão) para 1024m, conforme exemplo a seguir:



- Gerar um par de chaves SSH para acessar o repositório Git do PJe, conforme instruções em Geração do Par de Chaves
- Configurar no GitLab a chave pública gerada no passa anterior, conforme instruções em Configuração da Chave Pública no Servidor Git
- Gerar uma cópia Git local do repositório do PJe conforme descrito em Obtendo a cópia do repositório do PJe
- Criar um projeto a partir do repositório clonado no item anterior, conforme instruções em Criando o projeto a partir do repositório clonado
- Configurar o acesso ao artfactory do CNJ no arquivo setting.xml, conforme instruções em Dependências do Maven
- Remover as validações desnecessárias, conforme descrito em Removendo as validações desnecessárias
- Adicionar o servidor Jboss ao Eclipse
- Adicionar o pje-web ap Jboss

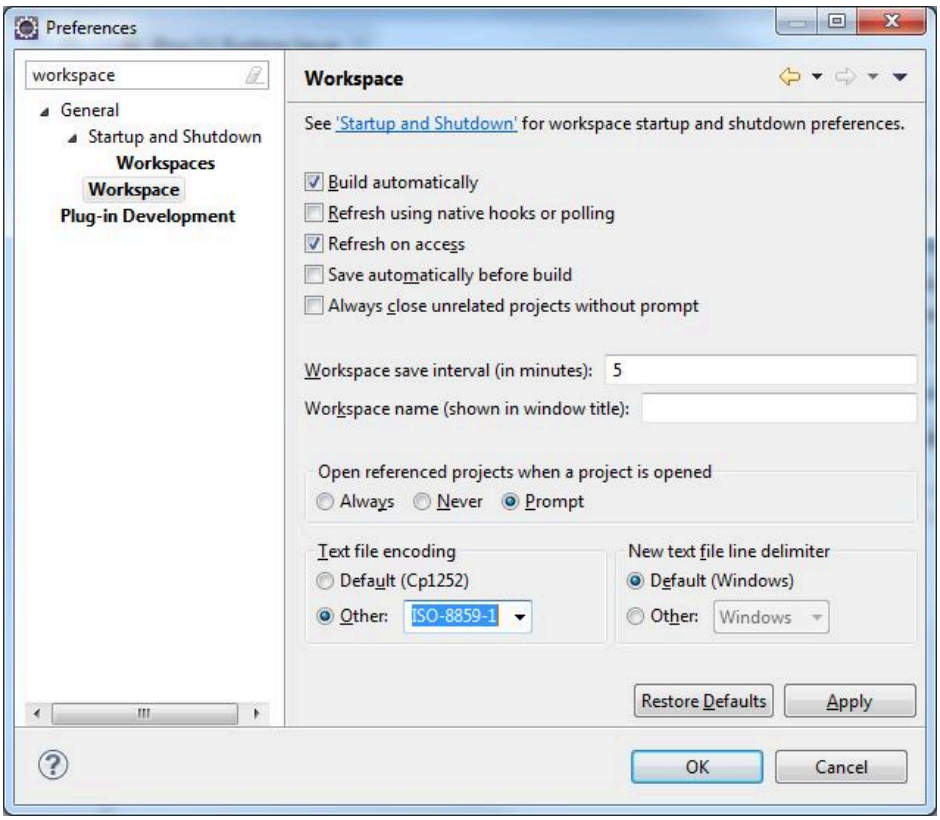
- Ativar o Jboss

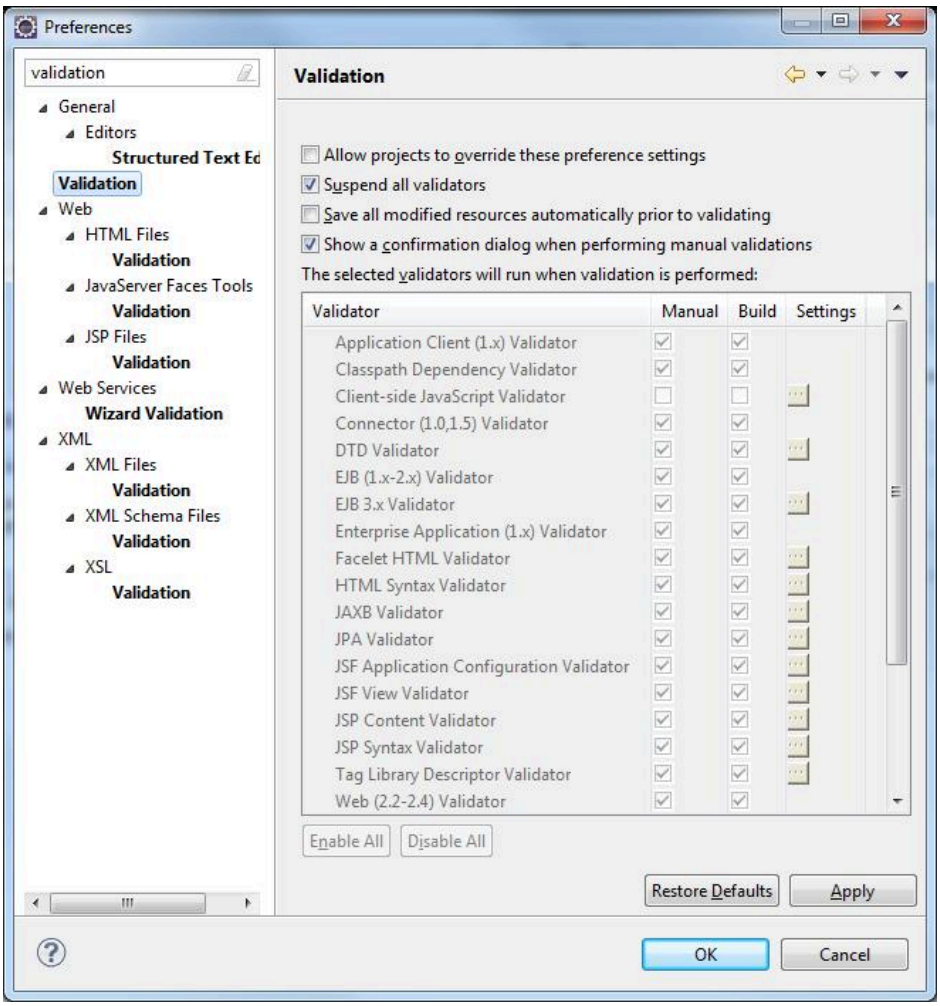
```
Se todos os procedimentos foram executados sem erros, seu ambiente está pronto
```

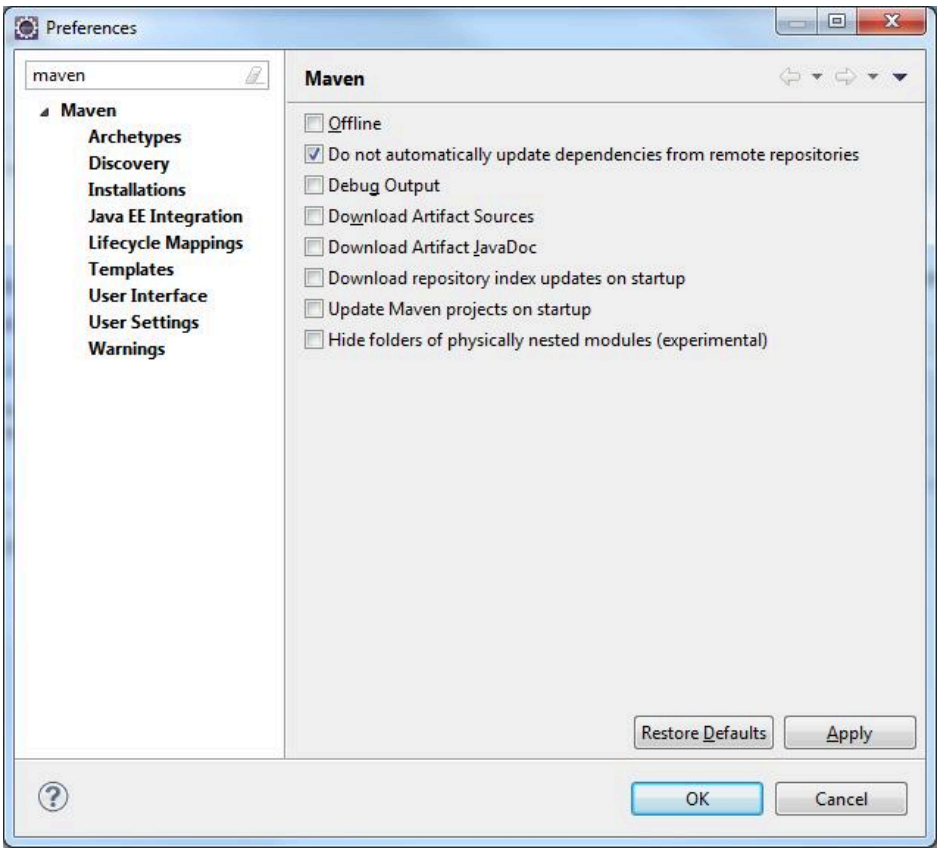
Preferências do Eclipse

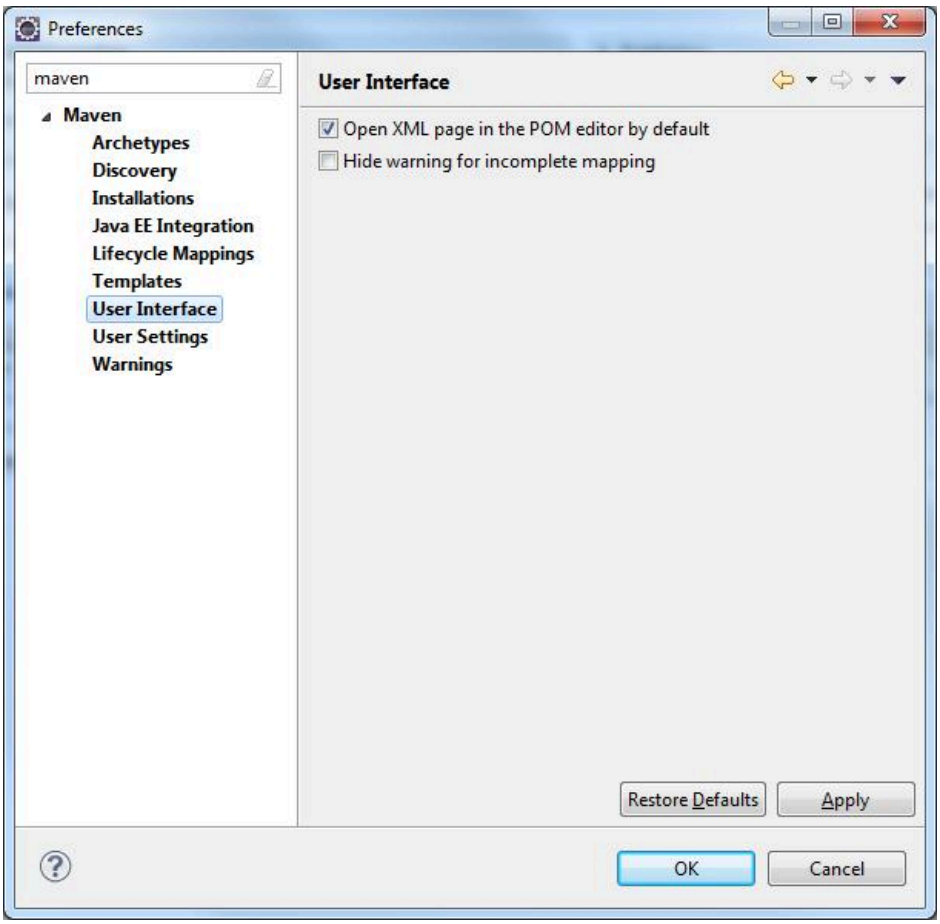
- Ir ao menu “Window > Preferences” e configurar as seguintes preferências do Eclipse:

```
Desmarcar limite de console: Window->Prferences->Debug/Run Console->Limit Console Output
```

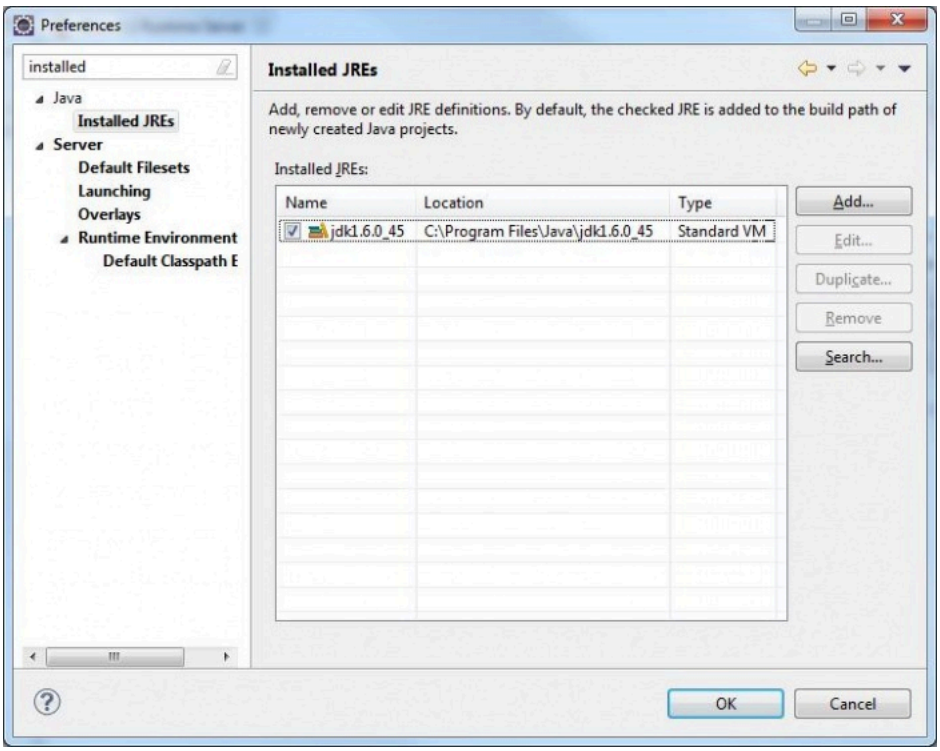



Preferences
validation
General
Editors
Structured Text Ed
Validation
Web
HTML Files
Validation
JavaServer Faces Tools
Validation
JSP Files
Validation
Web Services
Wizard Validation
XML
XML Files
Validation
XML Schema Files
Validation
XSL
Validation
Validation
Allow projects to override these preference settings
Suspend all validators
Save all modified resources automatically prior to validating
Show a confirmation dialog when performing manual validations
The selected validators will run when validation is performed:
Validator
Manual
Build
Settings
Application Client (1.x) Validator
Classpath Dependency Validator
Client-side JavaScript Validator
Connector (1.0,1.5) Validator
DTD Validator
EJB (1.x-2.x) Validator
EJB 3.x Validator
Enterprise Application (1.x) Validator
Facelet HTML Validator
HTML Syntax Validator
JAXB Validator
JPA Validator
JSF Application Configuration Validator
JSF View Validator
JSP Content Validator
JSP Syntax Validator
Tag Library Descriptor Validator
Web (2.2-2.4) Validator
Enable All
Disable All
Restore Defaults
Apply
OK
Cancel


Preferences
maven
Maven
Archetypes
Discovery
Installations
Java EE Integration
Lifecycle Mappings
Templates
User Interface
User Settings
Warnings
Maven
Offline
Do not automatically update dependencies from remote repositories
Debug Output
Download Artifact Sources
Download Artifact JavaDoc
Download repository index updates on startup
Update Maven projects on startup
Hide folders of physically nested modules (experimental)
Restore Defaults
Apply
OK
Cancel



- Adicionar o JDK e remover as JREs existentes (A JDK pode ser obtida através do site da Oracle (http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html?ssSourceSiteId=otnpt)]. Antes de clicar no botão "Add" da imagem acima, faça o download e instale a JDK. Depois, adicione através do botão "Add", apontando para a pasta raiz onde o JDK foi instalado.

