



Asynchrone Requestverarbeitung

Java EE Grundlagen

Inhalte dieses Kapitels

Allgemeines

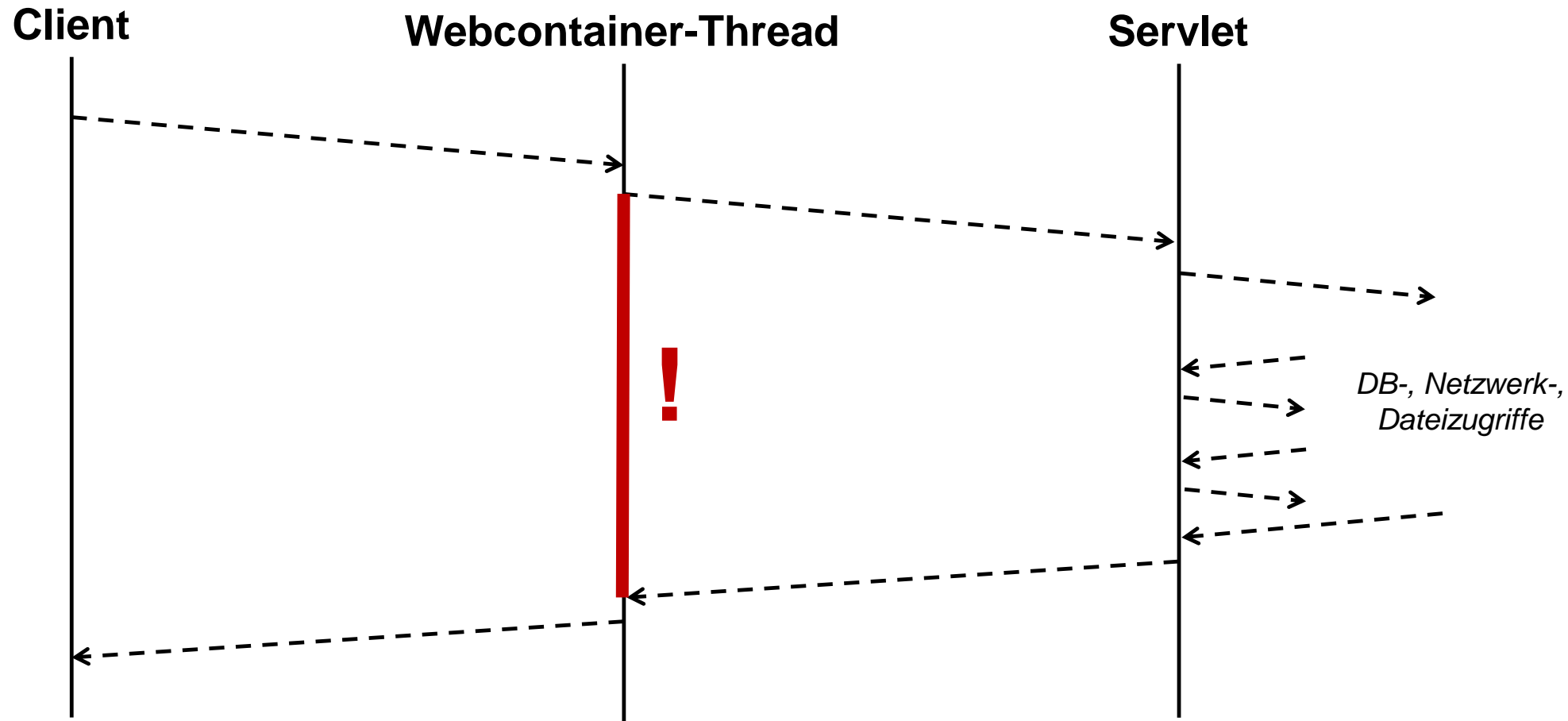
Umsetzung in der Servlet API

Zusammenfassung

Allgemeines [1|2]

