Websphere Liberty - Introdução

coleta de dados, avaliações e aceleradores de implementação

Última atualização: Setembro de 2024

Duração: 30 minutes

1. Introdução

O WebSphere Liberty (Liberty) é um framework aberto leve para construção de microserviços Java nativo e eficiente em nuvem. Construa apps nativas e microserviços nativos enquanto executa apenas o que você precisa. É o *runtime* mais flexível disponível para os desenvolvedores Java.

Liberty é construído no código aberto Open Liberty codebase.

- Entrega as APIs Java mais recentes e integra-se às ferramentas de Desenvolvedor e Construção mais populares.
- Possui inovação embutida como zero-migração para reduzir custos de tempo de execução e esforço de entrega.
- Liberty é downstream da Open Liberty para qualquer coisa que funcione no Open Liberty funciona também no Liberty.
- Mesmo Ciclo de Liberação de Entrega Contínua Mensal como Liberty
- Você não precisa mudar para o Liberty para suporte comercial

Liberty Tools é um conjunto de ferramentas de desenvolvedor intuitivas para os ambientes de desenvolvimento Eclipse IDE, Visual Studio Code e IntelliJ IDEA. Essas ferramentas adotam uma abordagem centrada em Maven / Gradle e possibilitam o desenvolvimento de aplicações Java nativas de nuvem rápida e iterativa por meio do modo Liberty dev. O Liberty Tools também fornece recursos úteis, de economia de tempo, como preenchimento de código, diagnósticos de configuração para APIs de Jakarta EE, APIs do MicroProfile e configuração Liberty.

2. Objetivos

Estes são os objetivos no laboratório:

- Como desenvolvedor de aplicativos:
    - Tarefas:
        - Desenvolver o aplicativo.
        - Cria uma configuração básica do Liberty para o aplicativo
    - Ferramentas:
        - Apache Maven
            - para construir o projeto
            - para fazer o download do tempo de execução do servidor Open Liberty a partir do repositório maven
        - Liberty Maven Plugin para desenvolvimento de loop interno via Liberty Dev Mode
            - para construir um aplicativo WAR e implantá-lo no Liberty
        - Código do Visual Studio
            - como IDE para construir o código de aplicação
        - Plugin do Liberty Tools para o Visual Studio Code
            - fornece um painel Liberty com integração Dev Mode na IDE
            - fornece assistência de código Jakarta EE e MicroProfile
            - fornece assistência de configuração Liberty
        - Projeto do Liberty Starter
            - para gerar um projeto maven para a Liberty
        - Abrir Liberty / WebSphere Liberty
            - como tempo de execução para o aplicativo Java que será desenvolvido
            - para criar um pacote de servidores

- Como operador de configuração Liberty:
    - Tarefas:
        - Extraia pacote Liberty da linha de comando e implementa atualizações dinâmicas da Liberty
        - Desenvolver trechos de configuração Liberty portáteis usando inclusos, variáveis e muito mais.
    - Ferramentas:
        - Código do Visual Studio com o plugin Liberty Tools como editor com assistência de configuração

- Como administrador do Liberty:
    - Tarefas:
        - Instala o Liberty
        - Configura Liberty para o aplicativo de destino usando trechos de configuração Liberty

        - Aplica a segurança para endurear a configuração do Liberty
        - Configurar o logging usando o configDropins
        - Revise a configuração usando o LIberty REST APIs
        - Monitore o Liberty usando o Centro Administrativo
    - Ferramentas:
        - Comando do Liberty server para criar uma instância Liberty e iniciar, parar ou lixá-lo
        - Liberty installUtility para instalar recursos ausentes
        - Liberty securityUtility para criar um armazenamento de chaves ou coditer uma senha
        - APIs REST do Liberty e Centro Administrativo

3. Executar Tarefas de Laboratório

3.1 Verifique o software instalado

1. Abra um terminal clicando em **Activities** e selecionando terminal.

A janela do terminal se abre.

2. Confira a versão do Maven através do seguinte comando:

```
mvn -version
```

3. Confira a versão do Docker através do seguinte comando:

```
docker -v
```

4. Confira a versão Git através do seguinte comando:

```
git -v
```

3.2 Criar os diretórios de trabalho necessários

1. Crie o diretório **Student** e alguns sub-diretórios utilizados no laboratório com os comandos abaixo:

```
mkdir ~/Student
mkdir ~/Student/dev
mkdir ~/Student/ops
mkdir ~/Student/assets
```

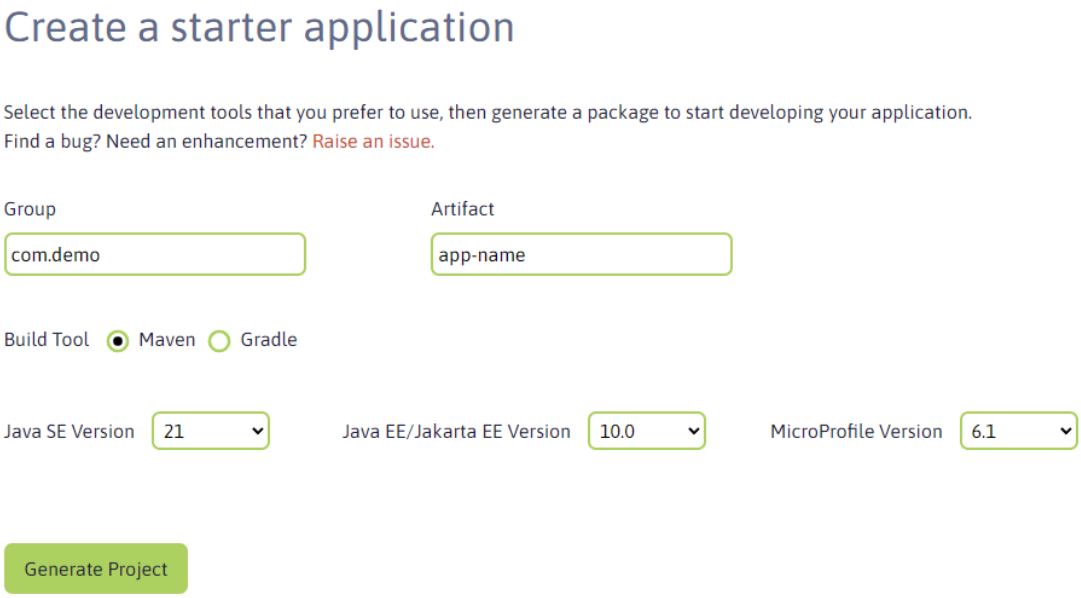
3.3 Desenvolva um aplicativo Web Liberty

