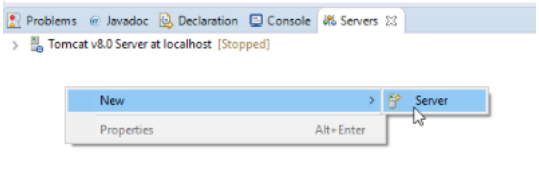
02

Mãos à obra: Colocando segundo tomcat localmente

Vimos que a configuração do Sticky Session resolver o problema que tínhamos e agora o balanceador é capaz de redirecionar as demais requisições para a mesma instância que possui a sessão do usuário. Porém, acabamos perdendo um pouco a funcionalidade do balanceamento. Para evitar isso, utilizaremos o banco de dados não relacional do Redis.

Vamos fazer o teste inicial localmente e depois levaremos a aplicação para a Amazon. Para isso, vá no Eclipse e na parte Servers clique com o botão direito do mouse e escolha a opção New -> Server.



Como temos um Tomcat já configurado, vamos alterar as portas utilizadas por esse segundo Tomcat para evitar assim conflitos com o primeiro Tomcat que possuimos. Para isso, clique duas vezes com o botão esquerdo do mouse no segundo Tomcat e troque as portas utilizadas:

Como vamos fazer esse teste inicialmente na nossa máquina local com o objetivo de verificar que tudo funciona, precisaríamos trocar a configuração do banco de dados para que utilizemos o usuário, senha e url locais, para evitar ficar trocando tais informações a todo momento, vamos utilizar o recurso de **profiles** do Spring. Para isso, vá até a classe **ServletSpringMVC** e sobrescreva o método **onStartup** e chame o método **setInitParameter** passando como argumento a chave **spring.profiles.active** e o valor **dev**

```
@Override
public void onStartup(ServletContext servletContext) throws ServletException {
    super.onStartup(servletContext);
    servletContext.setInitParameter("spring.profiles.active", "dev");
}
```

Com isso, vá a classe **JPAConfiguration** e configure dois métodos datasource um para o ambiente de desenvolvimento e outro para o ambiente de produção e coloque a anotação **@Profile** especificando os métodos com os respectivos ambientes os quais deverão ser utilizados:

```
    @Bean
    @Profile("dev")
    public DataSource dataSourceDev() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setUsername("root");
        dataSource.setPassword("");
        dataSource.setUrl("jdbc:mysql://localhost:3306/casadocodigo");
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");
```

```
        return dataSource;
    }

    @Bean
    @Profile("prod")
    public DataSource dataSourceProd() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setUsername("casadocodigo");
        dataSource.setPassword("casadocodigo");
        dataSource.setUrl("jdbc:mysql://[ENDPOINT BANCO AMAZON]:3306/casadocodigo");
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");
        return dataSource;
    }
```

Feito isso, iremos na próxima etapa realizar a configuração do Redis.