**Configurando JPA no servidor de aplicações**

**Criando o datasource de conexão com SQLServer**

1. Insira a seguinte configuração dentro da tag <**drivers**> no arquivo **standalone.xml** do servidor Wildfly.

<driver name="sqlserver" module="com.microsoft.sqlserver">

<xa-datasource-class>

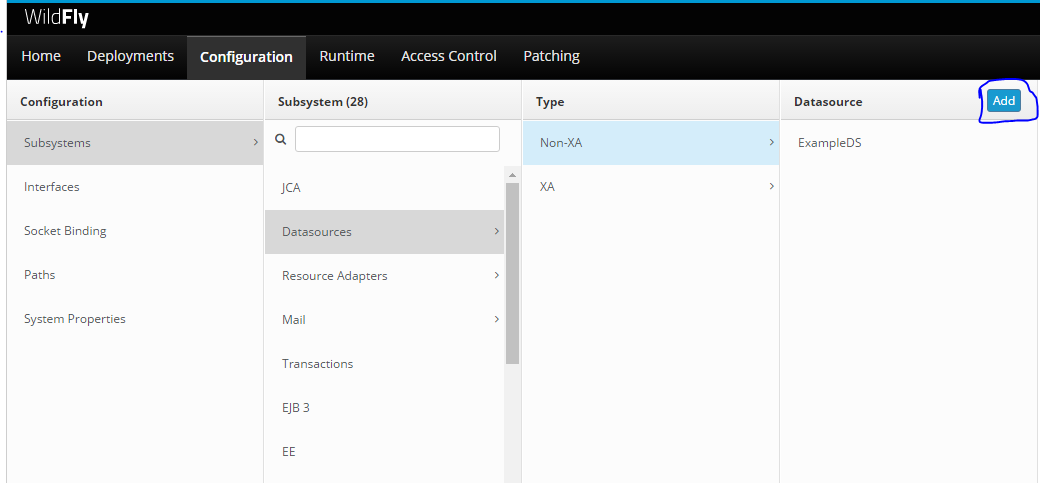
com.microsoft.sqlserver.jdbc.SQLServerXADataSource

</xa-datasource-class>

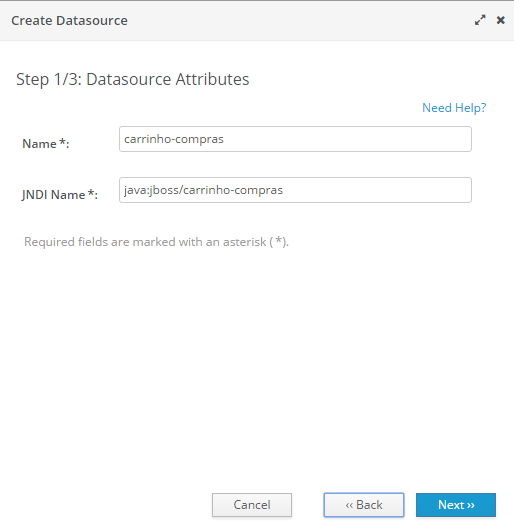
</driver>

**Obs.:** Pode ser feito o deploy do arquivo .**jar** do driver SQLServer diretamente no servidor de aplicação, como se fosse um arquivo **.war**. Contudo, isso pode gerar um problema: as vezes o jar do drive é excluído pelo servidor e é necessário realizar o deploy do **jar** novamente a cada inicialização do Wildfly**.**

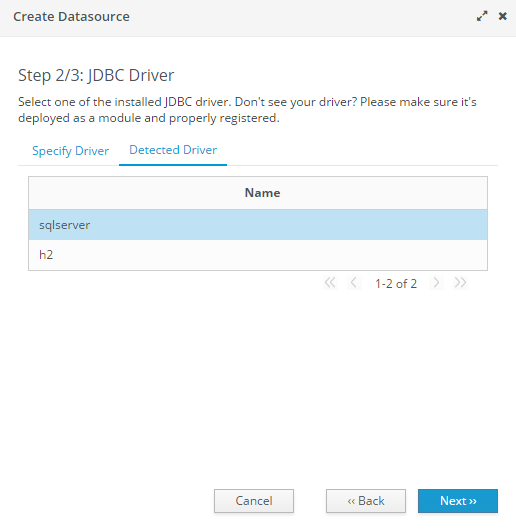
1. Copie a pasta (disponibilizada durante o treinamento) sqlserver para dentro do diretório **\modules\system\layers\base\com** que fica na pasta de instalação do servidor de aplicação.
2. Crie o datasource



