Creazione di un sistema di logging delle visite di un sito web con Java Spring Boot

Il monitoraggio delle visite di un sito web è fondamentale per comprendere il comportamento degli utenti e raccogliere dati per ottimizzare il servizio offerto. In questo articolo vedremo come creare un sistema di logging delle visite di un sito web utilizzando Java con il framework Spring Boot. Esploreremo come registrare dettagli sulle visite, come l'indirizzo IP, la data e ora della visita, l'URL visitato, e altri parametri che possono fornire insight sull'interazione dell'utente con il sito.

Iniziamo creando un progetto Spring Boot. Puoi farlo tramite Spring Initializr selezionando le seguenti dipendenze:

- **Spring Web**: per creare un'applicazione web RESTful.
- **Spring Data JPA**: per l'interazione con il database.
- **MySQL Driver**: per connetterci a un database MySQL.

Nella directory principale del progetto, apri il file `application.properties` o `application.yml` (a seconda del formato che preferisci). Configura le impostazioni di connessione al database:

```
# application.properties
spring.datasource.url=jdbc:mysql://localhost:3306/nome_database
spring.datasource.username=username
spring.datasource.password=password
```

```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Assicurati di sostituire `nome_database`, `username`, e `password` con le informazioni corrette del tuo database MySQL.

Per registrare le visite, creiamo una classe `VisitLog` che rappresenta la tabella del database in cui registrare le informazioni. I campi includeranno l'indirizzo IP dell'utente, l'URL visitato, la data e l'ora della visita.

```
import javax.persistence.*;
import java.time.LocalDateTime;

@Entity
public class VisitLog {

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

    private String ipAddress;
    private String url;
    private LocalDateTime timestamp;

    public VisitLog() {
    }

    public VisitLog(String ipAddress, String url,
LocalDateTime timestamp) {
        this.ipAddress = ipAddress;
        this.url = url;
```

```
        this.timestamp = timestamp;
    }

    // Getter e Setter
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getIpAddress() {
        return ipAddress;
    }

    public void setIpAddress(String ipAddress) {
        this.ipAddress = ipAddress;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public LocalDateTime getTimestamp() {
        return timestamp;
    }

    public void setTimestamp(LocalDateTime timestamp)
{
```

```
        this.timestamp = timestamp;
    }
}
```

Ora, creiamo un’interfaccia `VisitLogRepository` che estende `JpaRepository`. Questa ci permetterà di interagire con il database tramite operazioni CRUD.

```
import
org.springframework.data.jpa.repository.JpaRepository
;

public interface VisitLogRepository extends
JpaRepository<VisitLog, Long> {
}
```

Creiamo una classe `VisitLogService` che contiene la logica per salvare i dati delle visite. Questa classe userà il `VisitLogRepository` per salvare ogni visita.

```
import
org.springframework.beans.factory.annotation.Autowire
d;
import org.springframework.stereotype.Service;
import java.time.LocalDateTime;

@Service
public class VisitLogService {
```

```
    @Autowired
    private VisitLogRepository visitLogRepository;

    public void logVisit(String ipAddress, String
url) {
        VisitLog visitLog = new VisitLog(ipAddress,
url, LocalDateTime.now());
        visitLogRepository.save(visitLog);
    }
}
```

Gli interceptor di Spring ci permettono di intercettare ogni richiesta HTTP in arrivo. Creiamo un `VisitLogInterceptor` che si occuperà di registrare ogni visita.

```
import
org.springframework.beans.factory.annotation.Autowire
d;
import org.springframework.stereotype.Component;
import
org.springframework.web.servlet.HandlerInterceptor;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@Component
public class VisitLogInterceptor implements
HandlerInterceptor {

    @Autowired
    private VisitLogService visitLogService;

    @Override
```

```
    public boolean preHandle(HttpServletRequest
request, HttpServletResponse response, Object
handler) throws Exception {
        String ipAddress = request.getRemoteAddr();
        String url =
request.getRequestURL().toString();
        visitLogService.logVisit(ipAddress, url);
        return true;
    }
}
```

Per far sì che l’interceptor venga applicato a tutte le richieste, configurarlo in una classe `WebConfig` che implementa `WebMvcConfigurer`.

```
import
org.springframework.beans.factory.annotation.Autowire
d;
import
org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.Int
erceptorRegistry;
import
org.springframework.web.servlet.config.annotation.Web
MvcConfigurer;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Autowired
    private VisitLogInterceptor visitLogInterceptor;
```

```
    @Override
    public void addInterceptors(InterceptorRegistry
registry) {
        registry.addInterceptor(visitLogInterceptor);
    }
}
```

Conclusione

In questo tutorial, abbiamo creato un semplice sistema di logging delle visite di un sito web utilizzando Java e Spring Boot. Grazie a Spring Interceptor, siamo stati in grado di registrare automaticamente ogni visita senza aggiungere codice ad ogni controller dell’applicazione, rendendo il sistema scalabile e facilmente gestibile. Questo tipo di sistema è molto utile per monitorare l’andamento delle visite e per migliorare l’esperienza degli utenti sul sito web.