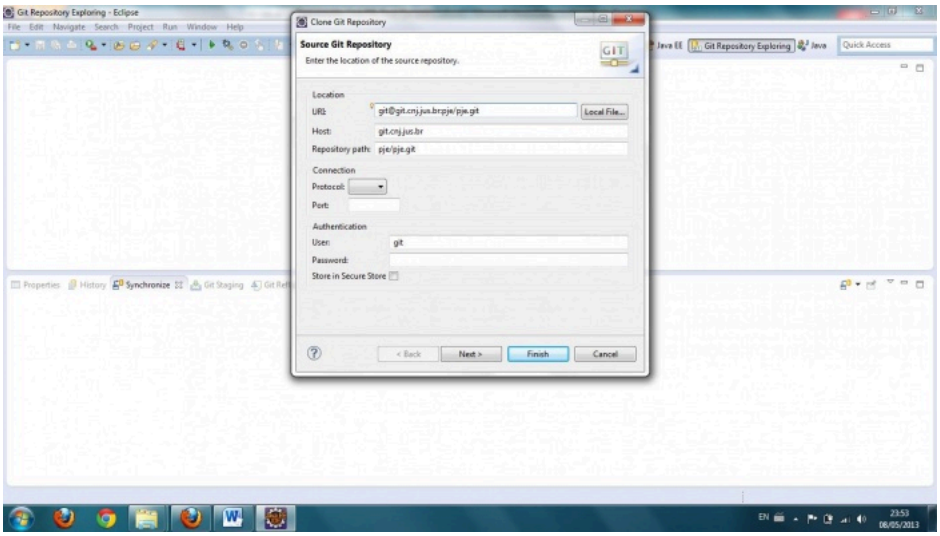


Ir para início do Roteiro para Instalação do Eclipse e de seus plugins

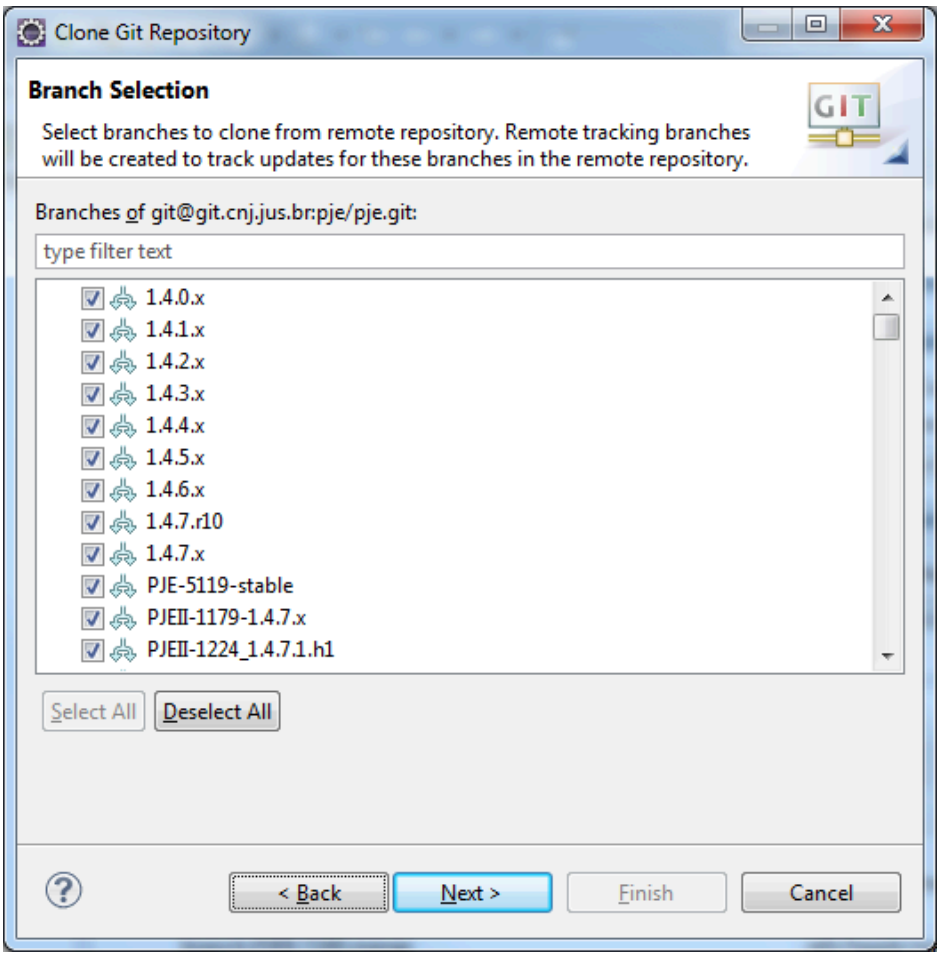
Obtendo a cópia do repositório do PJe

```
Atenção! Para obter uma cópia do PJe é necessário que um par de chaves SSH tenha sido gerado e configurado no servidor Git
```

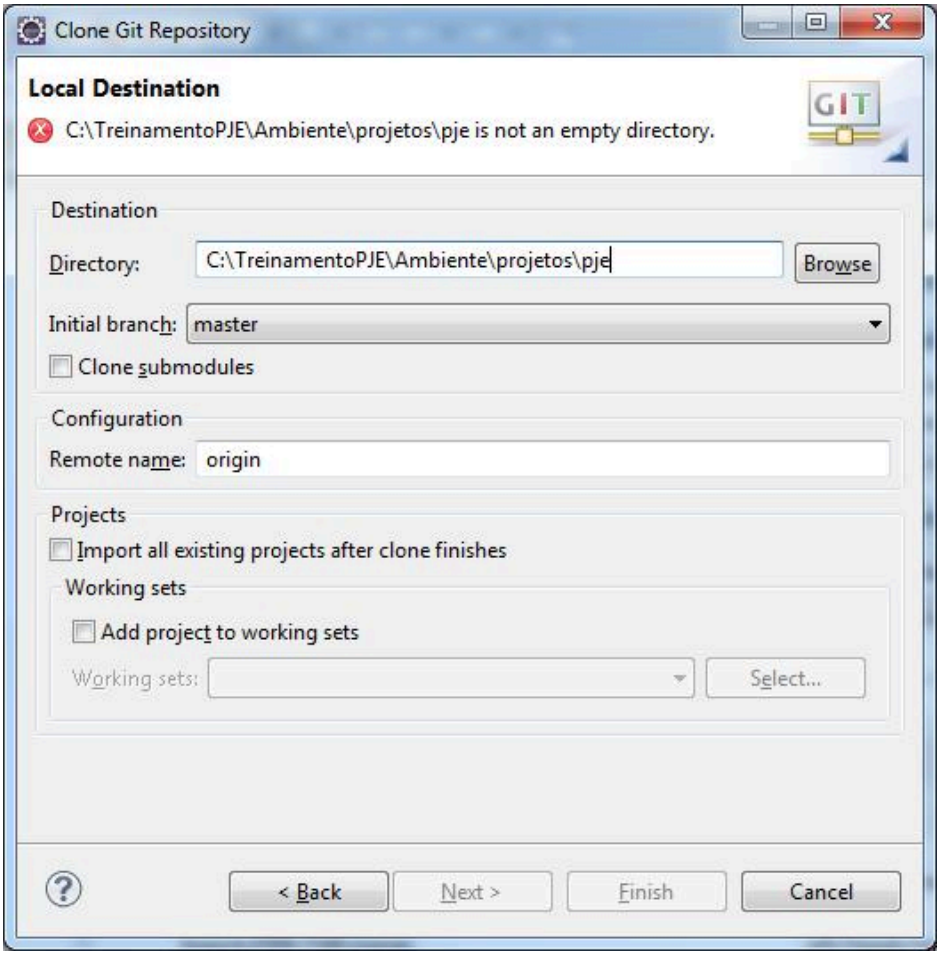
- Clonar o repositório por meio da funcionalidade “Clone Git Repository”, na aba Git Repositories na perspectiva Git do Eclipse. Na primeira clonagem é possível, dependendo da instalação do Eclipse, que sejam apresentadas duas janelas de diálogo seguidas solicitando confirmações. Confirme ambas



- Deixar todos os branchs selecionados e clicar em “Next”



- Escolher a pasta onde o repositório será clonado e clicar em “Next” (por exemplo: C:\TreinamentoPJE\Ambiente\projetos\pje)

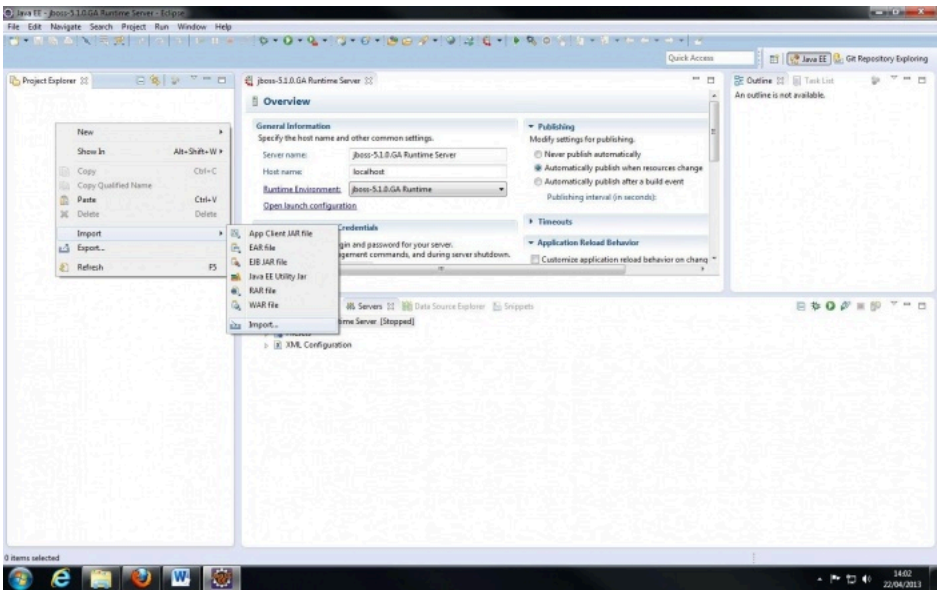


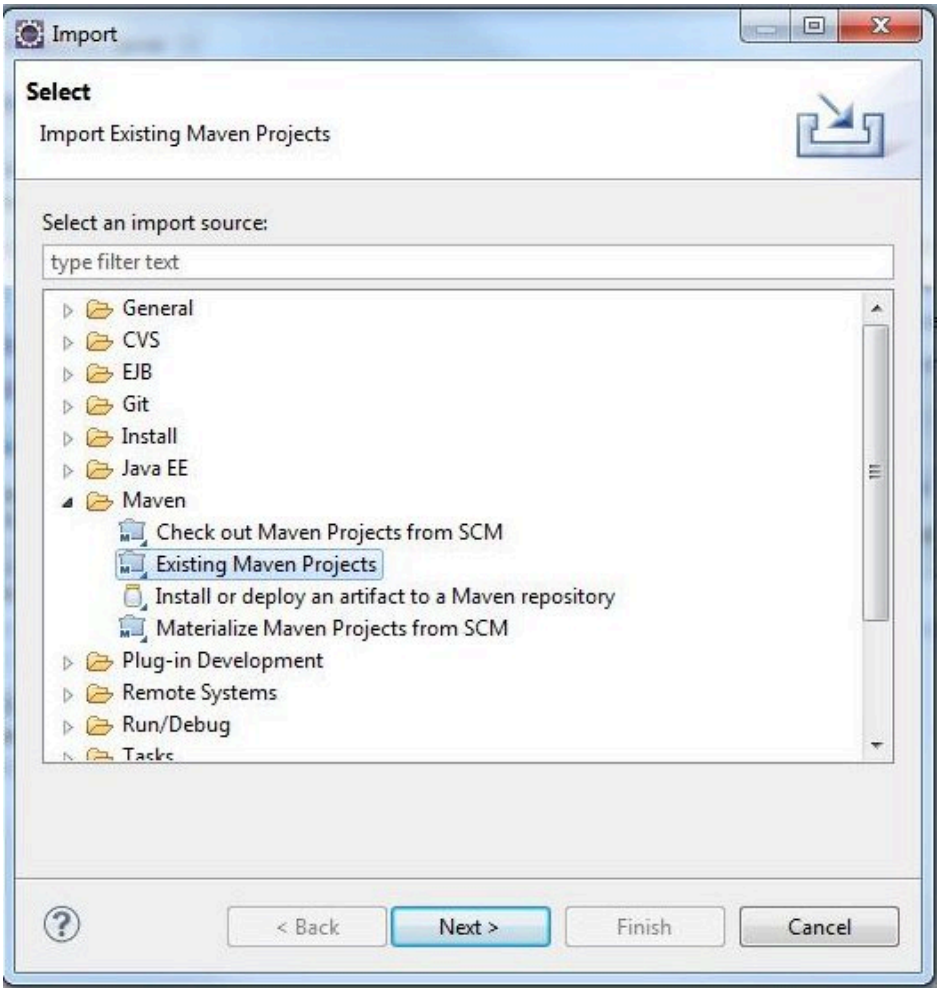
Ir para início do Roteiro para Instalação do Eclipse e de seus plugins

Criando o projeto a partir do repositório clonado

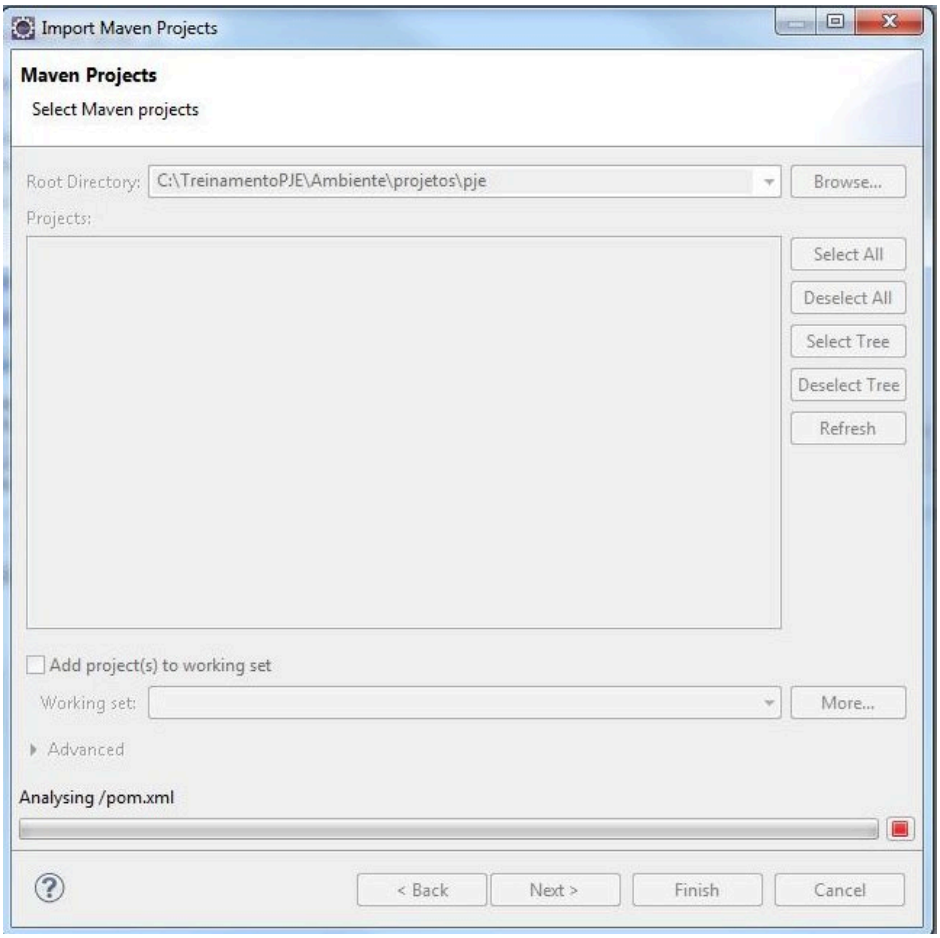
```
Atenção! Antes de criar o projeto, deve ter sido realizada a clonagem do repositório do PJe
```

- Ir para a perspectiva Java EE e clicar com o botão direito no Project Explorer

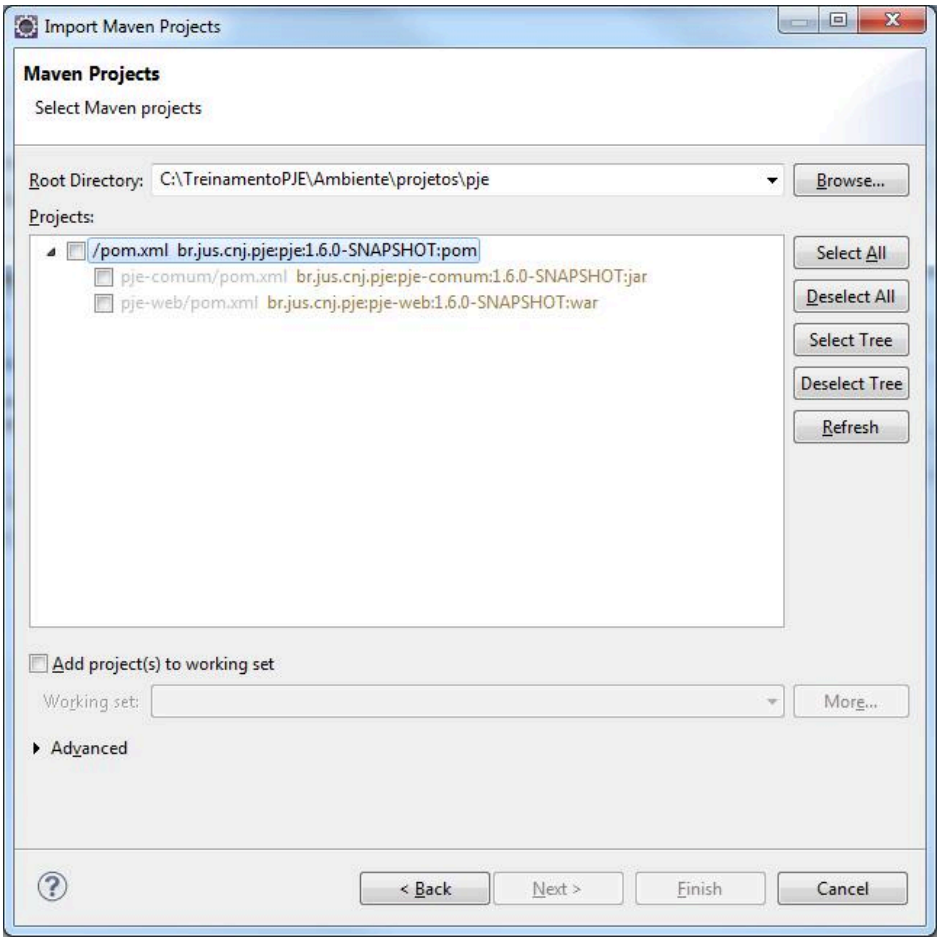


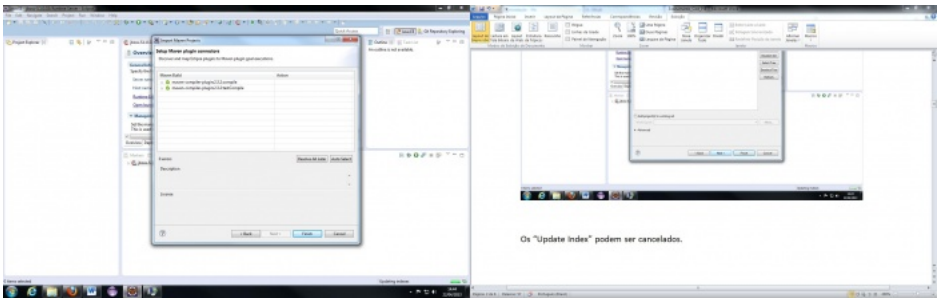


- Selecionar o caminho onde foi clonado o repositório do GIT, no passo anterior (por exemplo: C:\TreinamentoPJE\Ambiente\projetos\pje)