O objetivo desta seção é desenvolver um aplicativo web simples para a Liberty. Você usará um aplicativo Liberty para começar do zero e usar o Visual Studio Code com o Liberty Tools para construir o aplicativo.

3.3.1 Criar um projeto de app iniciante.

Neste cenário, deseja-se criar um aplicativo Web Jakarta EE 10 com o nome **simpleweb** e usará maven para construí-lo. A maneira mais rápida de começar é utilizar um aplicativo

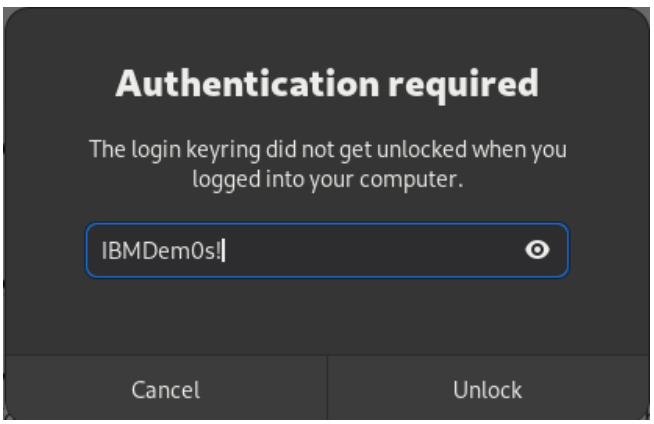
inaugural Open Liberty que gera um projeto com a configuração do maven, bem como um setup básico do Liberty.



O iniciador Open Liberty dá uma maneira simples e rápida de obter os arquivos necessários para começar a construir um aplicativo no Open Liberty. Não há necessidade de pesquisar como descobrir o que adicionar aos seus arquivos de construção maven ou gradle. Um arquivo RestApplication.java simples é gerado para você começar a criar um aplicativo baseado em REST. Um arquivo de configuração server.xml é fornecido com os recursos necessários para as versões MicroProfile e Jakarta EE que você selecionou anteriormente.

1. Abra uma janela do navegador clicando em **Activities** e, em seguida, selecione o ícone do navegador Firefox.

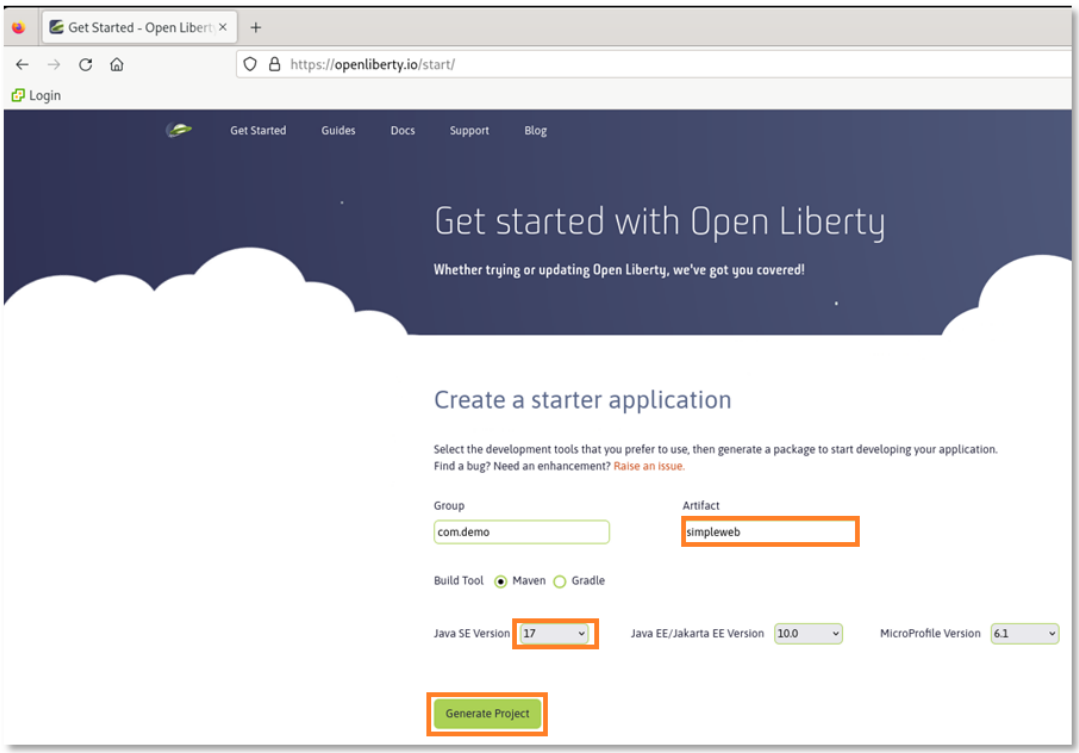
Se você obter um pop-up que a Autenticação é necessária, digite IBMDem0s!.



