02

Preparando o ambiente

Durante as formações do Alura vimos muitas tecnologias web, nas trilhas de front-end e java. Nesse curso nosso objetivo é integrar as tecnologias e linguagens que foram estudadas para desenvolvermos um projeto web para controle de horários dentro de uma empresa.

Para esse projeto, utilizaremos o VRaptor 4 no desenvolvimento da parte web do projeto. Para ler e gravar informações no banco de dados, utilizaremos o JPA (Java Persistence API) com o banco de dados MySQL. O layout da aplicação será desenvolvido com uma biblioteca de componentes CSS chamada Twitter Bootstrap junto com as taglibs da JSTL. As dependências do projeto serão gerenciadas através do Maven e, por fim, para fazermos o controle de versão do código, utilizaremos o git.

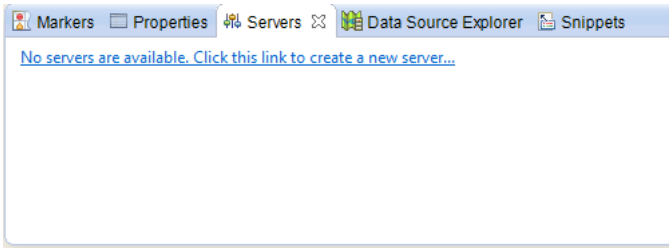
Baixando os componentes necessários

Agora que já sabemos o objetivo desse curso, vamos começar nosso projeto web. A forma mais simples de iniciarmos um projeto web com o VRaptor 4 é baixando o blank project que pode ser encontrado no bintray do Vraptor (https://bintray.com/caelum/VRaptor4/br.com.caelum.vraptor/ (https://bintray.com/caelum/VRaptor4/br.com.caelum.vraptor/)) a versão que será utilizada no curso é a 4.1.1 que pode ser baixada nesse link: http://dl.bintray.com/caelum/VRaptor4/vraptor-blank-project-4.1.1.zip (http://dl.bintray.com/caelum/VRaptor4/vraptor-blank-project-4.1.1.zip)

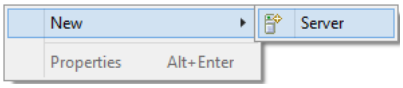
Além do blank project, também precisaremos de um servidor web para hospedar o projeto em nossas máquinas, nesse curso o servido utilizado será o Tomcat versão 7 versão 7.0.56 (que pode ser baixado nesse endereço: https://archive.apache.org/dist/tomcat/tomcat-7/v7.0.56/bin/apache-tomcat-7.0.56.zip (https://archive.apache.org/dist/tomcat/tomcat-7/v7.0.56/bin/apache-tomcat-7.0.56.zip)).

Configurando o ambiente do projeto

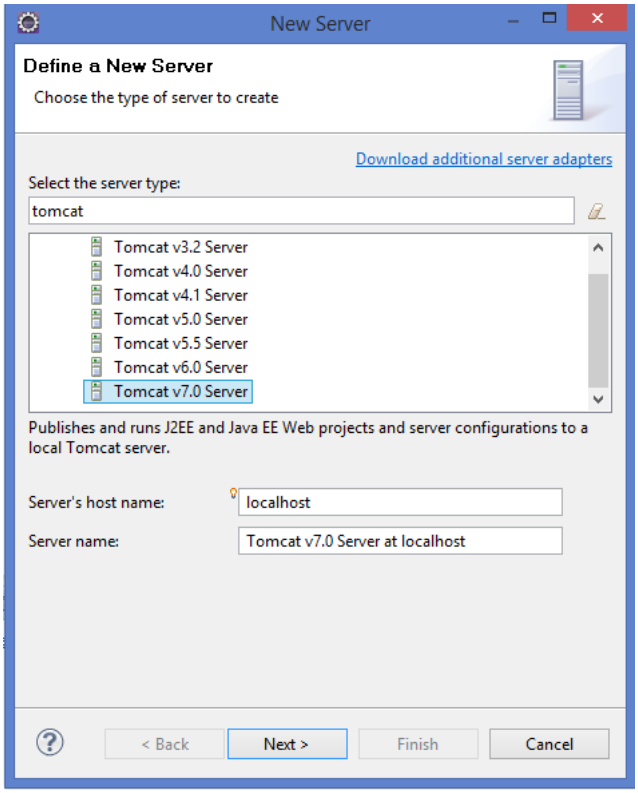
Agora que baixamos os componentes necessários, vamos configurar nosso ambiente de desenvolvimento. Depois de descompactar os arquivos baixados, abra o eclipse for Java EE developers e dentro dele procure a aba chamada `Servers`:



Dentro dela, clique com o botão direito do mouse e no menu mostrado, selecione a opção `New > Server`:

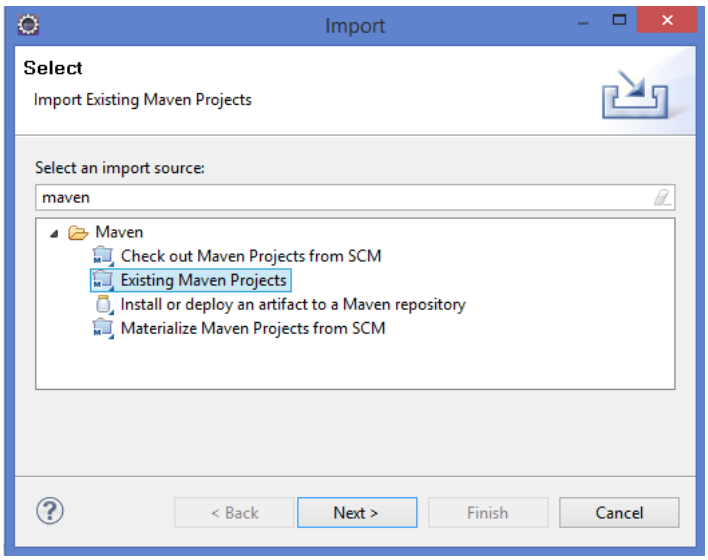


Na janela que é aberta pelo eclipse, selecione a opção `Tomcat v7.0 Server` e depois clique em `Next`.

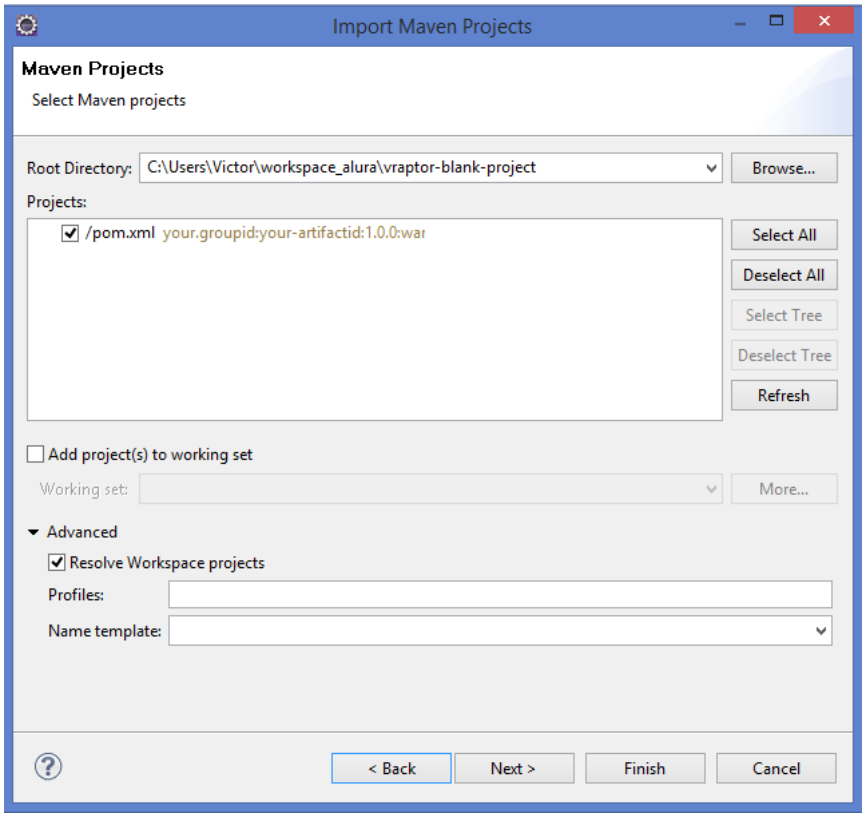


Na próxima janela, precisamos indicar ao eclipse onde está a pasta com a instalação do Tomcat. Clique no botão `Browse` e escolha a pasta onde o tomcat foi descompactado. Depois de configurar a pasta de instalação do Tomcat, clique no botão `Finish`.