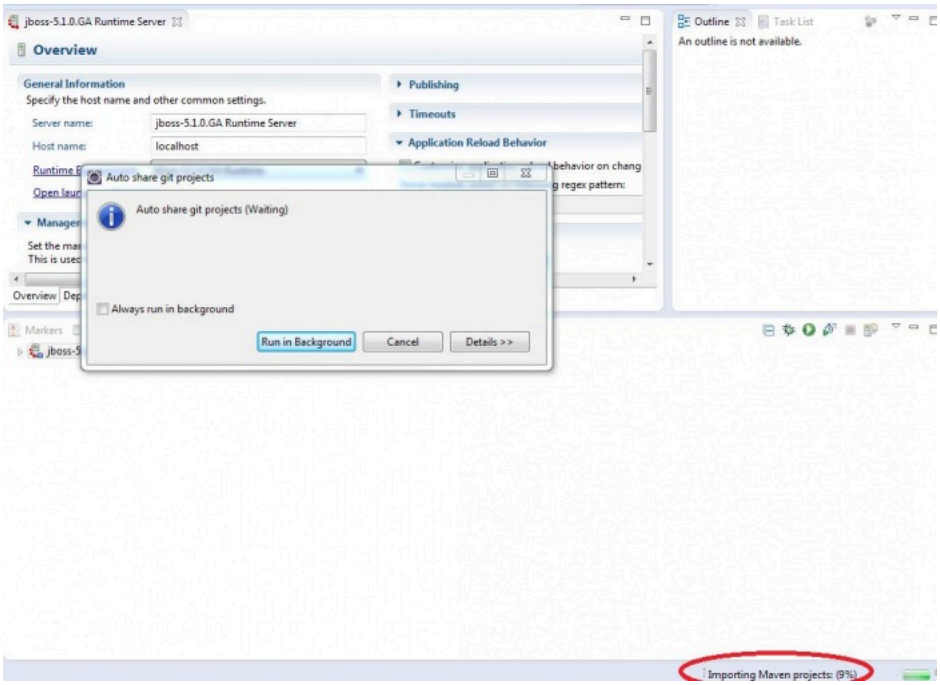
- Selecionar todos os pom's listados e clicar em "Next"





- Clicar em "Finish"

```
Atenção! A partir desse momento o download das dependências gerenciadas pelo Maven será iniciado.
Na primeira criação de projeto, em função da grande quantidade de dependências, o processo levará aproximadamente 30 minutos
```



Ir para início do Roteiro para Instalação do Eclipse e de seus plugins

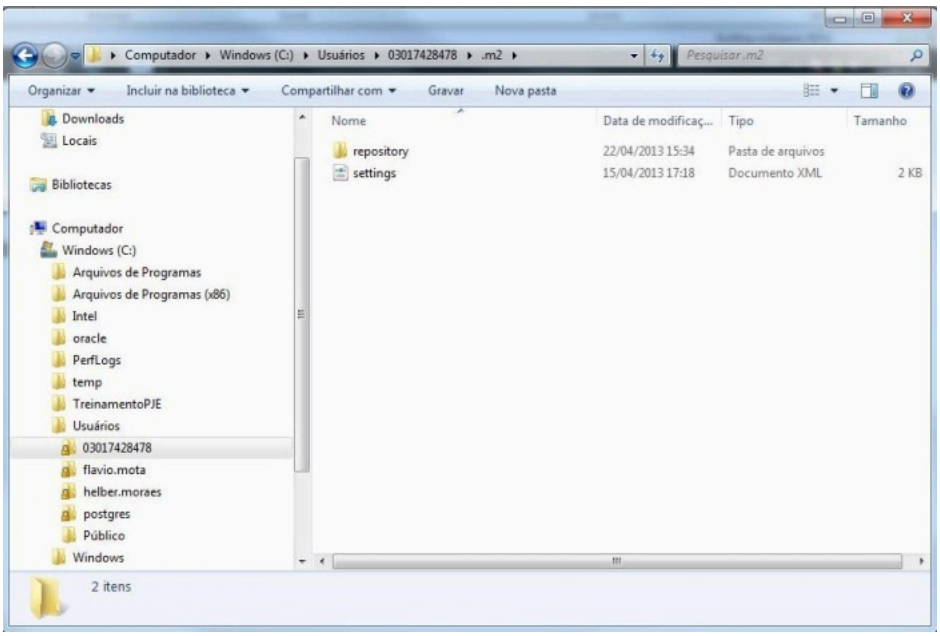
Dependências do Maven

```
Atenção! O arquivo settings.xml possui as configurações de acesso ao artfactory do CNJ.
```

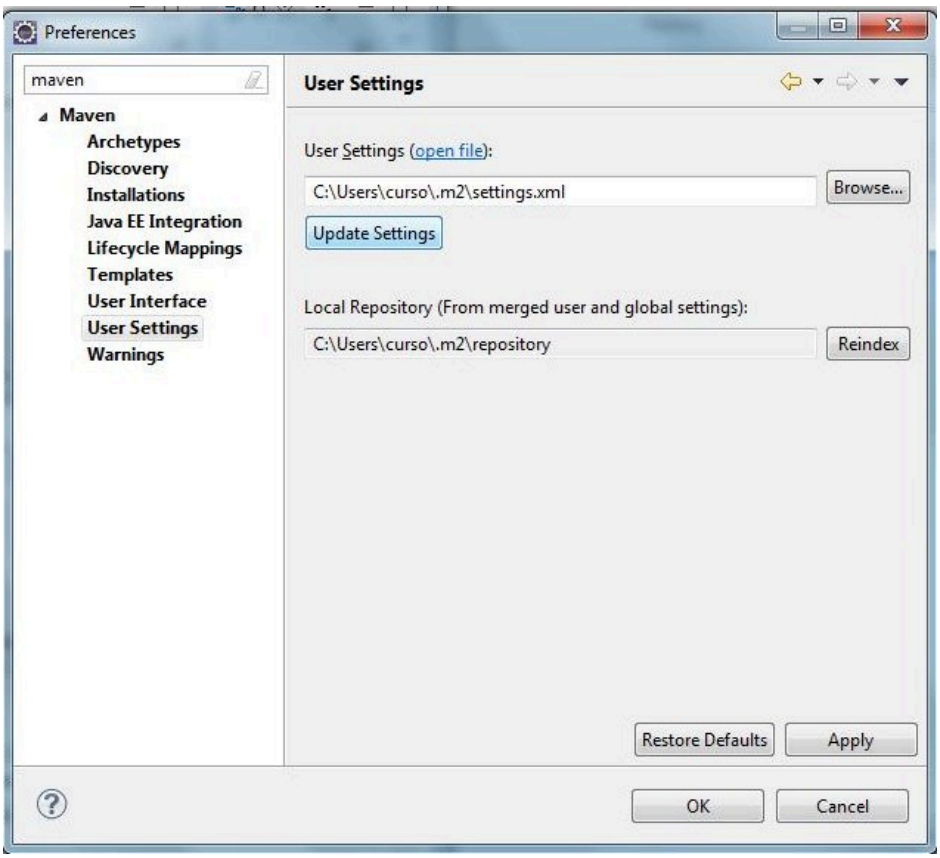
- Adicionar o arquivo settings.xml ao diretório .m2, cujo conteúdo segue exemplificado no trecho de código a seguir (insira seu login e senha do artifactory)