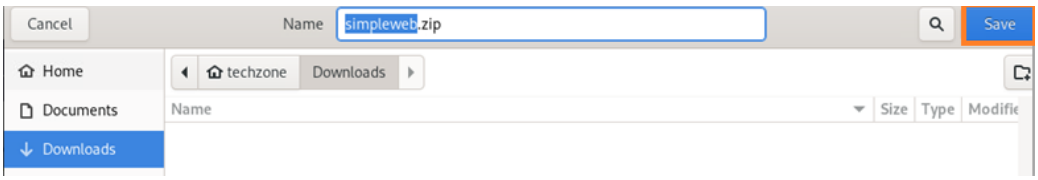
2. Insira a URL **https://openliberty.io/start/**

   Se a página não abrir, feche o navegador e abra-o novamente.

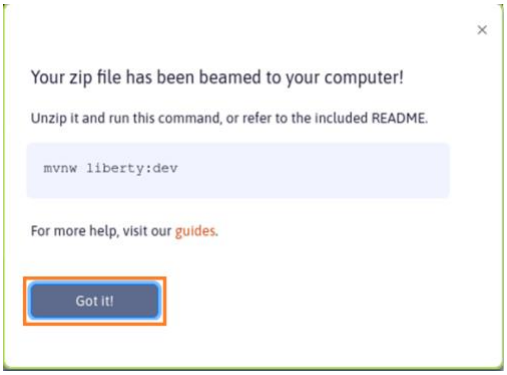
3. Altere o nome de artefato para **simpleweb**, altere o nível Java para **17** e deixe o restante como está, em seguida, clique no projeto Gerar



4. Clique em **Save** para salvar o projeto no diretório Downloads.



Você verá um pop-up como o abaixo. Clique em **Got it!** para fechar a janela.

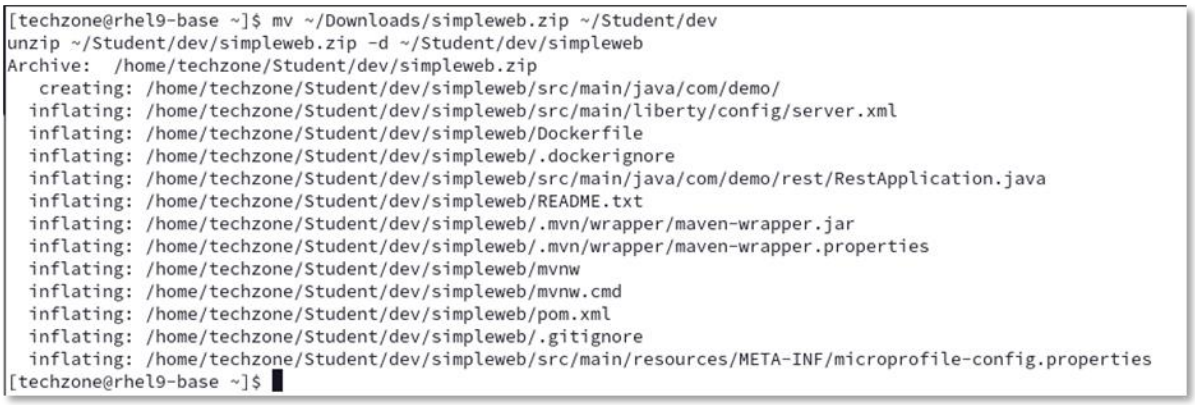


5. Extraia o arquivo.

   a. Clique em **Atividades** e alterne para a janela do terminal.

   b. Move o projeto para o diretório do desenvolvedor e extraia-o com os comandos a seguir:

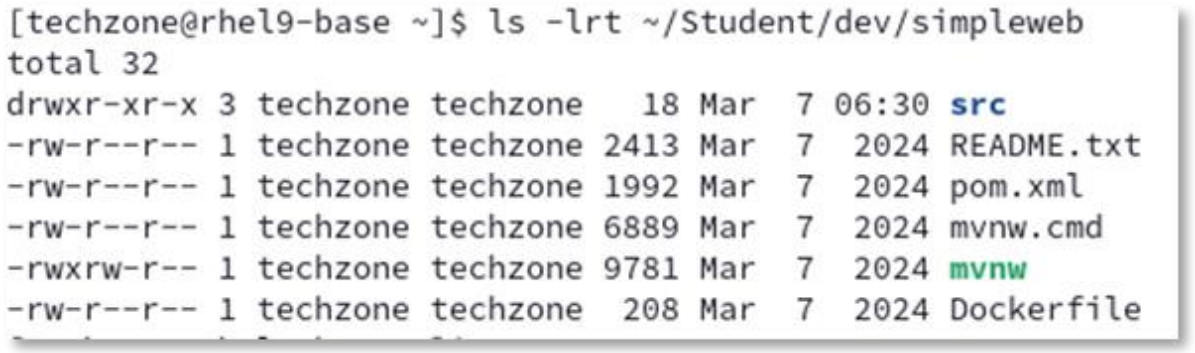
```
mv ~/Downloads/simpleweb.zip ~/Student/dev
unzip ~/Student/dev/simpleweb.zip -d ~/Student/dev/simpleweb
```



   O projeto foi criado sob o diretório ~/Student/dev/simpleweb.

   c. Liste o conteúdo via comando a seguir:

```
ls -lrt ~/Student/dev/simpleweb
```

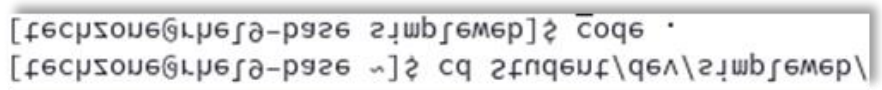


3.3.2. Inspecione o projeto inicial usando Visual Studio

Agora você usará o Visual Studio Code para ver o que foi gerado como parte do projeto iniciante.

