Aluno: Sérgio Luiz Teixeira Nunes Júnior

Aluno: Lucas Andrei Conceição do Sacramento

Documentação do Projeto

Descrição Geral:

Este sistema é uma aplicação Web que utiliza a arquitetura Servlet e integra várias bibliotecas populares para construção de aplicações Java robustas. O foco principal é gerenciar um conjunto de dados (como os de pacientes) e realizar operações de CRUD (criar, ler, atualizar e deletar). A aplicação faz uso de Maven para o gerenciamento de dependências e para facilitar a construção e empacotamento do projeto em formato WAR, que pode ser implantado em servidores como o Tomcat.

A solução é composta por diversos pacotes, cada um com responsabilidades específicas, como processamento de requisições HTTP, persistência de dados, e utilização de anotações customizadas para maior flexibilidade e organização.

Estrutura do Projeto

O projeto segue a estrutura padrão de um aplicativo Web em Java utilizando Maven:

- **src/main/java**: Contém as classes principais da aplicação, organizadas em pacotes.
  - **annotations**: Pacote para anotações customizadas que podem ser usadas para processar ou marcar classes e métodos.
  - **servlet**: Pacote onde as classes responsáveis pelo processamento de requisições HTTP (Servlettes) estão localizadas.
  - **model**: Contém as entidades de dados (por exemplo, Paciente), que são mapeadas para o banco de dados usando JPA (Java Persistence API).
  - **dao**: Contém as classes responsáveis pela manipulação de dados no banco (por exemplo, PacienteDAO).
- **src/main/resources**: Contém arquivos de configuração, como o web.xml, que é usado para definir os mapeamentos de Servlets e outras configurações relacionadas ao servidor.
- **pom.xml**: Arquivo de configuração do Maven, onde as dependências e plugins necessários são especificados. Este arquivo gerencia as bibliotecas externas e o empacotamento da aplicação.
- **src/test/java**: Contém os testes unitários da aplicação, garantindo que a funcionalidade do sistema esteja correta.

Componentes do Sistema

1. **Anotações Customizadas (Pacote annotations)**

- O pacote contém anotações personalizadas que são usadas para marcar métodos ou classes. Essas anotações podem ser processadas em tempo de execução para realizar ações específicas, como configurar comportamentos ou validar dados.

2. **Servlets (Pacote servlet)**

- O servlet principal (CentralServlet) gerencia as requisições HTTP e determina como responder a elas. Ele é mapeado para responder a qualquer URL, indicado pelo padrão /*. Esse servlet pode ser expandido para incluir mais lógica de negócios, como manipulação de dados de pacientes.

3. **Modelo de Dados (Pacote model)**

- Contém as entidades que representam os dados do sistema. A classe Paciente, por exemplo, é uma entidade que armazena informações sobre um paciente e está mapeada para um banco de dados relacional usando a API JPA.

4. **Acesso a Dados (Pacote dao)**

- Este pacote contém as classes responsáveis pela persistência dos dados. A classe PacienteDAO é usada para realizar operações de CRUD no banco de dados, utilizando a entidade Paciente. O uso do EntityManager permite realizar transações de forma simples e eficiente.

Funcionamento do Sistema

1. **Requisição e Resposta via Servlet**

- Quando um usuário acessa a aplicação através de um navegador, uma requisição HTTP é feita. O CentralServlet captura essa requisição e executa a lógica necessária, como retornar uma página HTML com informações ou processar dados enviados pelo usuário.
- As requisições são tratadas por métodos como doGet e doPost, e a resposta pode ser configurada para retornar HTML ou outro tipo de conteúdo.

2. **Persistência de Dados com JPA**

- As entidades do sistema (como Paciente) são mapeadas para tabelas no banco de dados usando JPA. A persistência é gerenciada por meio de um EntityManager, que é utilizado nas classes DAO (Data Access Object) para realizar operações de CRUD.

3. **Injeção de Dependência com Guice**

- O sistema utiliza o **Guice**, uma biblioteca de injeção de dependência, para criar instâncias de objetos de forma automática. Isso facilita a gestão das dependências e promove um código mais desacoplado e testável.

Como Funciona o Processo de Implantação

1. **Compilação e Construção**: O Maven é utilizado para compilar o código e construir o arquivo WAR (Web Archive), que é a forma empacotada da aplicação. O arquivo WAR contém todos os arquivos necessários para rodar a aplicação em um servidor web.
2. **Execução**: Após construir o WAR, ele pode ser implantado em um servidor de aplicação, como Tomcat. Uma vez em execução, a aplicação pode ser acessada pelos usuários via navegador.
3. **Testes**: Antes da implantação, os testes unitários devem ser executados para garantir que o sistema está funcionando conforme esperado. O JUnit é usado para testes automatizados das funcionalidades do sistema.

Testes

- O sistema inclui uma série de testes unitários que validam a funcionalidade do código. O principal objetivo é garantir que os Servlets, a persistência de dados e os componentes de injeção de dependência estejam funcionando corretamente.
- Os testes podem ser realizados utilizando o JUnit, que está configurado no pom.xml do projeto. Eles são executados durante o processo de build, assegurando que erros possam ser identificados logo no início.

Dependências e Tecnologias Utilizadas

1. **Servlet API**: Para criar e gerenciar Servlets.
2. **JPA (Java Persistence API)**: Para mapear e persistir entidades no banco de dados.
3. **HSQLDB**: Banco de dados em memória utilizado para persistir dados durante o desenvolvimento e testes.
4. **JUnit**: Framework para realizar testes unitários.
5. **Guice**: Framework para injeção de dependência.
6. **Reflections**: Para explorar o código e facilitar a busca de classes anotadas.