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd"
xmlns="http://maven.apache.org/SETTINGS/1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <servers>
   <server>
     <username>seu login</username>
     <password>sua senha</password>
     <id>central</id>
   </server>
   <server>
     <username>seu login</username>
     <password>sua senha</password>
     <id>snapshots</id>
   </server>
   <server>
      <id>pje-descanso</id>
      <username>postgres</username>
      <password>P123456.</password>
   </server>
 </servers>
 <profiles>
   <profile>
     <repositories>
       <repository>
         <snapshots>
           <enabled>false</enabled>
         </snapshots>
         <id>central</id>
         <name>repo</name>
         <url>http://www.cnj.jus.br/artifactory/repo</url>
       </repository>
       <repository>
         <snapshots />
         <id>snapshots</id>
         <name>repo</name>
         <url>http://www.cnj.jus.br/artifactory/repo</url>
       </repository>
     </repositories>
     <pluginRepositories>
       <pluginRepository>
         <snapshots>
           <enabled>false</enabled>
         </snapshots>
         <id>central</id>
         <name>repo</name>
         <url>http://www.cnj.jus.br/artifactory/repo</url>
       </pluginRepository>
       <pluginRepository>
         <snapshots />
         <id>snapshots</id>
         <name>repo</name>
         <url>http://www.cnj.jus.br/artifactory/repo</url>
       </pluginRepository>
     </pluginRepositories>
     <id>artifactory</id>
   </profile>
   <profile>
      <id>pje-descanso</id>
```

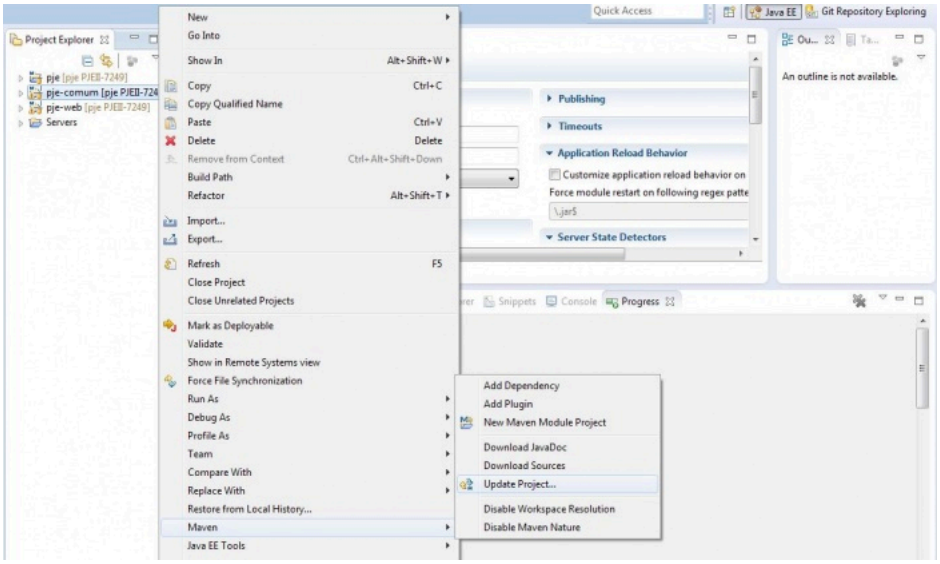
```
      <properties>
          <flyway.serverId>pje-descanso</flyway.serverId>
          <flyway.driver>org.postgresql.Driver</flyway.driver>
          <flyway.url>jdbc:postgresql://localhost:5432/pje_homologacao_1g</flyway.url>
      </properties>
    </profile>
  </profiles>
  <activeProfiles>
    <activeProfile>artifactory</activeProfile>
  </activeProfiles>
</settings>
```



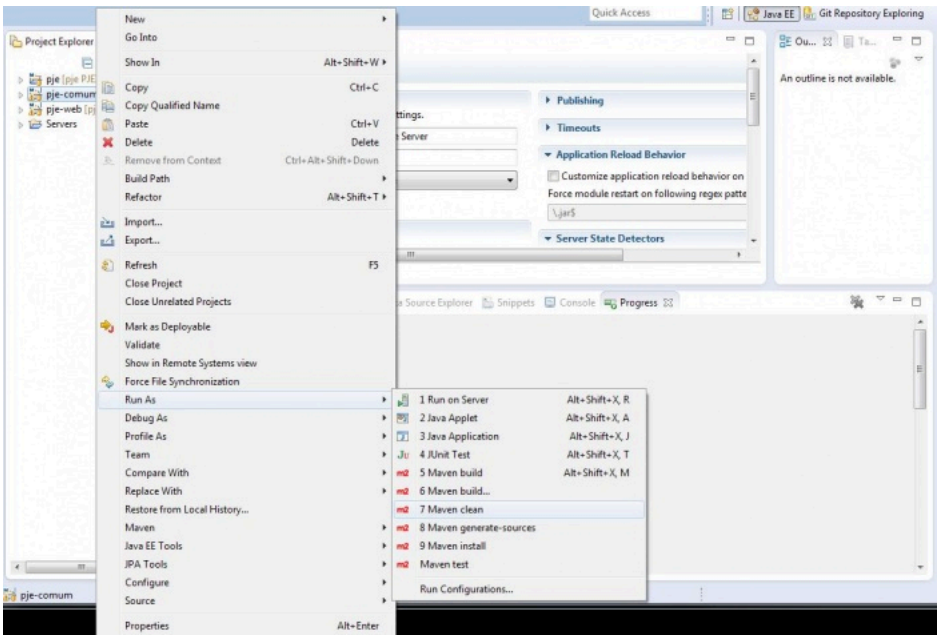
- Clicar em UPDATE SETTINGS do "Window > Preferences > Maven" (verifique se o caminho do arquivo settings.xml está adequado a sua instalação):



- Atualizar as dependências do Maven, agora com acesso ao artifactory do CNJ (sempre que necessário, atualizar as dependências do Maven)



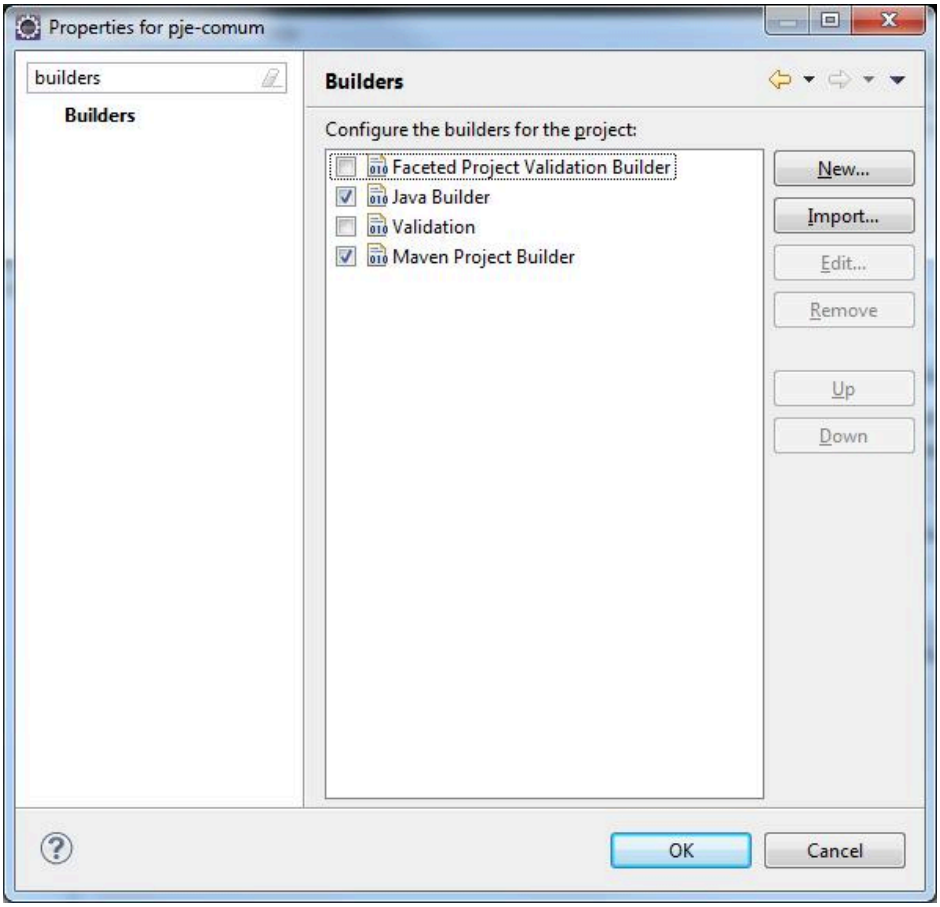
- Se ainda forem detectados problemas de dependência, executar procedimento de limpeza:

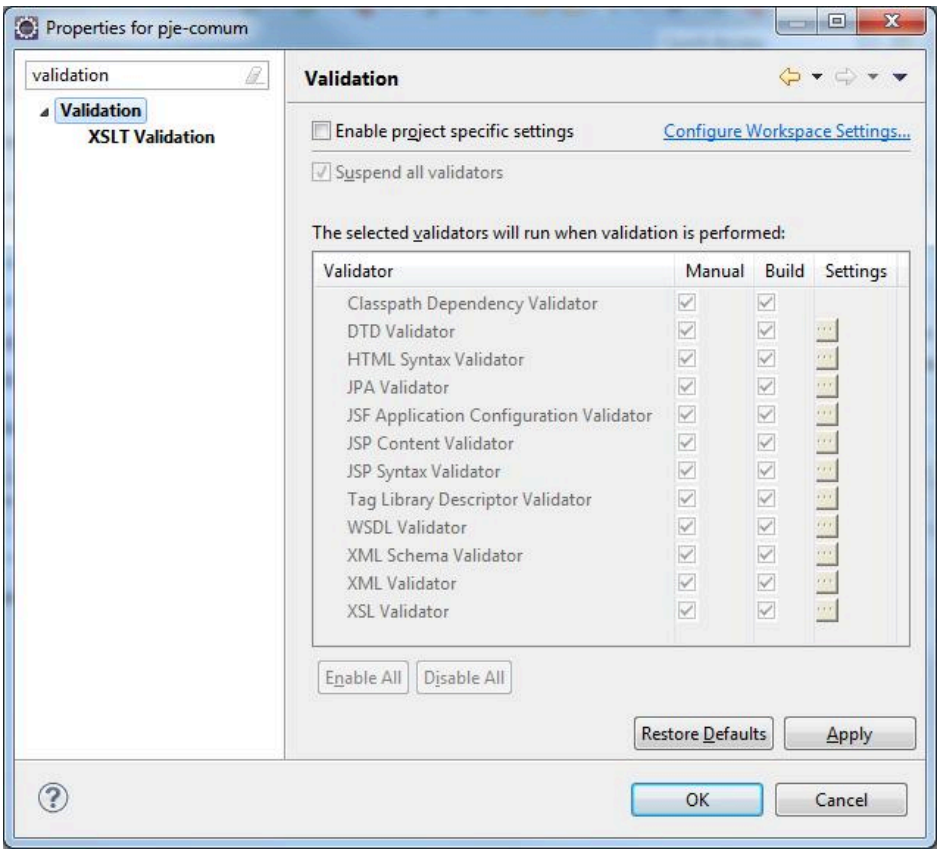


Ir para início do Roteiro para Instalação do Eclipse e de seus plugins

Removendo as validações desnecessárias

- Clicar com o botão direito em cada um dos projetos e configurar as seguintes propriedades:

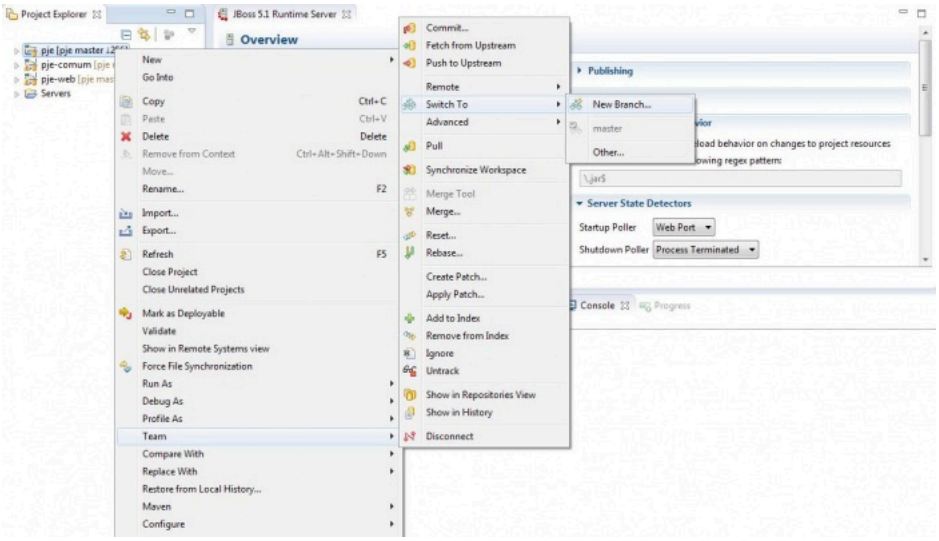




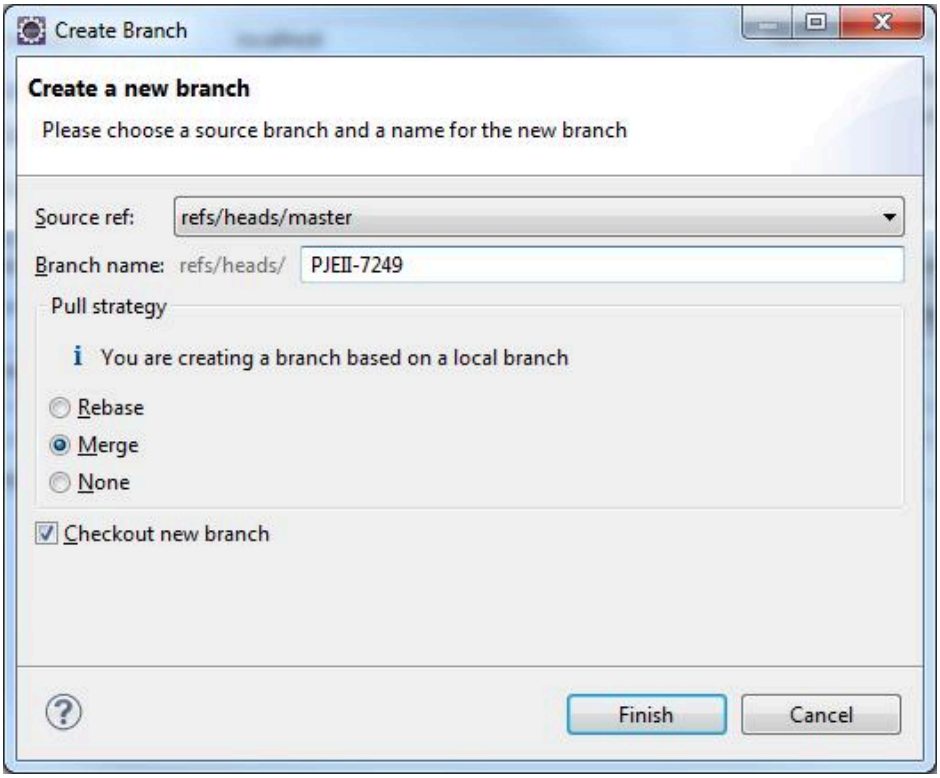
Ir para início do Roteiro para Instalação do Eclipse e de seus plugins

Criando um branch para iniciar a resolução de uma pendência (issue)

- Verificar instruções do EGit;
- Solicitar alteração do branch de trabalho para um novo branch, conforme abaixo:



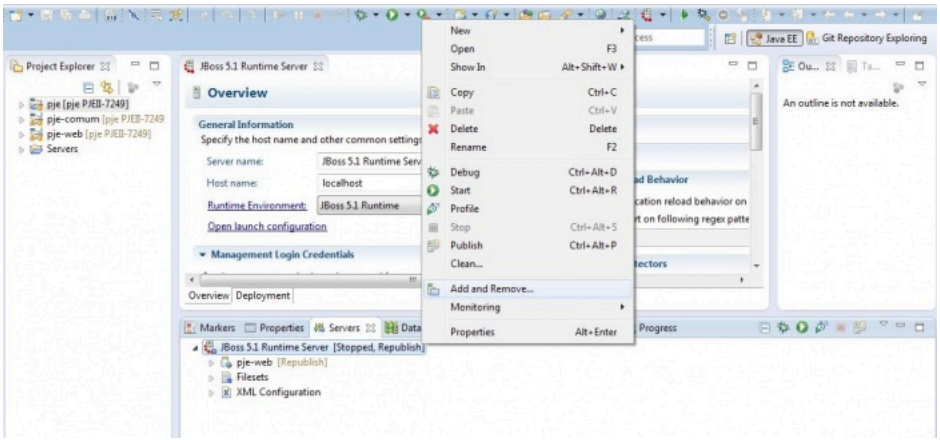
- Criar o branch novo, referenciando o nome da pendência a ser resolvida. Adicionalmente, para contemplar necessidade de diversos pedidos de integração devido às várias versões em uso do PJe, sugere-se acrescentar ao nome do branch novo o nome do branch base que se está usando para criar o branch. Por exemplo, PJEII-7249-master, ou PJEII-7249-stable, ou ainda PJEII-7249-1.4.6.x

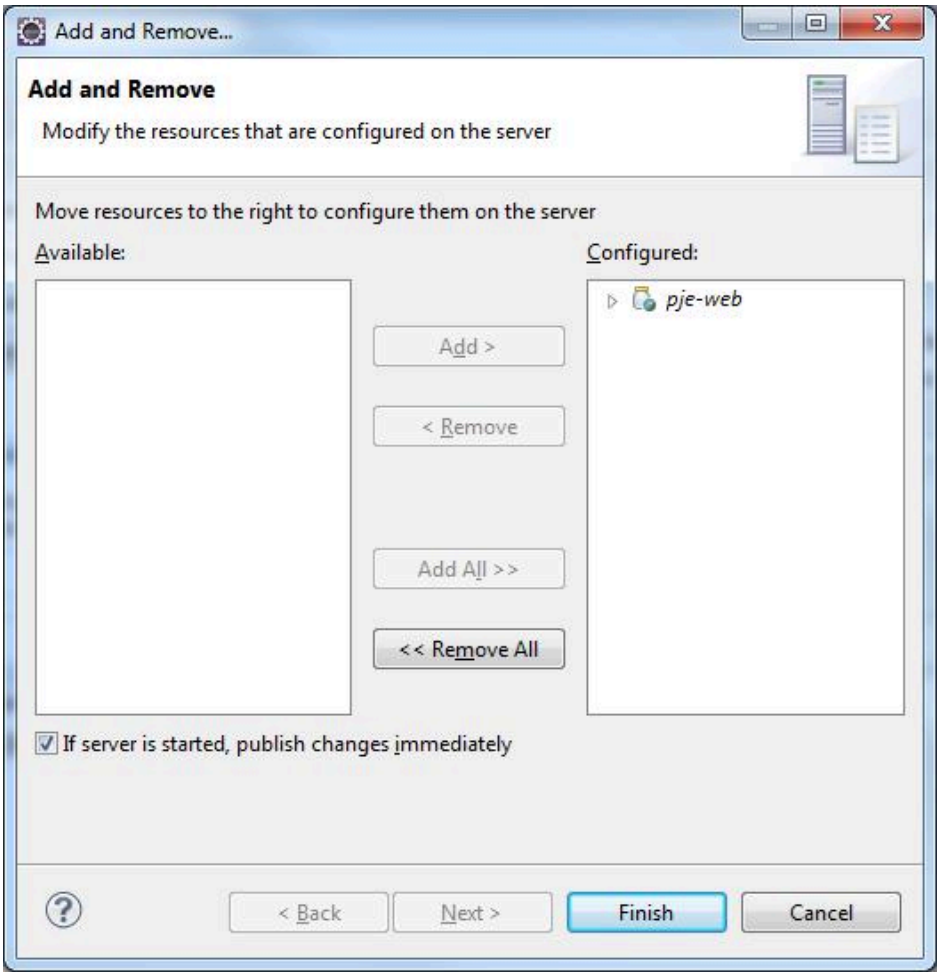


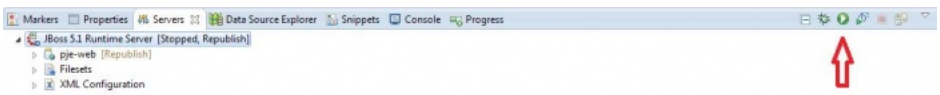
Após finalizado o procedimento, terá sido criado o branch em seu repositório local.

Publicando a aplicação com Jboss

- Na aba Servers, adicione um novo servidor;
- Aponte o ambiente de execução (runtime environment) para o diretório da instalação do JBoss;
- Copie e cole o arquivo de datasources “PJE-ds.xml” para o diretório “/server/default/deploy” contido na instalação do JBoss (Lembre-se de editar o arquivo XML conforme a necessidade).

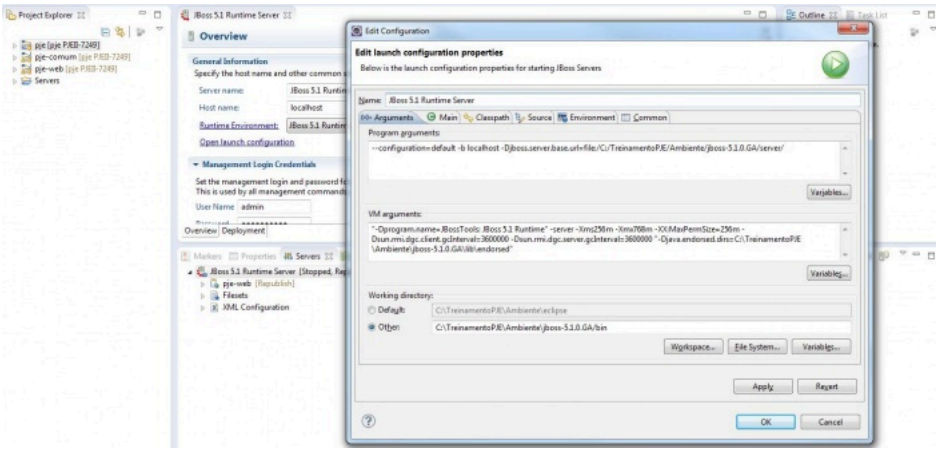






Alterando parâmetros de performance do Jboss

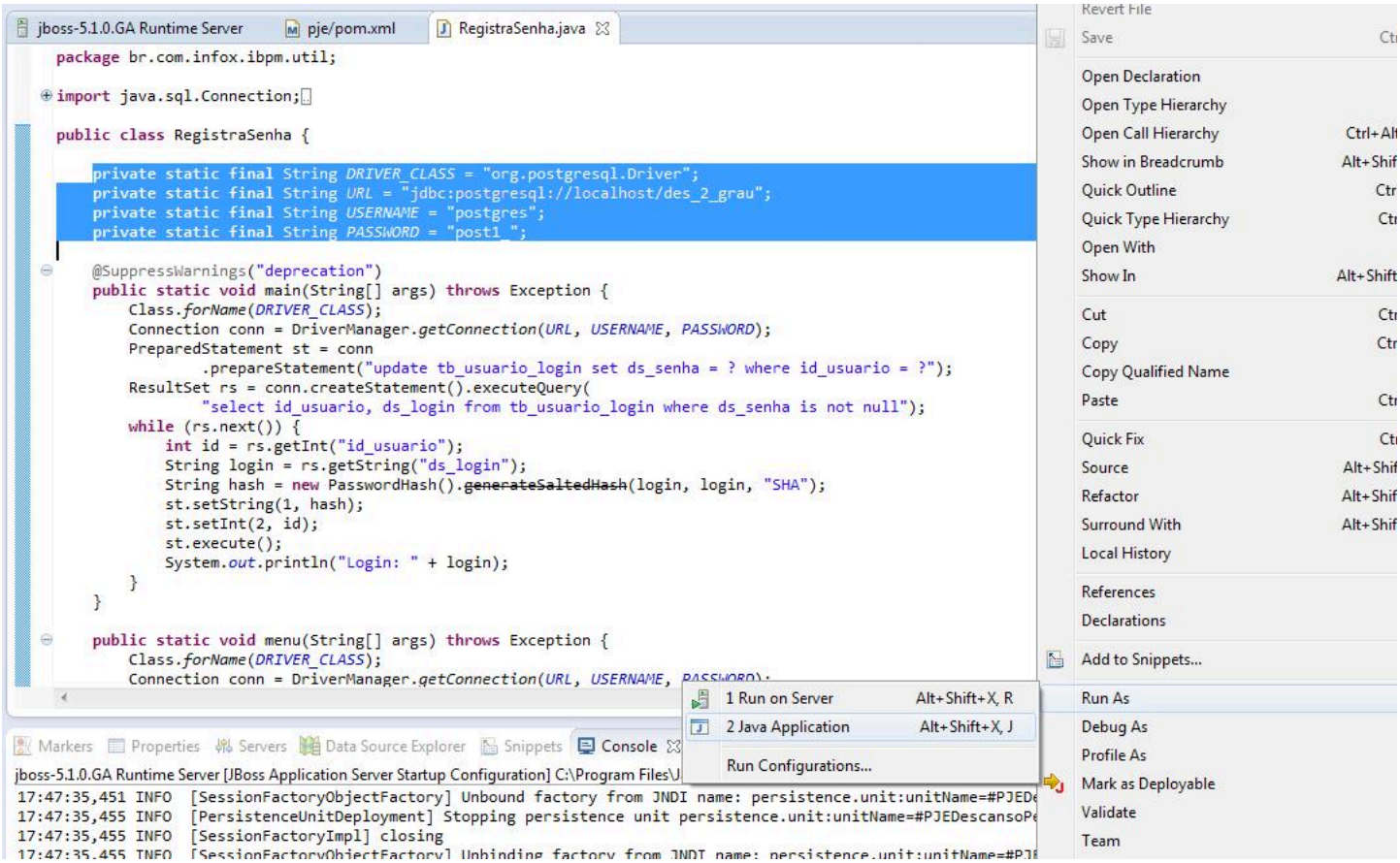
Pode ser que o Eclipse fique com sua execução prejudicada devido a problemas de performance do Jboss. Se for esse o caso, para alterar os parâmetros de performance, realize um clique duplo no servidor Jboss e depois clique em "Open launch configuration":



- Verifique os VM Arguments. Tente melhorar a performance do JBoss aumentando o Xms, Xmx e MaxPermSize. (atentar para orientações disponíveis no manual de instalação - Xms = 1024m, Xmx = 1024 e MaxPermSize=256)

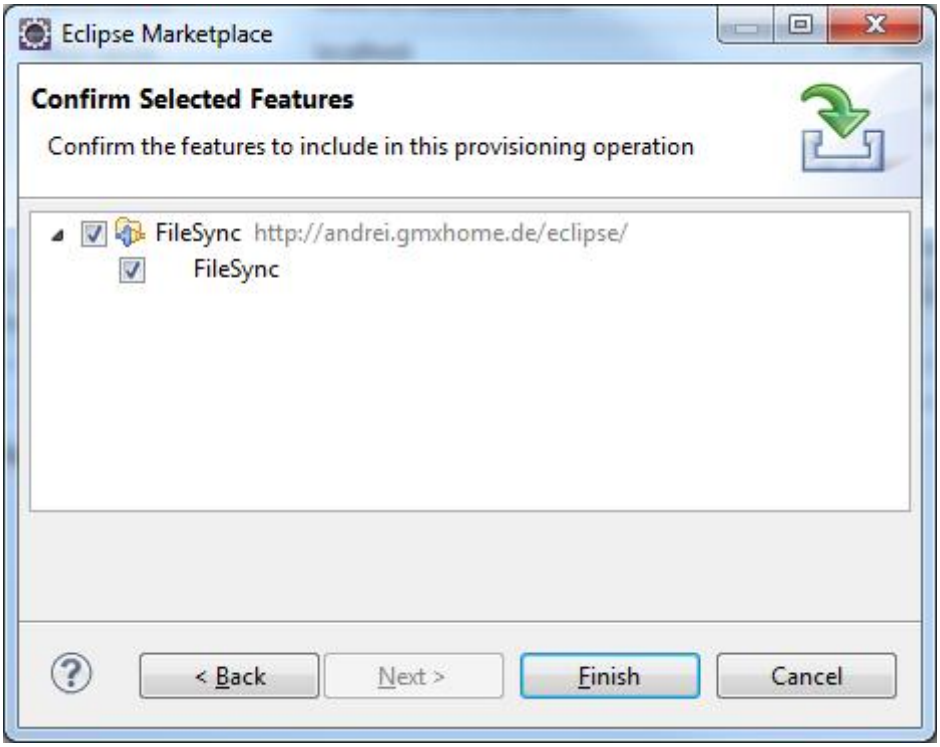
Usuários para acessar sua instância do PJE

Para “zerar” as senhas dos usuários no banco de dados, executar o método main da classe RegistraSenha (após configurar acesso ao banco):

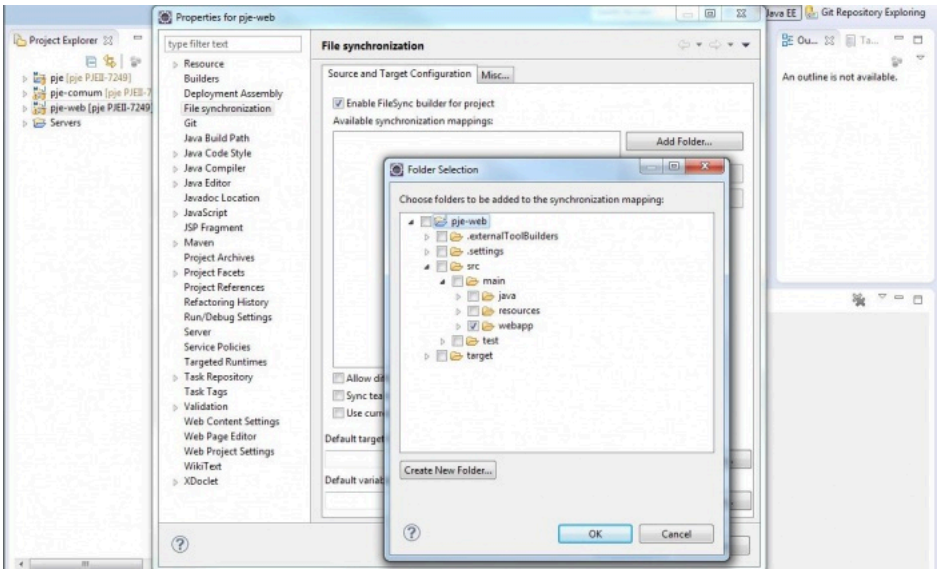


Instalando e configurando o FileSync

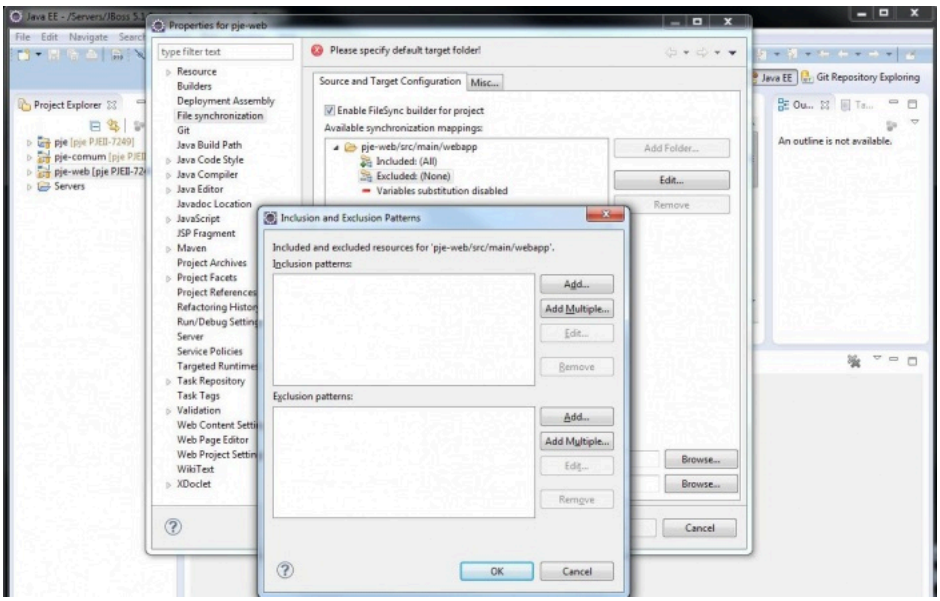
O FileSync é um plugin que executa no Eclipse, permitindo a 13.1. sincronização de arquivos mantidos por ele (leia http://marketplace.eclipse.org/content/filesync). Instale o plugin a partir do Eclipse em “Help > Eclipse Marketplace”:



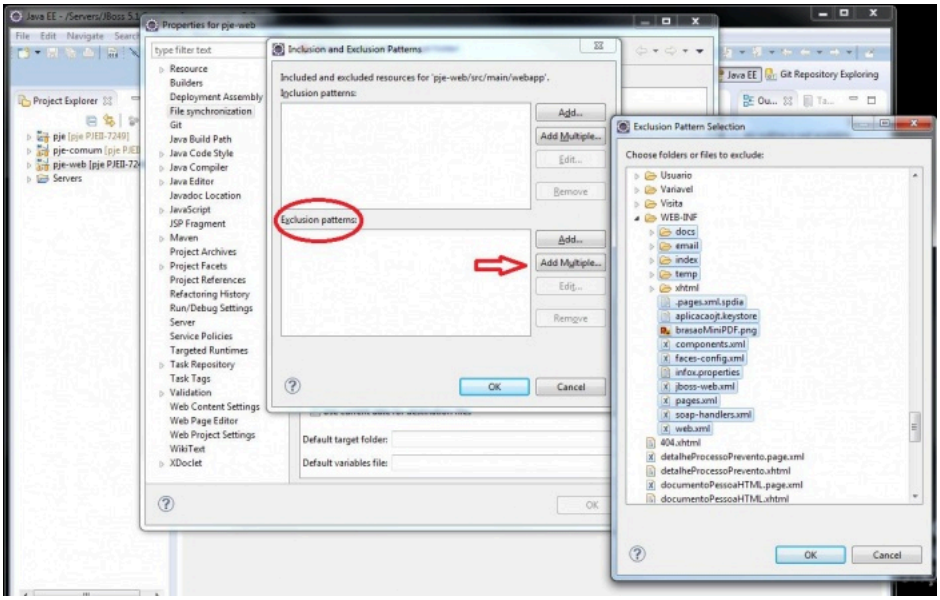
- Após a instalação, clicar com o botão direito no projeto que contém o ponto de início de execução (pje-web), selecionar "Properties" e localizar o item “File synchronization”. Clicar em “Add Folder” e adicionar a pasta /src/main/webapp:



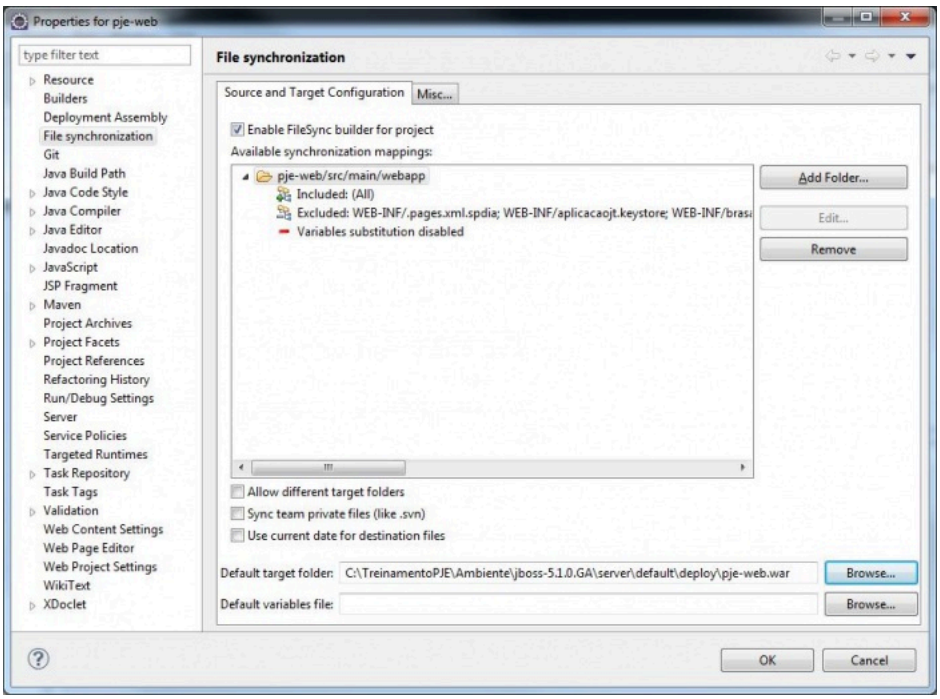
- Abrir a pasta incluída, selecionar “Excluded” e clicar em “Edit”:



- Clicar no botão “Add Multiple” do “Exclusion patterns”, selecionar todos os arquivos do WEB-INF exceto a pasta XHTML e clicar em “OK”:



- Informar em "Default target folder" o caminho do .war:

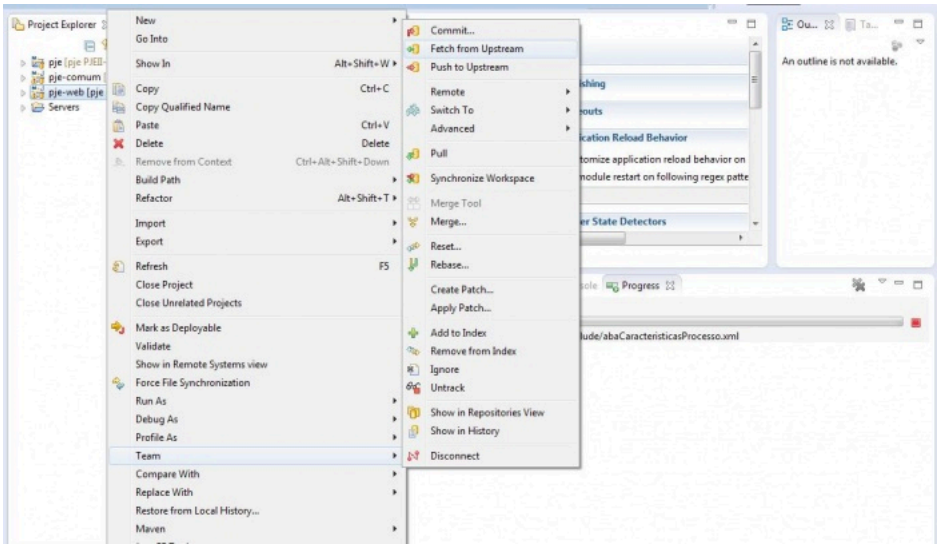


Comandos GIT

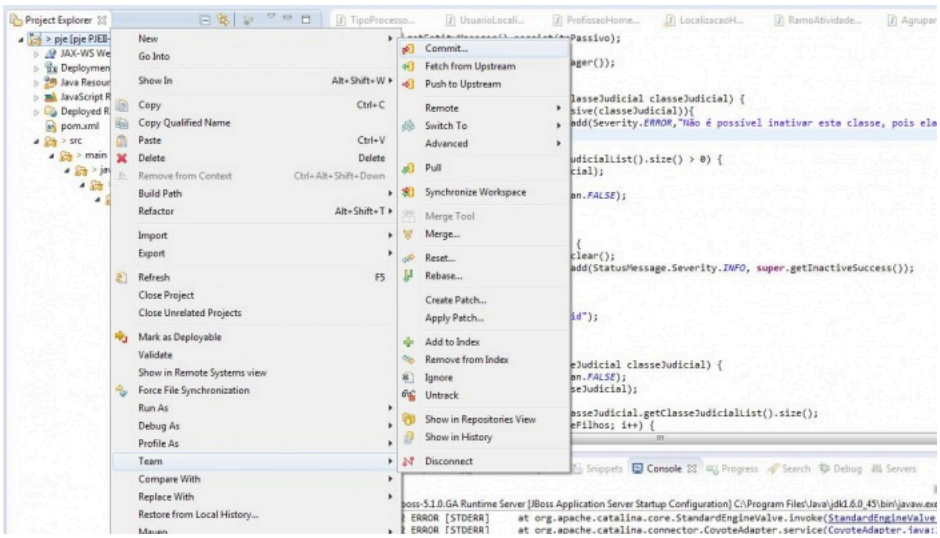
- Para começar, leia a documentação existente em http://www.atlassian.com/git. Estude primeiro o tutorial para iniciantes (http://www.atlassian.com/git/tutorial).

Exemplos de comandos:

- FETCH (atualizar um branch local com o código do servidor remoto):


Project Explorer
New
Go Into
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Refactor Alt+Shift+T
Import
Export
Refresh F5
Close Project
Close Unrelated Projects
Mark as Deployable
Validate
Show in Remote Systems view
Force File Synchronization
Run As
Debug As
Profile As
Team
Compare With
Replace With
Restore from Local History...
Maven
Commit...
Fetch from Upstream
Push to Upstream
Remote
Switch To
Advanced
Pull
Synchronize Workspace
Merge Tool
Merge...
Reset...
Rebase...
Create Patch...
Apply Patch...
Add to Index
Remove from Index
Ignore
Untrack
Show in Repositories View
Show in History
Disconnect
An outline is not available.
Progress

- COMMIT


Project Explorer
New
Go Into
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Refactor Alt+Shift+T
Import
Export
Refresh F5
Close Project
Close Unrelated Projects
Mark as Deployable
Validate
Show in Remote Systems view
Force File Synchronization
Run As
Debug As
Profile As
Team
Compare With
Replace With
Restore from Local History...
Commit...
Fetch from Upstream
Push to Upstream
Remote
Switch To
Advanced
Pull
Synchronize Workspace
Merge Tool
Merge...
Reset...
Rebase...
Create Patch...
Apply Patch...
Add to Index
Remove from Index
Ignore
Untrack
Show in Repositories View
Show in History
Disconnect
Console
Progress

Commit Changes

Commit Changes to Git Repository

Commit message