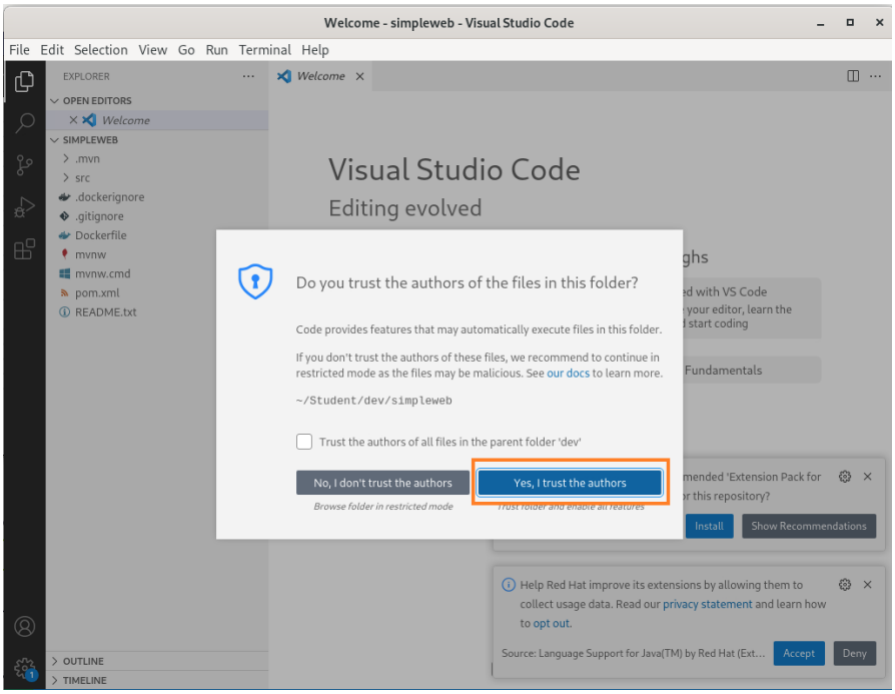
1. A partir da janela do terminal, inicie o Visual Studio Code

```
cd ~/Student/dev/simpleweb/
code .
```

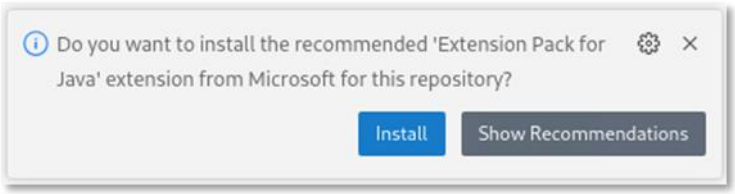


O Visual Studio Code UI será aberto.

2. Clique no **Yes, I trust the authors** para continuar.

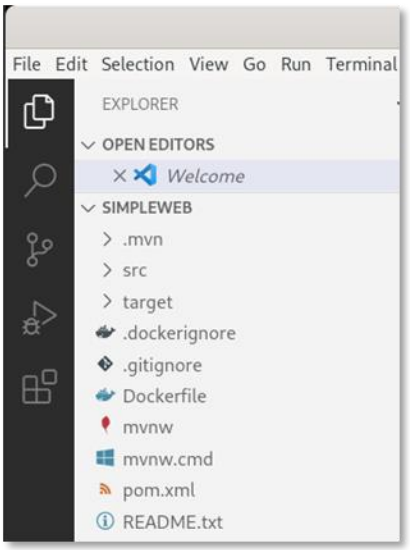


Se você visualizar durante o laboratório um dos pop-ups abaixo ou qualquer outro pop-up pedindo para instalar algo, feche o pop-up sem instalação clicando no X.



3. Investigar o projeto gerado:

   No Visual Studio Code, dê uma olhada na seção Explorer para ver o conteúdo do projeto. Você pode encontrar um **src** e uma pasta de destino, um **Dockerfile** e um arquivo de construção **maven** (pom.xml).



4. Dê uma olhada na configuração Maven gerada

   a. Clique no arquivo **pom.xml**.

   Na seção **build**, você pode encontrar a configuração do liberty-maven-plugin.

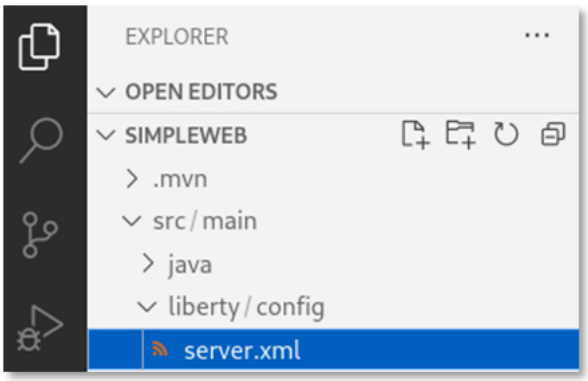
```
<build>
    <finalName>simpleweb</finalName>
    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.3.2</version>
            </plugin>
            <plugin>
                <groupId>io.openliberty.tools</groupId>
                <artifactId>liberty-maven-plugin</artifactId>
                <version>3.10.1</version>
            </plugin>
        </plugins>
    </pluginManagement>
    <plugins>
        <plugin>
            <groupId>io.openliberty.tools</groupId>
            <artifactId>liberty-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Não se preocupe se a versão do plugin mudou para 3.10.3 ou qualquer versão mais recente

b. feche o arquivo pom.xml.

5. Revise a configuração gerada para o Liberty

a. Abra **src> main> liberty> config> server.xml** para ver a configuração Liberty.



Como você pode ver, os recursos para jakartaee-10 e MicroProfile-6.1 foram configurados.

```
<!-- Enable features -->
<featureManager>
    <feature>jakartaee-10.0</feature>
    <feature>microProfile-6.1</feature>
</featureManager>
```

b. Scroll para baixo e você pode ver que o **http endpoint** e **web application** foram configurados.

