README TEMPLATE

README - Progetto Web Java con JSP, JSTL e Maven

Fabio Camarlinghi 5BI 04/03/2025

Descrizione del Progetto

Questo progetto è un'applicazione web sviluppata in Java utilizzando **JSP** (JavaServer Pages), **JSTL** (JavaServer Pages Standard Tag Library) e **Maven** come strumento di gestione delle dipendenze e di build. L'applicazione è un semplice esempio di gestione di una pagina web che visualizza informazioni sull'autore e un'area per visualizzare la data corrente..

Struttura del Progetto

Il progetto è organizzato nei seguenti file principali:

- **index.jsp**: La pagina di benvenuto che mostra la data corrente e un link verso una pagina che contiene informazioni sull'autore.
- **author.jsp**: La pagina che mostra informazioni sull'autore, inclusi il nome, cognome, classe e la data di visita.
- **pom.xml**: Il file di configurazione di Maven, che gestisce le dipendenze e la configurazione di build.

Componenti e Librerie Utilizzate

1. JSP (JavaServer Pages)

JSP è utilizzato per generare dinamicamente contenuti HTML. Le pagine JSP sono una combinazione di HTML e codice Java che viene eseguito sul lato server per produrre la pagina finale. Le seguenti pagine JSP sono utilizzate nel progetto:

- **index.jsp**: Visualizza un messaggio di benvenuto e la data corrente.

- **author.jsp**: Mostra le informazioni sull'autore (nome, cognome, classe) e la data di visita.

2. JSTL (JavaServer Pages Standard Tag Library)

JSTL è una libreria di tag standard per JSP che aiuta a evitare l'uso codice Java direttamente all'interno delle pagine JSP. In questo progetto, JSTL viene utilizzato per formattare la data.

- **Tag** `<fmt:formatDate>`: Utilizzato per formattare la data in formato `dd/MM/yyyy`. La data viene passata a JSP tramite un attributo e formattata correttamente.

3. Maven

Maven è utilizzato come strumento di gestione del ciclo di vita del progetto, delle dipendenze e della build. Il file `pom.xml` contiene tutte le dipendenze necessarie e configura le impostazioni di compilazione per il progetto.

- **Packaging WAR**: Il progetto è configurato per essere impacchettato come un file WAR (Web Application Archive) .
- **Dipendenze principali**:
  - **Servlet API**: Per la gestione delle servlet e la creazione di applicazioni web.
  - **Jersey**: Per il supporto alle API RESTful.
  - **JUnit**: Per i test unitari del progetto.
  - **JSTL**: Per il supporto ai tag standard JSP, come `<fmt:formatDate>`.

4. Jetty Maven Plugin

Jetty è un server HTTP che viene utilizzato per il testing in ambiente di sviluppo. Il plugin Maven `jetty-maven-plugin` consente di avviare il server Jetty per testare l'applicazione durante lo sviluppo.

5. Servlet API

La **Servlet API** (con la versione `4.0.1`) è utilizzata per la gestione delle servlet in Java. Questa API permette di gestire le richieste HTTP in modo strutturato.

6. JUnit

JUnit è un framework di testing utilizzato per eseguire test unitari sui componenti Java. La versione `4.13.2` è utilizzata in questo progetto per testare il codice Java.

File principali

- `index.jsp`: La pagina principale che mostra il benvenuto e la data.
- `author.jsp`: La pagina con le informazioni sull'autore.
- `pom.xml`: Il file di configurazione di Maven.

COMANDI UTILIZZATI:

Durante lo sviluppo del progetto ho utilizzato, o tramite la console oppure attraverso delle configurazioni di run di Eclipse, alcuni comandi per l'avvio del server e della compilazione e creazione del file in base alle direttive del file pom.xml.

- `jetty:run` : E' utilizzato per avviare un'applicazione web in un contenitore **Jetty** direttamente dal terminale, mentre lavori in un ambiente di sviluppo.
  Nel contesto di Maven, il comando
  `jetty:run` è legato al plugin **Jetty Maven Plugin**, che permette di eseguire un'applicazione web senza bisogno di un server esterno

- `mvn package` : E' usato per **compilare** il progetto e **creare il pacchetto** (come un file `.jar`, `.war`, `.ear`, ecc.) in base alla configurazione definita nel `pom.xml` del progetto.

PROBLEMATICHE RISCONTRATE

All'interno della cartella del mio progetto "library", non è possibile trovare il percorso del package `it.html.tutorial.library.api"` come indicato dal tutorial e mi era impossibile crearlo.

Per questo non mi è stato possibile implementare la parte del tutorial riguardante l'API.