```
[PJEII-5847] Algumas mensagens não condizem com a operação realizada.
Foi adicionada validação para identificar a operação realizada (no caso,
inativação) e então emitir a mensagem adequada.
```

Author: kgsilva <kgsilva@trf5.jus.br>

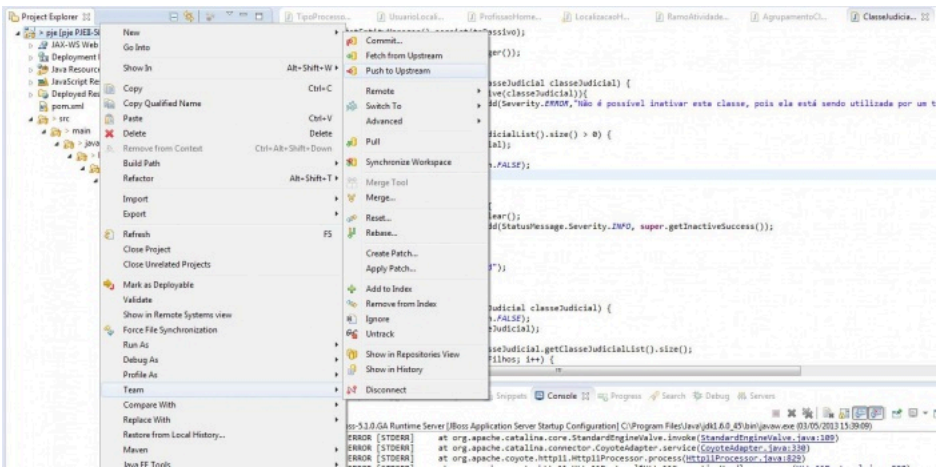
Committer: kgsilva <kgsilva@trf5.jus.br>

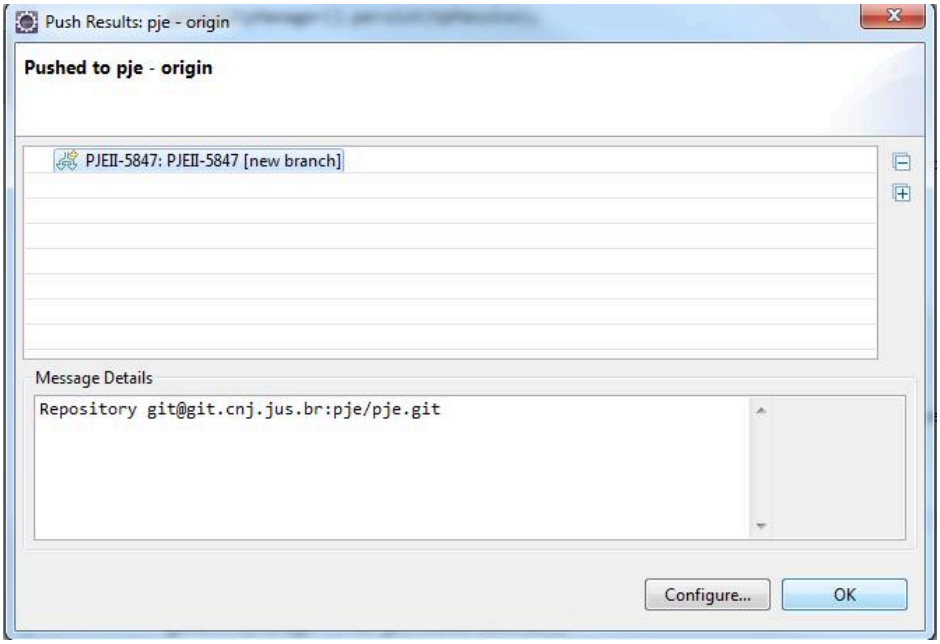
Files (7/13)

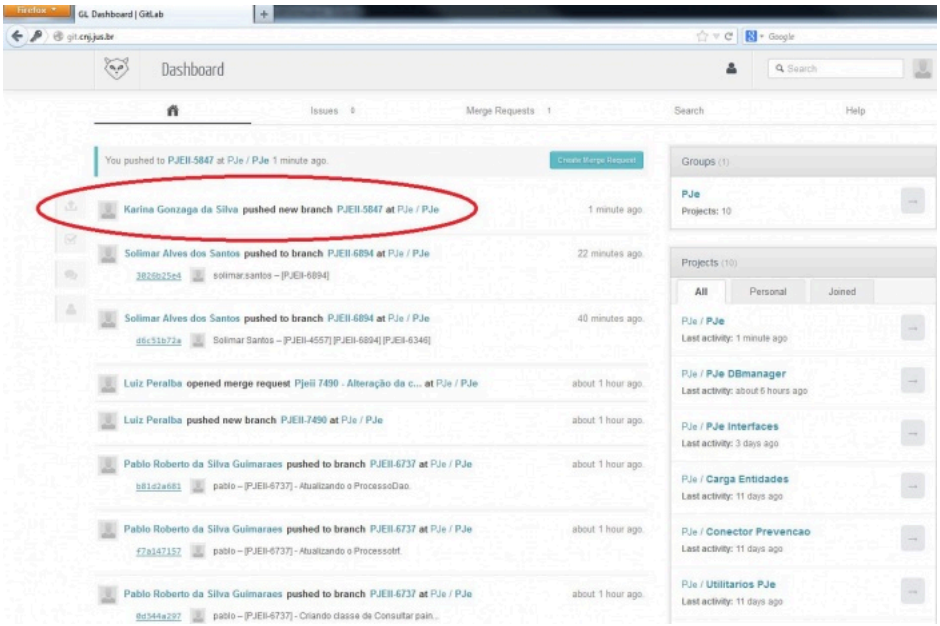
type filter text

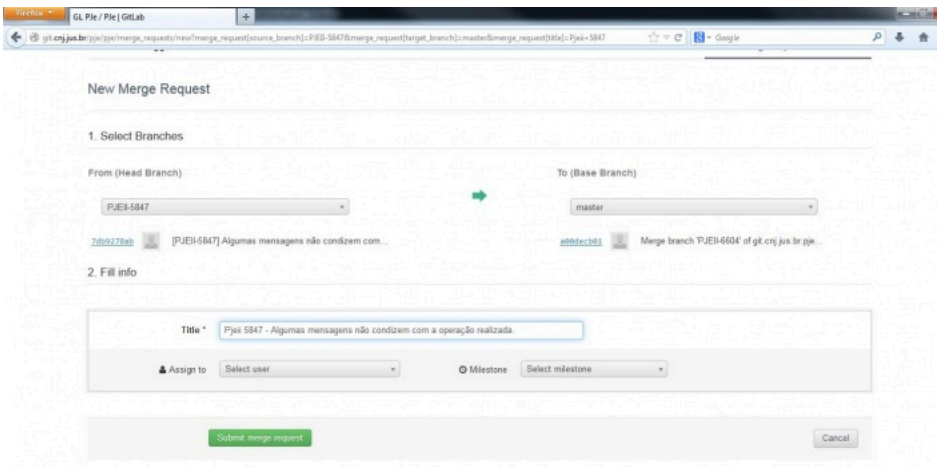
Status	Path
☐	src/main/java/br/com/infox/cliente/actions/anexarDocumentos/AnexarDocumentos.java
☑	src/main/java/br/com/infox/cliente/home/AgrupamentoClasseJudicialHome.java
☑	src/main/java/br/com/infox/cliente/home/ClasseJudicialHome.java
☑	src/main/java/br/com/infox/cliente/home/ProfissaoHome.java
☑	src/main/java/br/com/infox/cliente/home/RamoAtividadeHome.java
☑	src/main/java/br/com/infox/cliente/home/TipoProcessoDocumentoTrfHome.java
☑	src/main/java/br/com/infox/cliente/home/UsuarioLocalizacaoMagistradoServidorHome.java
☑	src/main/java/br/com/infox/ibpm/home/LocalizacaoHome.java
☐	src/main/resources/entity_messages.properties
☐	src/main/webapp/Processo/AnexarDocumentos/abaAnexarDocumentosPdf.xml
☐	src/main/webapp/TipoProcessoDocumento/listView.xhtml
☐	src/main/webapp/WEB-INF/xhtml/tags/form/fields/fileUploadBox.xhtml
☐	src/main/webapp/upload/uploadPopup.xhtml

Commit and Push | Commit | Cancel



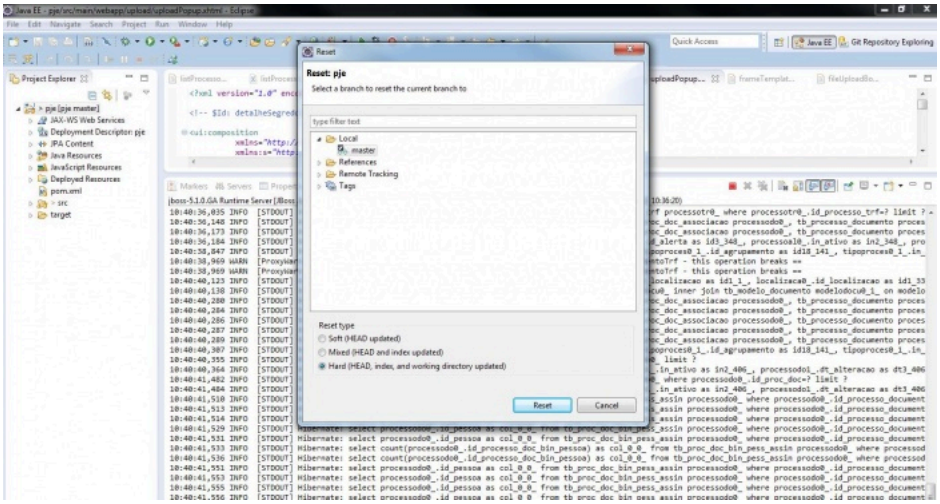






- RESET (apagar as modificações feitas em um branch local):