```
<!-- To access this server from a remote client add a host attribute
<httpEndpoint id="defaultHttpEndpoint"
              httpPort="9080"
              httpsPort="9443" />

<!-- Automatically expand WAR files and EAR files -->
<applicationManager autoExpand="true"/>

<!-- Configures the application on a specified context root -->
<webApplication contextRoot="/simpleweb" location="simpleweb.war" />
```

3.3.3 Ajuste a configuração do Liberty

O aplicativo **simpleweb** não requer o padrão completo de Jakarta EE 10, mas apenas a especificação do servlet.

Como melhor prática para otimizar o *footprint* do tempo de execução do aplicativo em relação à memória e espaço em disco e limitar o número de vulnerabilidades em potencial, deve-se definir apenas os recursos que são requeridos pelo aplicativo. Neste caso, você está se irá substituir o recurso **jakartaee-10** por um recurso de servlet apropriado.

1. No editor do Visual Studio Code para server.xml, role até a seção de recursos.

2. Exclua as linhas <**feature>jakartaee-10.0</feature>** e <**feature> MicroProfile-6.1</feature>.** Sua seção featureManager deve agora ficar assim:

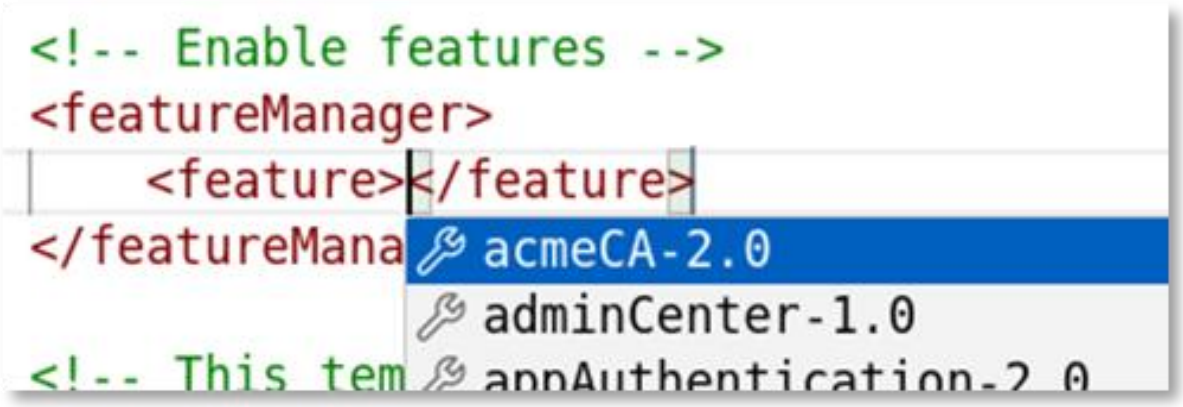
```
<!-- Enable features -->
<featureManager>

</featureManager>
```

3. Agora você usará o assistente de configuração do Liberty Tools para definir o recurso do servlet. Coloque o seu cursor no início de uma linha vazia na seção featureManager. Em seguida, pressione a tecla CTRL e pressione SPACE para ativar o assistente de configuração do Liberty Tools. Você deve ver algo como:

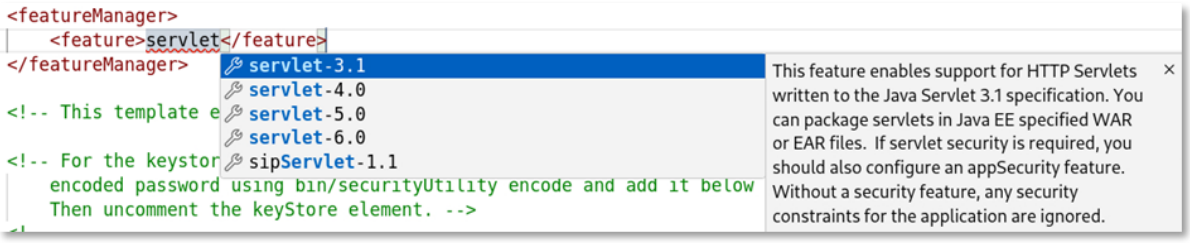
4. Selecione recurso e o elemento de recurso é adicionado.

```
<!-- Enable features -->
<featureManager>
    <feature></feature>
</featureManager>
```

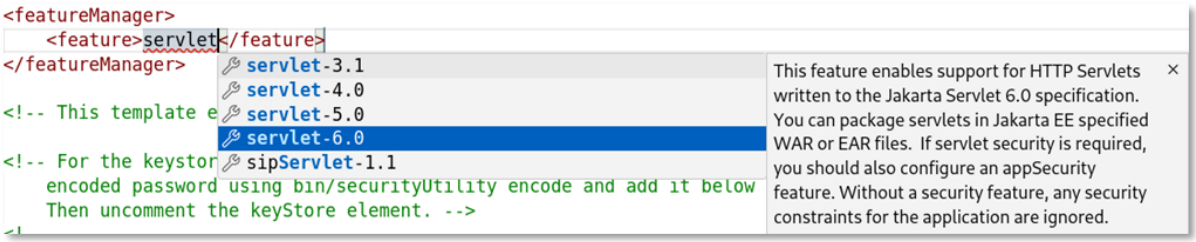
5. Use novamente CTRL + ESPAÇO para obter a lista de recursos disponíveis.



6. Digite a palavra servlet para ver os recursos do servlet disponíveis.



7. Use a tecla seta para baixo para obter a descrição para servlet-6.0.



8. Selecione o recurso servlet-6.0 e sua configuração deve agora ficar parecida com esta:

9. Para esta parte do laboratório, você não precisa definir um armazenamento de chaves ou o registro básico, portanto, é necessário excluir as entradas geradas. Sua configuração deve agora parecer assim:

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

    <!-- Enable features -->
    <featureManager>
        <feature>servlet-6.0</feature>
    </featureManager>


    <!-- To access this server from a remote client add a host attribute to the following element, e.g. host="*" -->
    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="9080"
                  httpsPort="9443" />

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

    <!-- Configures the application on a specified context root -->
    <webApplication contextRoot="/simpleweb" location="simpleweb.war" />

    <!-- Default SSL configuration enables trust for default certificates from the Java runtime -->
    <ssl id="defaultSSLConfig" trustDefaultCerts="true" />
</server>
```