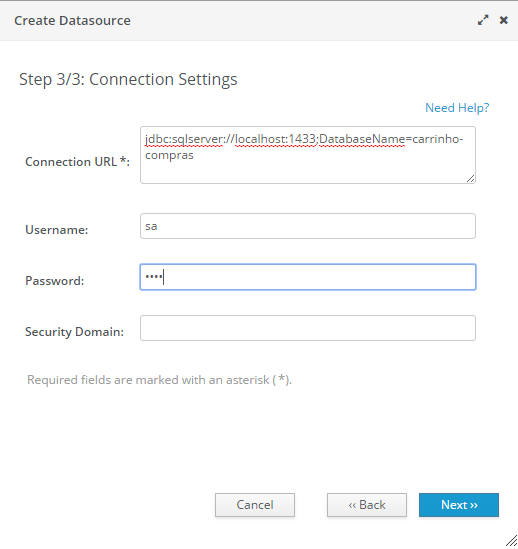
1. Insira as informações do datasource



1. Escolha o driver adicionado anteriormente no arquivo **standalone.xml**



1. Informe informações do banco e *login*.



* Após a criação do datasource, teste (botão “**Test Connection**”) a conexão o banco de dados. Caso esteja tudo certo, deverá aparecer a seguinte mensagem: “**Successfully connected to database test**”

**Configurando o persistence.xml**

Em qual banco de dados vamos gravar nossas entidades? Qual é o login? Qual é a senha? O JPA necessita dessas configurações, e para isso criaremos o arquivo persistence.xml.

<persistence-unit name="AlgumNome" transaction-type="JTA">

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

<jta-data-source> java:jboss/carrinho-compras</jta-data-source>

<properties>

<property name="hibernate.hbm2ddl.auto" value="none"/>

<property name="hibernate.dialect" value="org.hibernate.dialect.SQLServer2008Dialect" />

<property name="hibernate.show\_sql" value="true" />

<property name="hibernate.format\_sql" value="true" />

</properties>

</persistence-unit>

No arquivo acima podemos notar a presença de duas configurações importantes:

* **transaction-type** com valor "JTA": para identificar que as transações serão controladas pelo servidor de aplicação não localmente, ou seja, por sua aplicação (default RESOURCE\_LOCAL).
* **jta-data-source**: determina onde o contexto de persistência encontrará a conexão com o banco de dados, uma vez que ela não está configurada na aplicação. Esta tag apenas recebe um identificador para obter uma conexão com o banco, este identificador foi configurado anteriormente dentro do servidor de aplicações.

Como a configuração de persistência está no servidor de aplicações, o tratamento de instanciação e controle de transações é feito pelo servidor, sendo assim você, por padrão, não poderá instanciar e abrir transações no banco de dados. Para fazer uso de transações será necessário o uso de outra API chamada EJB.

**EJB**

É um componente da plataforma JEE que roda em um container de um servidor de aplicação. Seu principal objetivo consiste em fornecer um desenvolvimento rápido e simplificado de aplicações Java, com base em componentes distribuídos, transacionais, seguros e portáveis.

Para definir um EJB e ter acesso a todos os benefícios listados acima, apenas precisamos anotar uma classe (DAO) com uma das seguintes anotações:

* **@Stateless**: que não mantém o estado entre as requisições, ou seja, uma nova instância a cada chamada de método.
* **@Statefull**: que mantém o estado entre as requisições, ou seja, uma instância única entre todas as chamadas de métodos.

@Stateless

public class TestEJB {

@PersistenceContext

private EntityManager em;

public void salvar() {

Produto produto = new Produto();

produto.setDescricao("produto managedBean");

produto.setPrecoUnitario(10d);

em.persist(produto);

}

}

Para fazer uso dos **EJB's** definidos é somente necessário a declaração de um atributo e anotá-lo com a anotação **@Inject** do pacote **javax.inject.Inject**, isso indicará ao servidor de aplicações injetar uma instancia do referido EJB no atributo anotado, conforme exemplo abaixo:

@Inject

private TestEJB testEJB;

public void save() {

testEJB.salvar();

}

**Referências**

**Livro de apoio**: <http://alga.works/livro-javaee/>

**Tutorial JPA**: <http://www.objectdb.com/java/jpa>

**EJB Documentção**: <http://docs.oracle.com/javaee/7/tutorial/partentbeans.htm#BNBLR>