```
Lembre-se:
Quando você desfaz suas modificações com reset git, não existe maneira alguma de recuperá-las.
Leia: http://www.atlassian.com/git/tutorial/undoing-changes#!reset .
```



Depois, deve ser realizado um PULL a fim de recuperar a versão mais atualizada do branch.

Observações

O vídeo que descreve a configuração do projeto no Eclipse está um pouco desatualizado, mas é importante dar uma olhada:

- Teclas de atalho do Eclipse:
  - CTRL + SHIT + F = Identar
  - CTRL + SHIT + R = Localizar arquivo ou classe
  - CTRL + SHIT + T = Localizar tipo (classe)
  - CTRL + SHIT + O = Imports
  - CTRL + D = Apagar linha
  - CTRL + F = Localizar
  - CTRL + O = Localizar métodos e atributos dentro de um arquivo

---

A seguir, alguns vídeos que auxiliam na configuração do ambiente.

Download das ferramentas

Para iniciar o desenvolvimento do sistema PJE, faça o download das ferramentas necessárias através do endereço:
https://wwwh.cnj.jus.br/svn/projeto_pje/trunk/ambiente (atualizar - svn não mais necessário)

Veja como fazer o download: Vídeo

Instalando o Postgres

Para instalar o Postgres, execute o instalador que foi baixado anteriormente deixando as opções sugeridas. Na etapa de definição de senha de administrador, informe "P123456."

Criando base de dados

Crie duas bases de dados: PJE_1_4 e PJE_1_4_BIN, sendo a primeira necessária para armazemanento dos dados do PJE e a segunda para armazenamento dos documentos dos processos gerados pelo PJE.
Veja como fazer a criação da base de dados: Vídeo

Restaurando base de dados

Após criar as bases, vamos copiar os dados e a estrutura do banco de desenvolvimento e restaurá-lo em nosso banco local.
Veja como: Vídeo

Configurando o Eclipse

Tutorial que ensina as principais configurações para desenvolvimento no eclipse, tais como: configuração do MAVEN, download de códigos-fonte no SVN, configuração do JBoss, configuração de HotDeploy (atualizar - svn não mais necessário) Vídeo

Gerando certificados CACERTS

O PJE faz uso de alguns webServices para consulta de dados, como receita federal e OAB. Para que ele consiga executar as pesquisas, é necessário gerar alguns certificados. Veja no vídeo como fazê-lo: Vídeo

Desenvolvimento

Criando entidades

Neste vídeo, demonstraremos como criar entidades para armazenamento de dados no padrão exigido pelo PJE.
Script de criação das tabelas usadas no exemplo: Arquivo:Script criacao tabelas.txt
Vídeo

Criando DAO

Neste vídeo, demonstraremos como criar classes do tipo DAO para acesso aos dados de uma entidade. Uma DAO acessa dados de somente uma entidade.
Vídeo

Criando manager

Neste vídeo, demonstraremos como criar classes do tipo "manager" para adicionarmos regras de negócio e validações antes de persistir os dados de uma entidade. Managers controlam regras de negócio de apenas uma entidade.
Vídeo

Criando action

Neste vídeo, demonstraremos como criar classes do tipo "action" que farão o controle de uma determinada tela.
Vídeo

Criando páginas

Neste vídeo, demonstraremos como criar páginas "xhtml" de pesquisa e edição de registros.
Vídeo

Gerando Deploy

Vídeos que orientam o desenvolvedor sobre como gerar corretamente o deploy do PJE:

- Deploy através do MAVEN: Vídeo
- Deploy manualmente: Vídeo

Interfaces para extensão do sistema

O sistema PJe evidentemente não é capaz de contemplar toda a variedade de implementações existentes dadas pelos tribunais para os diversos serviços auxiliares ou acessórios que orbitam a atividade jurisdicional. Alguns desses serviços são intensamente próximos da atividade judicial, outros são um pouco mais distantes. O traço mais comum é, no entanto, a grande possibilidade de variação de implementação dada por cada um dos tribunais.

Em razão dessa apreensão, é imprescindível permitir que algumas atividades sejam implementadas pelo tribunal interessado e conectadas ao PJe. Essa conexão pode ser feita de várias maneiras, desde serviços web até implementações mais internas. Uma alternativa de implementação mais rápida é a construção de componentes Seam que possam ser instalados no PJe para sua posterior utilização em nós de fluxos.

A presente página apresenta as interfaces inicialmente disponibilizadas para algumas das atividades mais críticas do sistema e disponibiliza um projeto de exemplo contendo uma implementação vazia para uma dessas interfaces.

Interfaces disponibilizadas

Na versão 1.0 do projeto de interfaces do PJe, serão disponibilizadas interfaces para as seguintes atividades:

- verificação de prevenção em relação a outros sistemas - br.jus.cnj.pje.extensao.VerificadorPrevencaoExterna
- envio e recebimento de informações para sistema de diário eletrônico - br.jus.cnj.pje.extensao.PublicadorDJE
- envio e recebimento de comunicações postais - br.jus.cnj.pje.extensao.ConectorECT
- verificação de recolhimento de custas processuais - br.jus.cnj.pje.extensao.VerificadorCustasProcessuais

Essas interfaces estão definidas no projeto pje-interfaces-1.0.0, empacotadas em um arquivo JAR que deve ser utilizado na implementação da interface. A documentação relativa a cada uma das interfaces pode ser consultada diretamente na WIKI e nos arquivos de documentação do desenvolvedor.

Desenvolvimento de ponto de extensão

O desenvolvimento do ponto de extensão do PJe passa pelas seguintes etapas:

- implementação da interface como componente Seam
- definição de arquivo components.xml em que o componente da interface seja inicializado pela classe que a implementa
- empacotamento do ponto de extensão, devendo ser identificadas as bibliotecas de dependência para instalação junto ao PJe
- instalação do ponto de extensão e concretização de seus testes

Implementação da interface

O JBoss Seam 2.2.X permite que sejam criados componentes em pacotes autônomos, que podem ser posteriormente instalados em sistemas Seam. Uma vez instalados, esses componentes são inicializados juntamente com os demais componentes da aplicação e ficarão disponíveis para injeção e utilização pela aplicação. Para tanto, basta implementar a interface e, no arquivo components.xml, definir o componente e a classe que o implementa. O exemplo que será apresentado versará sobre a implementação da interface de verificação de prevenção. Essa interface tem uma só função definida:

```
String[] verificaPrevencao(String processoOrigem, int codClasse, int codAssuntoPrincipal, int[] codAssuntos, String[] nomesAutores, String[]
documentosAutores, String[] nomesReus, String[] documentosReus)
```

A implementação pode adotar o mecanismo que o tribunal implementador entender mais adequado, desde arquivo de sistemas até serviços Web. Neste exemplo, vamos adotar um mecanismo de verificação em banco de dados.