Remova o trecho abaixo:

```
    <!--For a user registry configuration, configure your user regi
        basicRegistry element. Specify your own user name below in
        generate an encoded password using bin/securityUtility encc
        Then uncomment the user element. -->
    <basicRegistry id="basic" realm="BasicRealm">
        <!--
        <user name="yourUserName" password="" />
        -->
    </basicRegistry>
```

10. Salve a configuração usando CTRL + S.

11. Feche o arquivo server.xml.

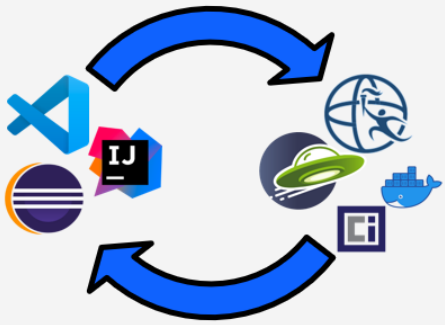
3.3.4 Usando o Liberty Dev Mode

O modo de desenvolvimento da Liberty, ou modo dev, permite desenvolver aplicativos com qualquer editor de texto ou IDE, fornecendo hot reload e implementação, sob teste de demanda e suporte de depurador. O Liberty Dev Mode é ativado através de projetos Maven e Gradle.

Seu código é compilado e implementado automaticamente para o seu servidor em execução, facilitando a iteração em suas alterações.

Você pode executar testes sob demanda ou até mesmo automaticamente para que você possa obter feedback imediato sobre suas alterações. Você também pode anexar um depurador a qualquer momento para depurar o seu aplicativo em execução.

Liberty dev mode



- Boosts developer productivity
- Immediate feedback for code and config changes
- No re-build necessary
- Hot deployment, testing and debugging
- Including in Containers

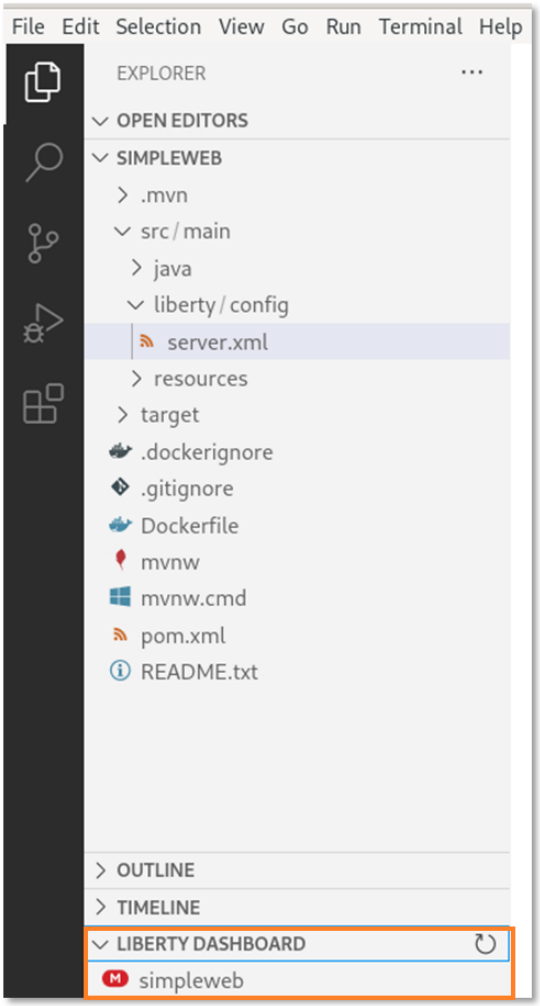
mvn liberty:dev *mvn liberty:devc*

gradle libertyDev *gradle libertyDevc*

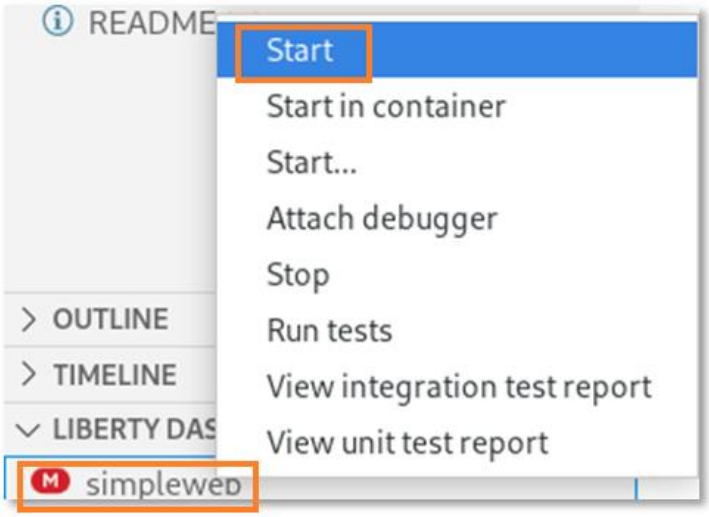
Você pode usar os recursos do modo Liberty dev dentro e fora de uma IDE. Isso lhe fornece a flexibilidade de escolha, assim você pode decidir qual IDE usar. Em uma janela de terminal, você usaria o Liberty no modo dev com o maven usando o comando mvn liberty: dev (ou mvn liberty: devc se quiser desenvolver em um container). Para gradle, os comandos relacionados são gradle libertyDev e gradle libertyDevc

Em ambiente de laboratório, o plugin Liberty tools foi instalado no Visual Studio Code. Portanto, você usará o painel do Liberty integrado.