Agora que o servidor foi configurado, é hora de importarmos o blank project para dentro do workspace do eclipse. Acesse o menu `File > Import` e dentro da janela aberta, escolha a opção `Existing Maven Projects`:



Na próxima janela, clique no botão `Browse` e escolha o local onde você descompactou o vraptor blank project e clique no botão `Finish`:



Além de importarmos o projeto, precisamos também criar o banco de dados que será utilizado dentro do MySQL. Abra o terminal do sistema e dentro dele digite o comando:

```
mysql -u root
```

Agora que a conexão com o banco de dados foi aberta, podemos criar o banco do curso com o seguinte comando:

```
create database alura_horas;
exit;
```

Mapeamento das entidades com a JPA

Agora que nosso ambiente de desenvolvimento está configurado vamos configurar a JPA e as entidades que serão mapeadas para o banco de dados. A instalação da JPA no projeto será feita através do Maven.

Abra o arquivo `pom.xml` do projeto e dentro dele procure a tag `dependencies` . Antes do fechamento dessa tag, declare a dependência para o `hibernate-entitymanager` e o driver para o mysql, o `mysql-connector-java` :

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>4.3.6.Final</version>
</dependency>

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.33</version>
</dependency>
```

Ainda dentro do `pom.xml`, vamos modificar o nome do artefato que será gerado pelo maven, para isso, no começo do arquivo, procure a tag `artifactId` e mude seu conteúdo para `alura-horas`:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-i
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0
    <modelVersion>4.0.0</modelVersion>

    <groupId>br.com.alura</groupId>
    <artifactId>alura-horas</artifactId>
        <!-- resto do pom.xml -->
```

Agora que já declaramos as dependências no `pom.xml`, podemos começar a criar as entidades para o sistema. Para esse projeto, utilizaremos duas entidades, o `Usuario` e a `HoraLancada`:

```
@Entity
public class Usuario {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int id;

    @NotEmpty
    private String nome;

    @NotEmpty
    private String login;

    @NotEmpty
    private String senha;

    @NotEmpty
    @Email
    private String email;

    // getters e setters
}

@Entity
public class HoraLancada {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int id;

    @NotNull
    @Temporal(TemporalType.DATE)
    private Calendar data;

    private String comentario;

    @NotEmpty
    @Pattern(regexp="\\d{2}:\\d{2}")
    private String horaInicial;
```

```
    @NotEmpty
    @Pattern(regexp="\\d{2}:\\d{2}")
    private String horaFinal;

    @ManyToOne
    private Usuario usuario;

    // getters e setters
}
```

Para terminar a configuração da JPA, precisamos criar o arquivo `persistence.xml` dentro da pasta `src/main/resources/META-INF` do projeto.

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
        http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
    version="2.0">
    <persistence-unit name="default">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>

        <properties>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/horas" />
            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="" />
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect'
            <property name="hibernate.hbm2ddl.auto" value="update"/>
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
        </properties>
    </persistence-unit>
</persistence>
```

Agora que o acesso ao banco de dados está configurado, vamos criar a página inicial do projeto. Para isso, vamos inicialmente remover o pacote `br.com.caelum.vraptor.controller` do projeto (Esse é um pacote que já vem criado dentro do blank-project. Depois de remover esse pacote, vamos criar o nosso controller para a página inicial do projeto, o `IndexController` dentro do pacote `br.com.alura.horas.controller` :

```
@Controller
public class IndexController {
    @Path("/")
    public void index(){}
}
```

Para finalizarmos só precisamos modificar o código da jsp da página inicial, o arquivo `WEB-INF/jsp/index/index.jsp` :

```
<!DOCTYPE html>
<html>
<head>
    <title>Controle de Horas</title>
</head>
```

```
<body>
    Bem vindo ao sistema de controle de horas!
</body>
</html>
```

Para testar a página inicial da aplicação, precisamos associar o projeto com o servidor Tomcat que configuramos anteriormente. Na aba `Servers`, clique com o botão direito no Tomcat 7 e selecione a opção `Add and Remove` e dentro da janela aberta, adicione o projeto `alura-horas` no Tomcat. Depois disso, inicie o servidor e entre na página `http://localhost:8080/alura-horas`