```
<code class="java">
@Name("verificadorPrevencaoExterna")
@Scope(ScopeType.APPLICATION)
@Install(precedence=Install.APPLICATION)
@Startup
public class VerificadorPrevencaoExemplo implements VerificadorPrevencaoExterna {

	@In(create=true)
	private DataSource prevencaoDS;

	@Override
	public String[] verificaPrevencao(String processoOrigem, int codClasse, int codAssuntoPrincipal, int[] codAssuntos, String[] nomesAutores,
	   String[] documentosAutores, String[] nomesReus, String[] documentosReus){
		List<String> nomesPartes = new ArrayList<String>();
		nomesPartes.addAll(Arrays.asList(nomesAutores));
		nomesPartes.addAll(Arrays.asList(nomesReus));

		List<String> documentosPartes = new ArrayList<String>();
		documentosPartes.addAll(Arrays.asList(documentosAutores));
		documentosPartes.addAll(Arrays.asList(documentosReus));

		int numeroNomes = nomesPartes.size();
		int numeroDocumentos = documentosPartes.size();

		String qStr = buildQuery(numeroDocumentos, numeroNomes);

		Connection con = ds.getConnection();
		PreparedStatement pstmt = con.prepareStatement(qStr);
		pstmt.setInt(1, codClasse);
		pstmt.setInt(2, codAssuntoPrincipal);
		for(int i = 3; i < 3 + numeroNomes; i++){
			pstmt.setString(i, nomesPartes.get(i - 3));
		}
		for(int i = 3 + numeroNomes; i < 3 + numeroNomes + numeroDocumentos; i++){
			pstmt.setString(i, documentosPartes.get(i - numeroNomes - 3));
		}
		ResultSet rs = pstmt.executeQuery();
		List<String> ret = new ArrayList<String>();
		while(rs.next()){
			ret.add(new String(rs.getString(1)));
		}
		return ret.toArray(new String[]{});
	}

	private String buildQuery(int numeroNomes, int numeroDocumentos){
		StringBuilder sb = new StringBuilder("SELECT DISTINCT p.numero FROM tb_processo AS p " +
				"JOIN tb_partes AS par ON par.processo = p.numero " +
				"JOIN tb_pessoa AS pes ON par.cod_pessoa = pes.cod_pessoa WHERE p.classe = ?1 AND p.assunto = ?2 AND ");
		sb.append("(pes.nome IN (");
		for(int i = 3; i < 3 + numeroNomes; i++){
			sb.append("?" + i);
			if(i != (2 + numeroNomes)){
				sb.append(", ");
			}
		}
		sb.append(") OR pes.documento IN (");
		for(int i = 3 + numeroNomes; i < (3 + numeroNomes+ numeroDocumentos); i++){
			sb.append("?" + i);
			if(i != (2 + numeroNomes + numeroDocumentos)){
				sb.append(", ");
			}
		}
		sb.append("))");
		return sb.toString();
	}
}
</code>
```

Atentem que essa implementação é *apenas um exemplo*, não se tendo buscado assegurar características mais profundas de segurança ou de performance. No exemplo, não são considerados os assuntos que não o principal e é adotada estratégia única de verificação de prevenção em que é imprescindível a coincidência de classe e assunto, ainda que a posição das partes não seja relevante.

Definição do arquivo `components.xml`

No exemplo acima, vimos que a implementação reclama a existência de uma fonte de dados. Essa fonte deve ser definida no arquivo `components.xml`.

```
<code class="xml">
<?xml version="1.0" encoding="UTF-8"?>
<components xmlns="http://jboss.com/products/seam/components"
	xmlns:core="http://jboss.com/products/seam/core"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://jboss.com/products/seam/core http://jboss.com/products/seam/core-2.2.xsd
		 http://jboss.com/products/seam/components http://jboss.com/products/seam/components-2.2.xsd">
	<factory name="prevencaoDS" value="DATASOURCE_CONFIGURADO_NO_SERVIDOR_DE_APLICACAO" scope="application"/>
</components>
</code>
```

Empacotamento do ponto de extensão

O ponto de extensão deve ser empacotado em um arquivo JAR contendo a(s) classe(s) necessárias à implementação e, aindaa seguinte estrutura mínima:

- arquivo `seam.properties` vazio na raiz do arquivo JAR
- pasta META-INF contendo o arquivo `components.xml`

A par desse arquivo JAR, é imprescindível que o desenvolvedor do ponto de extensão identifique as bibliotecas de que depende, devendo, especialmente, verificar:

- se a biblioteca já faz parte do servidor de aplicação JBoss em que o PJe será instalado

- se a biblioteca já faz parte das bibliotecas de que o próprio PJe depende (WEB-INF/lib)
- se há alguma incompatibilidade entre as bibliotecas dependentes e aquelas já utilizadas no PJe

Cumpridos os passos acima, o desenvolvedor deverá elaborar um pacote ZIP contendo o JAR do ponto de extensão e os JARs das bibliotecas extras necessárias.

Instalação do ponto de extensão

A instalação do ponto de extensão é feita inserindo nos JARs contidos no pacote ZIP acima referido na pasta WEB-INF/lib do PJe. Sua utilização será feita por meio de injeção em componentes Seam do PJe ou indiretamente, em chamadas EL contidas em arquivos xhtml. Tanto a injeção quanto o uso em XHTMLs será feita por meio do nome padronizado de cada interface, que é o nome CamelCase da interface, sendo a primeira letra minúscula.

- pje-interfaces-1.0.0.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.0.jar)
- pje-interfaces-1.0.0-javadoc.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.0-javadoc.jar)
- pje-interfaces-1.0.1.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.1.jar)
- pje-interfaces-1.0.1-javadoc.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.1-javadoc.jar)
- pje-interfaces-1.0.2.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.2.jar)
- pje-interfaces-1.0.2-javadoc.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.2-javadoc.jar)
- pje-interfaces-1.0.3.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.3.jar)
- pje-interfaces-1.0.3-javadoc.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.3-javadoc.jar)
- pje-interfaces-1.0.4.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.4.jar)
- pje-interfaces-1.0.4-javadoc.jar (ftp://ftp.cnj.jus.br/pje/wikipje/pje-interfaces-1.0.4-javadoc.jar)

Interoperabilidade

O padrão **MNI (Modelo Nacional de Interoperabilidade)** (http://www.cnj.jus.br/tecnologia-da-informacao/comite-nacional-da-tecnologia-da-informacao-e-comunicacao-do-poder-judiciario/modelo-nacional-de-interoperabilidade) estabelece as bases para o intercâmbio de informações de processos judiciais e assemelhados entre os diversos órgãos de administração da Justiça. A **última especificação do padrão MNI** está disponível em Versão 2.2.2 (desde 07/07/2014) (http://www.cnj.jus.br/tecnologia-da-informacao/comite-nacional-da-tecnologia-da-informacao-e-comunicacao-do-poder-judiciario/modelo-nacional-de-interoperabilidade/arquivos-do-modelo-nacional-de-interoperabilidade) e as especificações das versões anteriores estão disponíveis em Versões Anteriores (http://www.cnj.jus.br/versoes-anteriores) .

Sistemas externos podem ser integrados ao PJe por meio de serviços disponíveis no padrão **MNI**, esses serviços são explicados na subseção a seguir.

Serviços disponíveis pelo *Web Service* do PJe

A partir da versão 1.4.6.4 do PJe, estão disponíveis por meio da instalação do PJe do tribunal o serviço *WSDL* ***intercomunicacao?wsdl***. Exemplo de acesso: *http://[[ENDEREÇO DA APLICAÇÃO DO PJe]]/intercomunicacao?wsdl*. Nesse *WSDL*, das seis operações previstas no MNI, quatro estão implementadas:

1. **consultarAvisosPendentes** - consulta a existência de avisos de comunicação processual pendentes (ou expedientes pendentes de ciência). Pode ser específica em relação a uma parte representada ou relativa aos processos em que o consultante opera como órgão de representação processual (MP, defensoria pública, advocacia pública, escritório de advocacia e advogado). Seu retorno contém os dados básicos, uma lista dos avisos pendentes de conhecimento pelo consultante. Caso não haja aviso pendente o retorno será uma lista vazia.
2. **consultarTeorComunicacao** - consulta o teor específico de uma comunicação processual pendente (ou um expediente pendente). No ato desta consulta, se o expediente em questão está pendente de ciência, o sistema registra automaticamente a ciência desse expediente.
3. **consultarProcesso** - consulta um processo judicial se o nível de sigilo interno permitir a consulta pelo requerente. Os documentos do processo poderão encerrar apenas binários encriptados cuja chave será fornecida na consultaTeorComunicacao, caso haja intimação pendente para o documento transferido.
4. **entregarManifestacaoProcessual** - permite a entrega de manifestação processual, por órgão de representação processual ou por advogado, e a entrega de petição inicial. Retorna os dados básicos, o número do protocolo, a data da operação e, se bem sucedida, documento PDF contendo o recibo.
   A operação entregarManifestacaoProcessual também poderá ser utilizada para resposta de atos de comunicação (ou expedientes), desde que o referido expediente já tenha ciência dada. Para responder um expediente por meio dessa operação é preciso atentar-se para "montagem" do atributo "parametros" (que é um atributo do objeto "ManifestacaoProcessual"). O parâmetro que deve ser incluído é o par nome/valor assim: nome "mni:idsProcessoParteExpediente" e o valor será a sequência de ID´s dos expedientes. Para que a operação "entregarManifestacaoProcessual" identifique que essa entrega de manifestação é uma resposta de um expediente é preciso conter o parâmetro "mni:idsProcessoParteExpediente". Reproduzimos aqui um exemplo parcial por meio da linguagem *Java* da "montagem" do atributo "parametros":

```
ManifestacaoProcessual manifestacaoResposta = new ManifestacaoProcessual();
//Parâmetro responsável por armazenar os ID's (identificadores) dos EXPEDIENTES que serão respondidos.
Parametro parametro = new Parametro();
parametro.setNome("mni:idsProcessoParteExpediente");
parametro.setValor("113"); //Separe os ID's dos EXPEDIENTES com ';' caso deseje passar mais de um.
manifestacaoResposta.getParametros().add(parametro);
```

```
-----------------------------------------------------
Recomendação aos interessados na integração com o PJe:
-----------------------------------------------------
Recomendamos fortemente que a documentação (arquivos xsd, documentação Javadoc, documento pdf, etc) contida na especificação do MNI
seja analisada, pois a consulta à essa documentação é muito importante para compreensão da estrutura dos objetos das requisições e
das respectivas respostas. No caso de dúvidas, envie email para ''g-assistencia.desenvolvimento.pje@cnj.jus.br''.
```

Serviços para consultas complementares

Para auxiliar na integração com outros sistemas, o PJe também dispõe de serviços de **consultas** às informações sobre configurações e tabelas básicas referentes à instalação do PJe. Pode-se acessar a lista de operações de consulta por meio da instalação do PJe no tribunal, complementando o endereço de acesso inicial do PJe com o trecho ***ConsultaPJe?wsdl***. Esse é um acesso disponível em toda instalação do PJe.

Exemplo de acesso: *http://[[ENDEREÇO DA APLICAÇÃO DO PJe]]/ConsultaPJe?wsdl*.

As operações de consulta disponíveis são:

1. **consultarJurisdicoes()** - Lista todas as jurisdições (chamadas de localidades no MNI) ativas cadastradas no PJe.
2. **consultarOrgaosJulgadores()** - Lista todos os órgãos julgadores ativos cadastrados no PJe.
3. **consultarOrgaosJulgadoresColegiados()** - Lista todos os órgãos julgadores ativos cadastrados no PJe.
4. **consultarClassesJudiciais(<objeto do tipo Jurisdicao>)** - Lista todas classes judiciais pertencentes às competências ativas configuradas para a jurisdição informada. O parâmetro Jurisdicao é obrigatório.
5. **consultarAssuntosJudiciais(<objeto do tipo Jurisdicao>, <objeto do tipo ClasseJudicial>)** - Lista todos os assuntos judiciais ativos possíveis para a classe judicial informada a partir da configuração de competência na jurisdição informada. O parâmetro Jurisdicao é obrigatório.
6. **consultarCompetencias(<objeto do tipo Jurisdicao>, <objeto do tipo ClasseJudicial>, <coleção de objetos do tipo AssuntosJudiciais>)** - Lista todas as competências ativas. O parâmetro Jurisdicao é obrigatório e os demais parâmetros são opcionais; esses parâmetros restringem o resultado por jurisdição e, opcionalmente, por classe judicial e/ou por lista de assunto judicial.
7. **consultarTiposAudiencia()** - Lista os tipos de audiência ativos cadastrados no PJe.
8. **consultarTiposDocumentoProcessual(<identificadorPapel>)** - Lista os tipos de documentos ativos cadastrados no PJe. Além desse filtro, os tipos de documentos retornados possuem algum vínculo com algum papel/perfil do sistema. O parâmetro identificadorPapel é opcional. Pode ser usado para restringir a lista para um determinado papel/perfil configurado no controle de acesso do PJe.
9. **recuperarInformacoesFluxo(<objeto do tipo ClasseJudicial>)** - Recupera o *XML* do fluxo configurado para a classe judicial informada.
10. **consultarSalasAudiencia(<objeto do tipo OrgaoJulgador>)** - Recupera a lista de salas de audiência ativas cadastradas no PJe. O parâmetro OrgaoJulgador é opcional e pode ser usado para restringir a lista para um determinado órgão julgador.
11. **consultarPrioridadeProcesso()** - Recupera a lista de prioridades processuais ativas cadastradas no PJe.

Verifique no *WSDL* gerado na instalação do PJe o conteúdo retornado e a assinatura de cada operação.

Autenticação

Como pré-requisito usar as operações do MNI disponíveis no PJe, deve ser feita a autenticação dos clientes por meio da inscrição na Receita Federal (CNPJ ou CPF) extraída de certificados digitais ICP-Brasil. O protocolo de comunicação deve ser *https*.

Os órgãos de representação processual (MP, Defensoria Pública, Advocacia Pública e escritório de advocacia) são tratados no PJe como procuradorias. O cliente pode utilizar o certificado institucional ou pessoal. No primeiro caso, o CPF do responsável deve ser o mesmo que está associado à procuradoria cadastrada no PJe.

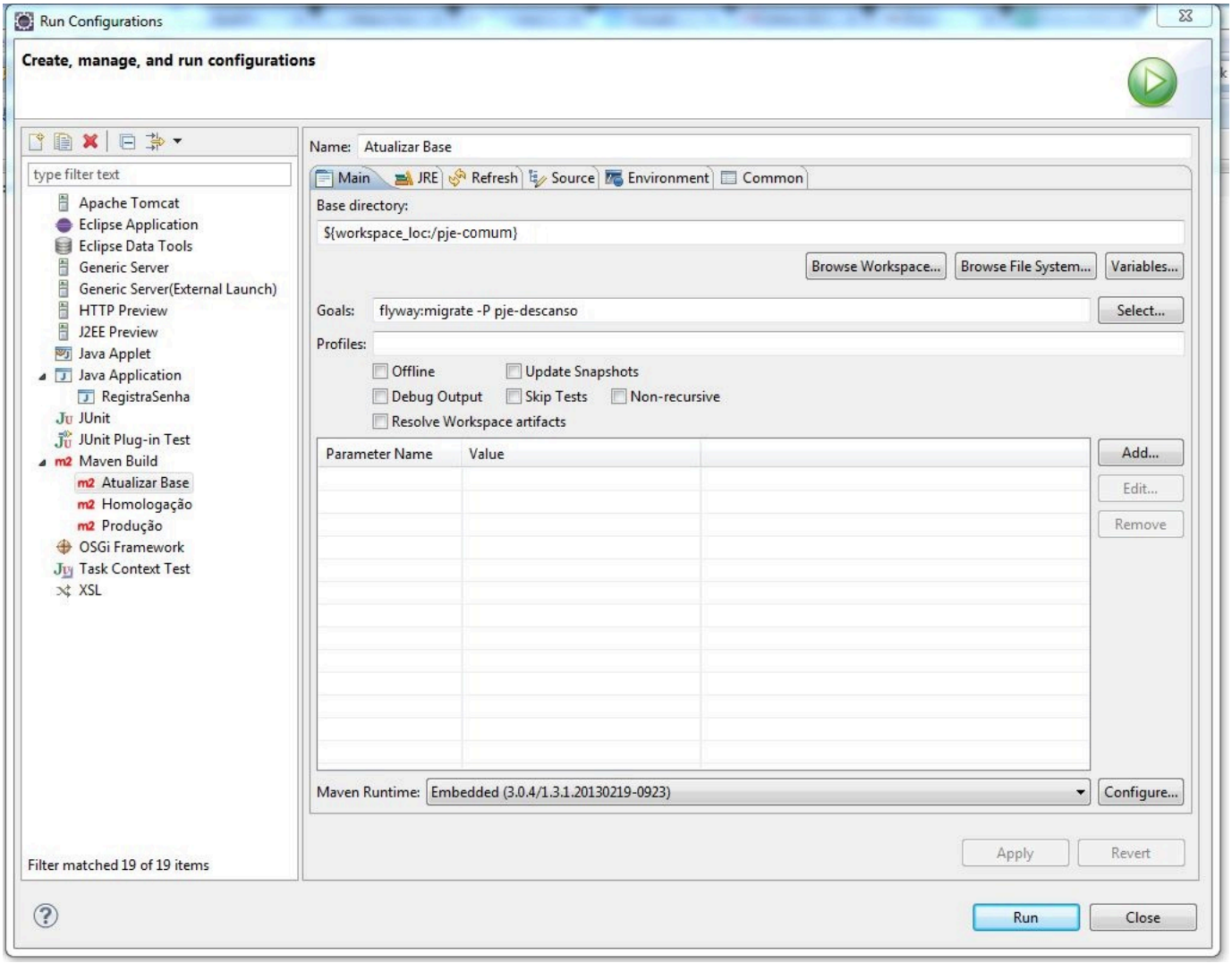
Flyway

O Flyway é um sistema de versionamento de banco de dados. Funciona como um plugin no maven. Anteriormente, no PJe, vinha se usando o Flyway através do projeto pje-dbmanager, mas, devido às dificuldades de integração de código e integração de banco em projetos separados, optou-se, na versão 1.5.0, por reintegrar os scripts do dbmanager ao PJe e utilizar o flyway direto no maven desse momento em diante. Para configurá-lo no projeto PJe, deve-se modificar o arquivo .m2/settings.xml e acrescentar os seguintes parâmetros:

```
<servers>
...
  <server>
    <id>pje-descanso</id>
    <username>postgres</username>
    <password>123456.</password>
  </server>
...
</servers>
<profiles>
...
  <profile>
    <id>pje-descanso</id>
    <properties>
      <flyway.serverId>pje-descanso</flyway.serverId>
      <flyway.driver>org.postgresql.Driver</flyway.driver>
      <flyway.url>jdbc:postgresql://localhost:5432/pje_1_4</flyway.url>
    </properties>
  </profile>
...
</profiles>
```

No início do trecho de arquivo, a configuração do <server> é a autenticação que deve ser utilizada para o <profile> de nome "pje-descanso", configurado mais abaixo. A configuração do <profile> contém o servidor (<flyway.serverId>) onde será executada a atualização, estabelecendo o driver para acesso ao banco (<flyway.driver>) e o endereço onde o banco se encontra (<flyway.url>).

Para atualizar a base, execute o comando "mvn flyway:migrate -P pje-descanso" através do eclipse, conforme imagem abaixo:



onde "pje-descanso" é o <flyway.serverId> definido no arquivo de configuração acima descrito.

A atualização de banco em versões anteriores é feita através do projeto dbmanager (que não mais existe a partir da 1.5.0), utilizando a classe atualizarBase.java. Ela deve ser executada diretamente pelo eclipse.

Catálogo de serviços

O PJe faz uso de comunicação com diversos serviços externos, como consulta à Receita Federal, consulta a instituições financeiras, consulta à OAB. Os endereços desses serviços ficam centralizados no Catálogo de serviços (http://www.cnj.jus.br/catalogo_de_servicos) . Para ter acesso ao catálogo, os tribunais devem solicitar ao CNJ através do email **g-assistencia.qualidade.pje@cnj.jus.br**.

Configuração para uso do *jcr-storage*

1. Baixar o projeto do *git(git@git.cnj.jus.br:utilitarios/jcr-storage.git)*.
2. Gerar o pacote "*mvn clean package -DskipTests*".
3. Entrar na pasta do projeto *jcr-storage/server/target* para iniciar o serviço:
    1. Executar o comando "*nohup java -Xmx2048m -jar jcr-storage-server-xec.jar -httpPort 9000*".
4. Incluir as seguintes dependências na pasta *lib* do *Jboss*:
    1. *commons-io-1.4.jar*
    2. *commons-lang3-3.1.jar*
    3. *jcr-storage-client-x.x.x.jar*
    4. *jcr-storage-common-x.x.x.jar*
5. Criar o arquivo "*jcr-storage.properties*" dentro da pasta *lib* (citada anteriormente) com os seguintes valores:
    1. *jcr.url=http://localhost:9000/storage/documents*
    2. *jcr.username=usuario*
    3. *jcr.password=senha*

Configuração para uso do *Storage*

DB-STORAGE

1. Baixar os arquivos do linque abaixo e informar a senha "pje@CNJ" http://www.cnj.jus.br/owncloud/public.php?service=files&t=bb1aecbc1cbf34bb0dc9bbb79b05bf9c
2. Criar um banco para o db-storage conforme o script create-db-storage.sql
3. Copiar as dependências da pasta "lib-jboss" para a pasta lib do servidor jboss.
    1. Configurar arquivo db-storage.properties para apontar para o banco criado.

JCR-STORAGE

1. Baixar os arquivos do linque abaixo e informar a senha "pje@CNJ" http://www.cnj.jus.br/owncloud/public.php?service=files&t=56c24847d6ce3c82f7fbf8ecd8b71ab1
2. Entrar na pasta "server" para iniciar o serviço.
    1. Mudar as propriedades "jcr.server.tempDir" e "jcr.server.repoDir" do arquivo "server.properties" para apontar onde o repositório deve ser criado.
        1. Iniciar o serviço conforme comando.
            1. java -jar jcr-storage-server-exec.jar -httpPort 9000 -Dbr.jus.cnj.jcr.serverProperties=<CAMINHO_DO_ARQUIVO>/server.properties
3. Copiar as dependências da pasta "lib-jboss" para a pasta lib do servidor jboss.
    1. o arquivo jcr-storage.properties não precisa ser mudado, pois já está com as configurações padrão. Somente precisa ser mudado se o servidor do jcr estiver sendo executado em outra máquina.

Serão gerados vários pacotes com as dependências dentro do próprio war e disponibilizados no ftp.

Configuração dos dados de conexão

db-storage - incluir na tabela tb_parameto, os seguintes parâmetros.

Variável	Descrição	Valor
db.driver	Driver de conexão	org.postgresql.Driver
db.url	Url de conexão	jdbc:postgresql://localhost:5432/dbstorage
db.username	Usuario de conexão	usuario
db.password	Senha de conexão	senha

jcr-storage - incluir na tabela tb_parameto, os seguintes parâmetros.

Variável	Descrição	Valor
jcr.url	Url de conexão	http://localhost:9000/storage/documents
jcr.username	Usuario de conexão	Usuario
jcr.password	Senha de conexão	Senha

Por questão de segurança, é recomendável que as informações de usuário e senha não fiquem no banco de dados. Nesse caso deve-se criar um arquivo properties no servidor

db-storage

1. Criar um arquivo "db-storage.proprerties" com as seguintes informações
    1. db.username=usuario
    2. db.password=senha
2. Passar esse arquivo como parâmetro para a jvm
    1. Alterar o arquivo /bin/run.conf dentro da pasta do jboss colocando ao final da variável “JAVA_OPTS” o seguinte parâmetro “-Dbr.jus.cnj.pje.db-storage.configuration=<caminho_do_arquivo>/db-storage.properties”

jcr-storage

1. Criar um arquivo "jcr-storage.proprerties" com as seguintes informações
    1. jcr.username=usuario
    2. jcr.password=senha
    3. Passar esse arquivo como parâmetro para a jvm
    4. Alterar o arquivo /bin/run.conf dentro da pasta do jboss colocando ao final da variável “JAVA_OPTS” o seguinte parâmetro “-Dbr.jus.cnj.pje.jcr-storage.configuration=<caminho_do_arquivo>/jcr-storage.properties”

Revisão de Código

Objetivo da revisão

- Encontrar e corrigir possíveis erros antes que o código seja integrado ao produto;
- Disseminar conhecimento relativo as ferramentas e ao projeto como um todo através da interação entre desenvolvedores e revisores;
- Aumentar a qualidade geral do código através do controle de padrões e técnicas aplicadas ao projeto;
- Aumentar a confiança de stakeholders com a forma e aplicação de mudanças no produto.

Papeis

- Desenvolvedor: aquele que produz o código e solicita sua revisão;
- Revisor: aquele que executa a revisão e solicita alterações na solução, se for o caso.

Boas práticas

- Desenvolvedor
    - A primeira revisão é sempre a do desenvolvedor;
    - Faça o seu checklist do que será analisado na revisão de código;
    - Não encare os erros encontrados como uma crítica a você;
    - Não reescreva o código sem ajuda ou consulta de colegas mais experientes;
    - Não desenvolva isoladamente, procure consultar os colegas, apresente seu trabalho e solicite sugestões;
    - Ajude a manter os padrões mínimos de codificação do projeto.
- Revisor
    - Revise o código e não os programadores
    - Trate os desenvolvedores com menos conhecimento com paciência.

- Lembre-se que um dos prós da tarefa de revisão é o ganho de conhecimento.
- Faça perguntas ao invés de afirmações. Ex.: Este método parece não estar sendo utilizado, correto?
- Tenha um padrão bem definido. Não há como cobrar por um padrão que não existe.
- Um problema pode ter várias soluções, aprenda a avaliar e sugerir novas soluções se for o caso.
- Não acelere uma revisão, mas não esqueça que sua revisão é aguardada por sua equipe e cliente.

Tipos de Revisão

- Individual
  - Revisão simples onde um revisor avalia e aprova, se for o caso, promovendo a integração da solução ao código do projeto.
- Técnica Ampliada
  - Revisores avaliam demandas mais complexas, com objetivo de identificar erros e possíveis melhorias na solução. Por fim aprovam e integram o código. Este tipo de revisão é receptiva a participação de vários desenvolvedores como espectadores.

Objetos de Revisão

1. **Do processo de criação do branch e respectivo Merge Request**
    1. O branch de desenvolvimento obedece ao padrão PJEII-<Nº ISSUE> ou PJEII-<Nº ISSUE>-<Versão>?
    2. Os commits do branch possuem comentários descritivos sobre o que está sendo alterado?
    3. Os commits do branch possuem comentários que obedecem ao padrão [PJEII-?????]<Descrição>?
    4. O branch do desenvolvedor está sincronizado com branch origem?
    5. O Merge Request foi criado com título obedecendo o padrão: [PJEII-?????][<órgão_desenvolvedor>]<Descrição>?
    6. Caso o MR tenha sido fechado, o mesmo MR deverá ser reaberto contendo as correções sugeridas pela equipe de revisão.

2. **Do escopo da implementação**
    1. As alterações promovidas na implementação estão dentro do escopo da demanda genitora da correção/nova funcionalidade?

3. **Da análise do código**
    1. Estrutura do Projeto
        1. As alterações nos fontes afetam de forma nociva o encoding padrão do projeto?
        2. Possíveis recursos adicionados ao projeto estão nos pacotes adequados?
    2. Variáveis e Atributos
        1. As variáveis criadas utilizam nomes coerentes com sua finalidade?
        2. Os tipos das variáveis são adequados para resolução do problema?
        3. Existem variáveis declaradas não utilizadas?
        4. As variáveis foram corretamente inicializadas?
        5. Existem atributos que deveriam ser variáveis locais?
        6. Os atributos utilizam modificadores de acessos adequados?
    3. Métodos
        1. Os métodos implementados possuem modificadores de acesso adequados ao seu propósito e utilização?
        2. Os nomes dos métodos descrevem de maneira clara o seu propósito?
        3. Os parâmetros passados ao método são verificados antes de sua efetiva utilização?
        4. Os métodos possuem escopo bem definido?
    4. Classes
        1. Os nomes das classes descrevem bem seu propósito?
        2. A classe possui modificador de acesso adequado?
        3. A classe implementa corretamente as interfaces que utiliza?
        4. As extensões da classe são adequadas a seu propósito?
        5. A classe possui escopo bem definido?
        6. As entidades estão mapeadas corretamente observando nomes de colunas e cardinalidades?
        7. As entidades possuem coleções com FetchType adequado (Lazy e Eager)?
    5. Implementação Geral
        1. O código implementado é repetido ou demasiadamente semelhante a outro já presente no projeto?
        2. A framework está sendo utilizada da maneira correta?
        3. A arquitetura está sendo utilizada da maneira correta (Action, Manager e DAO)?
        4. As exceções são tratadas de forma correta, e não genericamente?
        5. Foram verificados possíveis pontos de NPE?
        6. A implementação utiliza de maneira correta os escopos do framework (evento, página, conversação, sessão e aplicação)?
        7. A camada de visão está livre de lógica de negócio?
        8. A lógica aplicada ao controle de fluxo dos algoritmos implementados é adequada, não permitindo simplificação imediata?
        9. Os loops utilizados apresentam saídas coerentes e não possibilitam ciclos infinitos?
        10. Existem testes desnecessário nos algoritmos?
        11. Os parâmetros de sistema estão sendo utilizados corretamente observando possível nulidade?
        12. O código está livre de implementações hardcoded?
        13. Os recursos alterados são críticos e capazes de provocar impactos severos no funcionamento geral da aplicação?
        14. As funcionalidades implementadas observam aspectos de segurança e controle de acessos aos dados?
        15. O código compila?
        16. Foram retirados imports desnecessários?
        17. O código está bem identado?
        18. O código utiliza de maneira adequada o log da aplicação?
        19. Há parte do código que pode ser substituído por alguma outra função já implementada?
        20. Há código comentado?
    6. Scripts SQL
        1. Os scripts estão separados em script DDL e script DML, conforme a necessidade da demanda?
            1. Não deve-se juntar script DDL e DML em um único script.
        2. O script possui nomenclatura correta de forma a manter íntegra a execução do aplicativo Flyway?
            1. A nomenclatura do script SQL deverá obedecer a seguinte fórmula (ou sintaxe):
                1. **PJE**_*{informar aqui a numeração do pacote da versão do PJe correspondente}*_*{informar aqui a ordem numérica sequencial crescente correspondente}*__*{informar aqui a sigla **DDL** se o script for de DDL; ou a sigla **DML** se o script for de DML}*_*{informar aqui de forma geral o que o script está proposto a fazer}*.**sql**.
                Veja um exemplo: PJE_2.0.1.0_56__DDL_CRIAR_TABELA_LIBERACAO_PUBLICACAO_DECISAO.sql

1. Por convenção, o script inicial de todo pacote de versão sempre será PJE_{informar aqui a numeração do pacote da versão do PJe correspondente}_1__VERSAO_INICIAL.sql; exemplo: PJE_1.7.2.0_1__VERSAO_INICIAL.sql
2. O conteúdo do script inicial deverá conter apenas os comentários: data da criação em dd/mm/aaaa e o texto "Arquivo inicial dos scripts da versão {informar aqui a numeração do pacote da versão do PJe correspondente}"; veja o exemplo:

```
/*
 * 19/01/2014
 * Arquivo inicial dos scripts da versão 1.7.2.0
 */
```

3. O script possui cabeçalho de acordo com o padrão?
    1. O conteúdo do cabeçalho deverá conter: número da issue e uma descrição resumida do propósito do script; veja o exemplo:

```
/*
 * NR DA ISSUE: PJEII-19483.
 * DESCRIÇÃO: Inclusão da coluna in_temporario na tabela tb_pess_doc_identificacao.
 *            Esta coluna serve para indicar se um documento de identificação pode (ou não)
 *            ser manipulado (alterado ou deletado) pelo usuário externo que o incluiu.
 */
```

4. As tabelas criadas possuem colunas e *constraints* adequados ao seu propósito?
5. Scripts de inserção de parâmetros da aplicação checam a prévia existência do mesmo?
6. O tipo de dado utilizado para as colunas é adequado?
7. Há instruções SQL (sejam do script de DML ou de DDL) comentadas e sem nenhum valor significativo para compreensão de seu propósito?
    1. Para maior clareza das instruções SQL, recomendamos não deixar instruções comentadas.
8. Scripts de DDL checam a prévia existência do objeto que será criado/alterado/excluído?
    1. Dado um exemplo de checagem no caso de criação de uma nova coluna em uma tabela existente em um esquema existente no SGBD PostgreSQL:

```
/*
* NR DA ISSUE: {informar aqui o número da ISSUE}
* DESCRIÇÃO: {informar aqui a descrição resumida do propósito do script}
*/

DO $$
BEGIN
    IF NOT EXISTS (
        SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
        WHERE
        TABLE_SCHEMA =  /* Informar aqui entre aspas simples o nome do esquema da forma que foi criado;
                        para o PJe adota-se a escrita do nome em letras minúsculas.*/
        AND
        TABLE_NAME =    /* Informar aqui entre aspas simples o nome da tabela da forma que foi criado;
                        para o PJe adota-se a escrita do nome em letras minúsculas.*/
        AND
        COLUMN_NAME =   /* Informar aqui entre aspas simples o nome da nova coluna que deseja criar;
                        para o PJe adota-se a escrita do nome em letras minúsculas.*/  ;
    )   THEN
        /* ESCREVER AQUI O SCRIPT DDL DE CRIAÇÃO DA NOVA COLUNA DA TABELA EM QUESTÃO. */
    END IF;
END $$;
```

9. Nos scripts de DDL que criam novos objetos, há nomes de novos objetos cuja quantidade de caracteres é superior a 30 caracteres?
    1. Há uma demanda antiga do PJe (issue PJEII-5829 (http://www.cnj.jus.br/jira/browse/PJEII-5829)) que prevê a compatibilidade com o SGBD Oracle. Dada essa necessidade, recomendamos que todos os scripts de DDL que criam novos objetos (exemplo: *column, table, view, trigger, constraint, function, sequence, index*) contenham os nomes desses objetos com uma quantidade de caracteres **que não seja superior a 30** caracteres.
10. **Exclusivamente para o Revisor de Código:** os scripts DDL já foram validados e aprovados pela área de Administração de Dados(AD)? Para solicitar a validação do script DDL pela equipe de AD, basta colocar a issue na situação "Validação do Modelo de Dados" no JIRA. Feito isso, um dos membros da equipe de AD irá analisar e validar os scripts.

4. **Da documentação interna**
    1. As classes e métodos implementados foram documentados utilizando os recursos do Javadoc?
    2. Os métodos possuem documentação Javadoc que explica de modo claro e objetivo o seu propósito?
    3. Algoritmos complexos possuem documentação interna descrevendo a lógica aplicada?
    4. Regras negociais complexas são acompanhadas de documentação explicativa e referenciam suas respectivas issues?