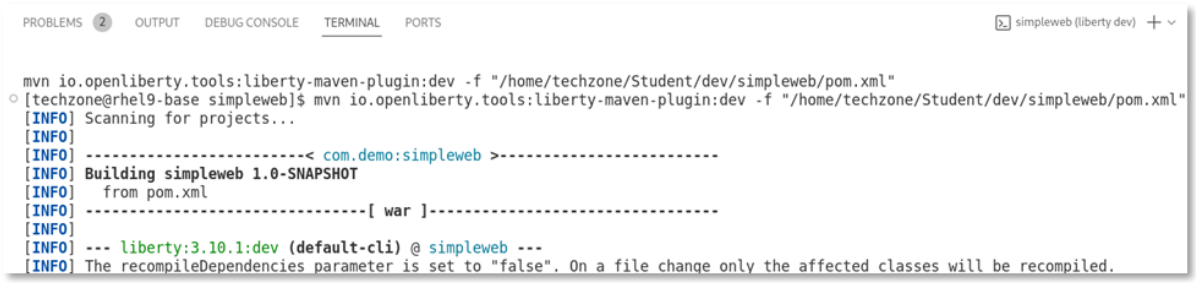
1. No Visual Studio Code, expanda o Liberty Dashboard.



2. Clique com o botão direito do mouse em **simpleweb** e depois inicie o servidor no **modo dev**.

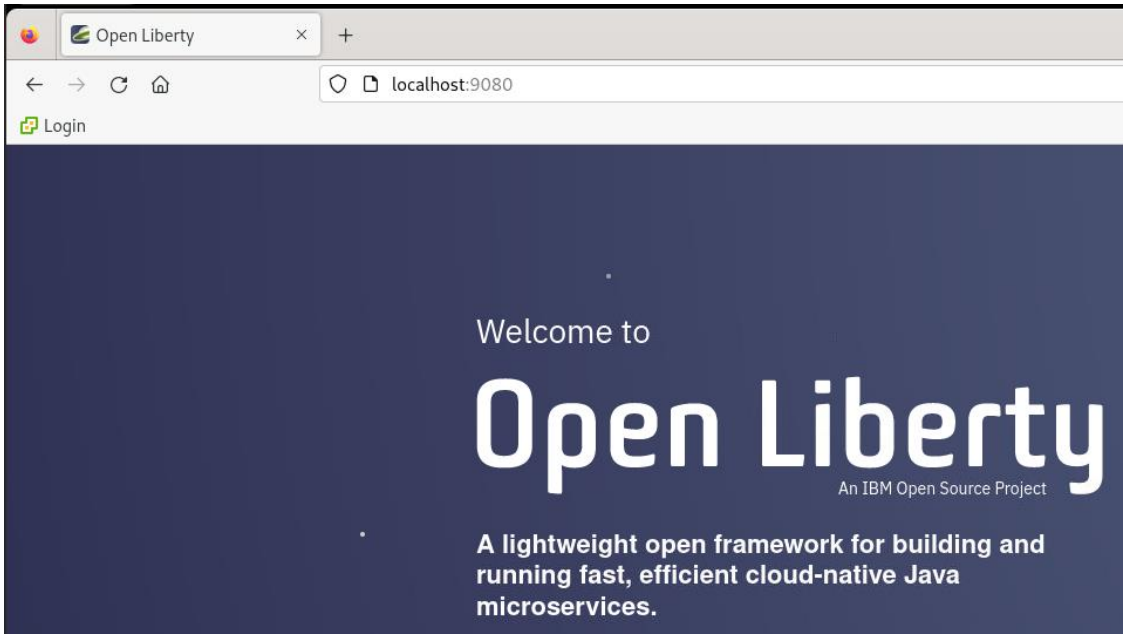


3. Um terminal se abre dentro do Visual Studio Code e você pode ver que o início do processo de construção é acionado.



4. O plugin Liberty, assim como os artefatos do servidor Liberty são baixados, então o servidor está pronto para testes.

5. Mude para a janela do navegador e digite a URL **localhost:9080.** Você deve ver algo assim:



Se você obter um pop-up com Autenticação necessária, digite a senha IBMDem0s! e clique em Desbloquear.

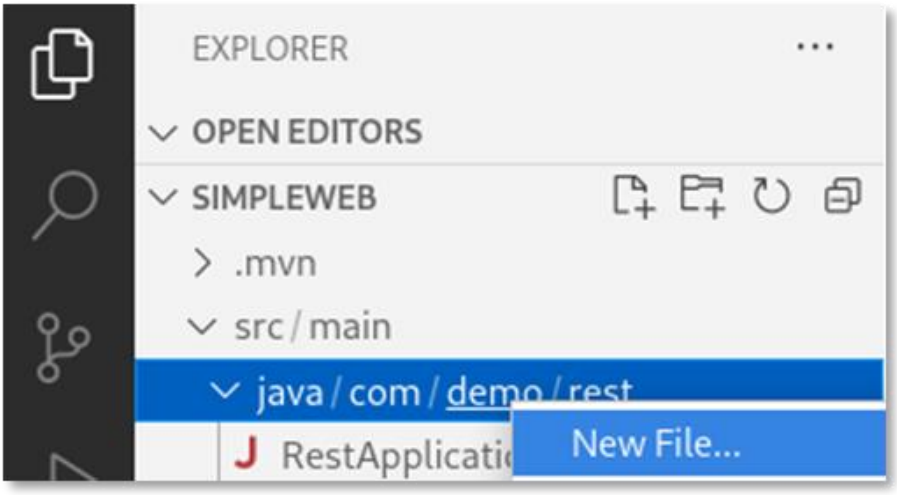
Agora, vamos criar um aplicativo web simples.

