```
public class PageVisitEntry {
    private String url;
    private long timestamp;

    public PageVisitEntry(String url, long timestamp) {
        this.url = url;
        this.timestamp = timestamp;
    }

    public String getUrl() {
        return url;
    }

    public long getTimestamp() {
        return timestamp;
    }
}
```

```
import jakarta.annotation.PostConstruct;
import jakarta.annotation.PreDestroy;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.HandlerInterceptor;

import java.util.*;
import java.util.concurrent.*;

@Component
public class VisitInterceptor implements HandlerInterceptor {

    // Buffer for incoming page requests per session
    private final ConcurrentHashMap<String, List<PageVisitEntry>> visitBuffer = new
ConcurrentHashMap<>();

    // Scheduler to process and persist visits
    private final ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();

    private static final long VISIT_WINDOW_MS = 4000; // 4 seconds window to group requests

    @PostConstruct
    public void startScheduler() {
        scheduler.scheduleAtFixedRate(this::processBufferedVisits, 5, 5, TimeUnit.SECONDS);
    }

    @PreDestroy
    public void shutdownScheduler() {
        scheduler.shutdownNow();
    }

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
Object handler) {
        String sessionId = request.getSession().getId();
        String url = request.getRequestURI();
        long timestamp = System.currentTimeMillis();

        PageVisitEntry entry = new PageVisitEntry(url, timestamp);

        visitBuffer.computeIfAbsent(sessionId, k -> new CopyOnWriteArrayList<>()).add(entry);

        return true;
    }

    private void processBufferedVisits() {
        long now = System.currentTimeMillis();

        for (Map.Entry<String, List<PageVisitEntry>> entry : visitBuffer.entrySet()) {
            String sessionId = entry.getKey();
            List<PageVisitEntry> requests = entry.getValue();

            List<PageVisitEntry> toProcess = new ArrayList<>();

            // Collect entries within the visit window
            Iterator<PageVisitEntry> iterator = requests.iterator();
            while (iterator.hasNext()) {
                PageVisitEntry visit = iterator.next();
                if (now - visit.getTimestamp() <= VISIT_WINDOW_MS) {
                    toProcess.add(visit);
                    iterator.remove(); // Remove processed
                }
            }

            if (!toProcess.isEmpty()) {
                // Filter the main request (skip .js, .css, API calls etc.)
                Optional<PageVisitEntry> mainVisit = toProcess.stream()
                    .filter(v -> isPrimaryVisit(v.getUrl()))
                    .findFirst();

                mainVisit.ifPresent(visit -> persistVisit(sessionId, visit));
            }
        }
    }

    private boolean isPrimaryVisit(String url) {
        return !(url.endsWith(".js") || url.endsWith(".css") || url.endsWith(".png") ||
                 url.endsWith(".jpg") || url.contains("/api/") || url.contains("/static/"));
    }

    private void persistVisit(String sessionId, PageVisitEntry visit) {
        // You can replace this with DB logic using a repository/service
        System.out.println("Persisting visit for session: " + sessionId +
                ", url: " + visit.getUrl() +
                ", time: " + new Date(visit.getTimestamp()));
    }
}


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Autowired
    private VisitInterceptor visitInterceptor;

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