6.3.5 Editar o aplicativo simpleweb

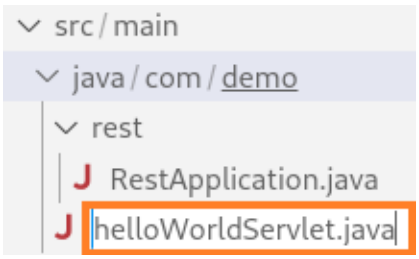
Agora você vai editar o aplicativo simpleweb que consiste apenas em um servlet. Graças ao assistente de código Liberty Tools, você não tem que escrever o código por conta própria.

1. Mude para o Visual Studio Code.

2. No Visual Studio Code, expanda o caminho para **src/main/java/com/demo/rest**, depois clique com o botão direito do mouse em demo e selecione Novo Arquivo.

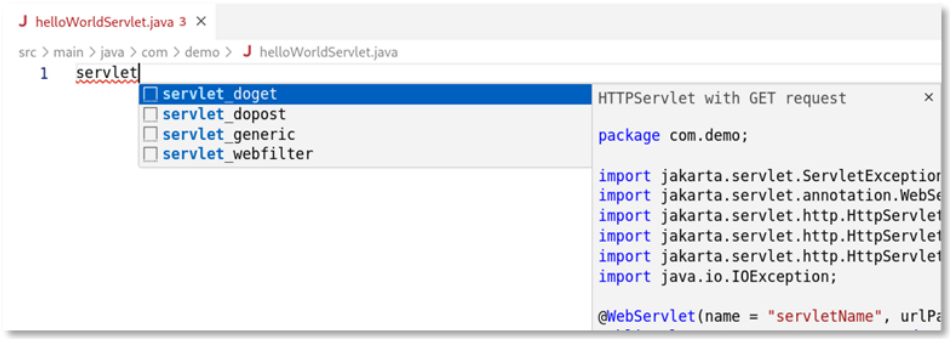


3. Digite o nome helloWorldServlet.java e pressione ENTER.



4. O arquivo **src/main/java/com/demo/helloWorldServlet.java** é gerado e abre em um editor.

5. Remova todos os códigos do arquivo. Em seguida, digite o servlet e pressione CTRL + ESPAÇO.



6. O assistente de código oferece alguns métodos de servlet para Jakarta EE. Selecione **servlet_doget,** e o código inicial necessário é gerado. Como você pode ver, os campos que devem ser alterados são destacados.

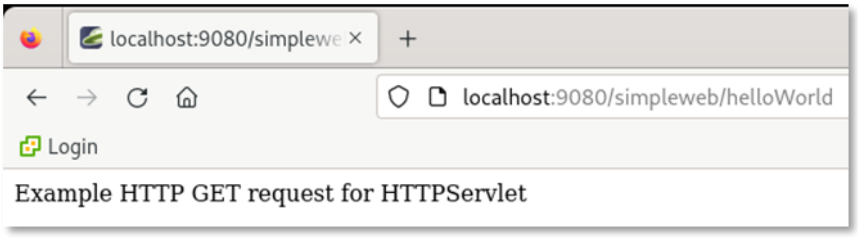
7. Altere o servletName para **helloWorldServlet** e os urlPatters para **/helloWorld**. O código deve agora ficar assim:

```
@WebServlet(name = 'helloWorldServlet', urlPatterns = { "/helloWorld" })
public class helloWorldServlet extends HttpServlet {
```

8. Pressione CTRL + S para salvar a alteração de código. Dê uma olhada na saída do terminal. Como o Liberty foi iniciado no DevMode, as alterações de código são captados automaticamente, o código fonte é compilado e o Liberty é atualizado.

```
[INFO] Source compilation was successful.
[INFO] [AUDIT   ] CWWKT0017I: Web application removed (default_host): http://localhost:9080/simpleweb/
[INFO] [AUDIT   ] CWWKZ0009I: The application simpleweb has stopped successfully.
[INFO] [AUDIT   ] CWWKT0016I: Web application available (default_host): http://localhost:9080/simpleweb/
[INFO] [AUDIT   ] CWWKZ0003I: The application simpleweb updated in 0.149 seconds.
```

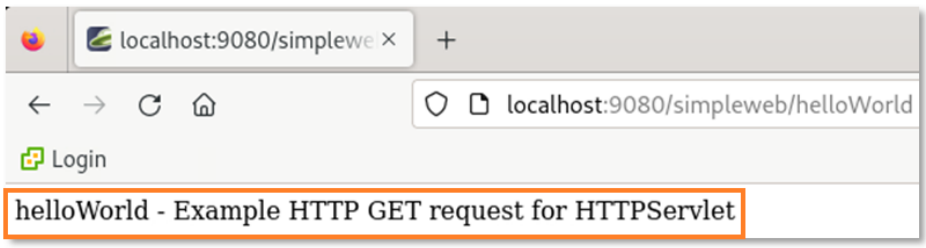
9. Mude para o navegador e abra a URL **localhost:9080/simpleweb/helloWorld.** Você deve ver a saída do servlet criado.



10. Mude de volta para o Visual Studio Code e altere o código-fonte do texto de resposta do servlet para algo como este: **helloWorld - Exemplo HTTP GET request for HTTPServlet**

```
@WebServlet(name = "helloWorldServlet", urlPatterns = { "/helloWorld" })
public class helloWorldServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        res.setContentType(type:"text/html;charset=UTF-8");
        res.getWriter().println(x:"helloWorld - Example HTTP GET request for HTTPServlet");
    }
```

11. Salve as alterações e refogue a página no navegador. A saída deve ser atualizada.



12. Mude de volta para o Visual Studio Code e feche o editor para o arquivo **helloWorldServlet.java**.

Parabéns !

Você concluiu com sucesso a implantação da aplicação no Liberty Profile