Asynchrone Requestverarbeitung

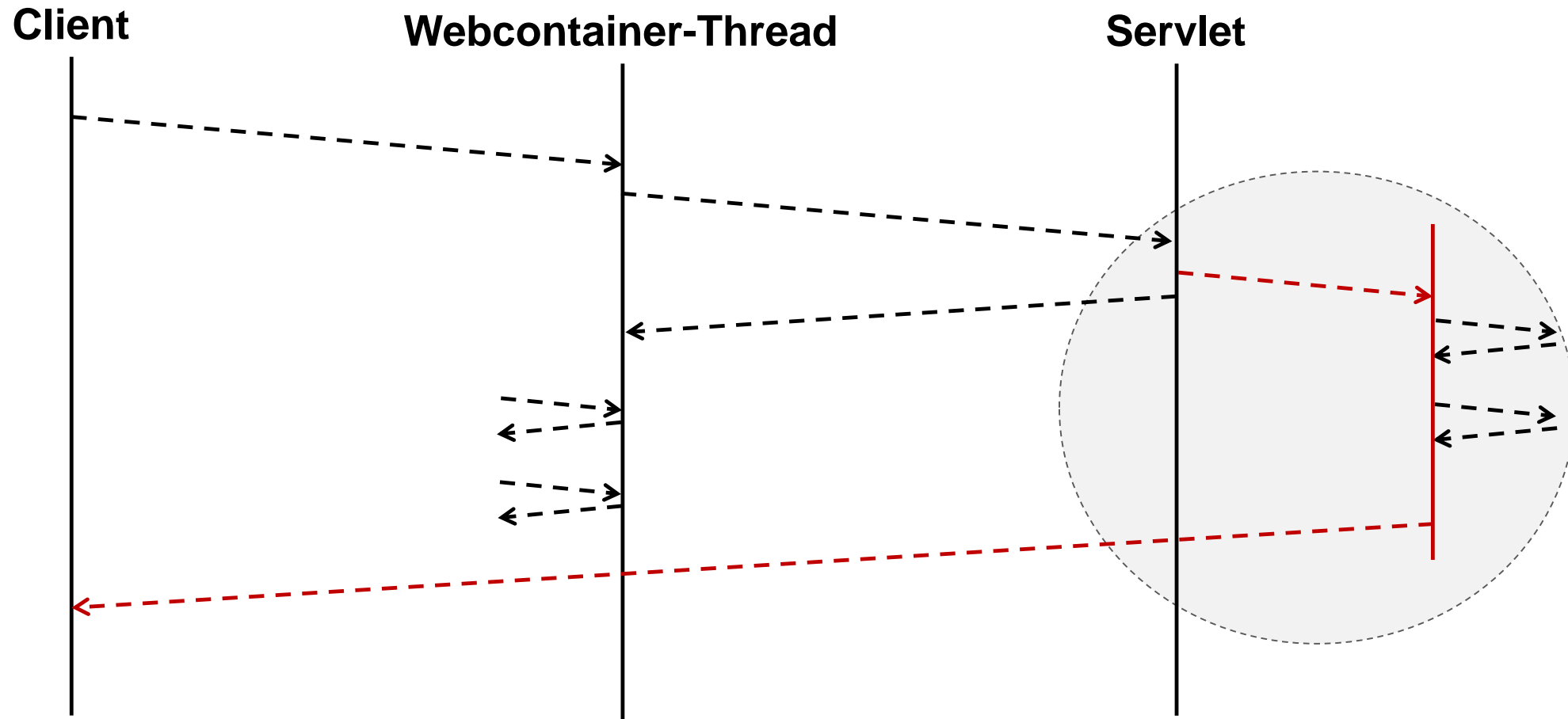
- Problemstellung bei synchroner Verarbeitung (serverseitig)



Allgemeines [2|2]

Asynchrone Requestverarbeitung

- Lösungsansatz



Umsetzung in der Servlet API [1|4]

Asynchrone Requestverarbeitung

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // Anforderung asynchroner Verarbeitung (optional: mit Request/Response-Wrapper als Parameter)
    final AsyncContext async = request.startAsync();
    // Starten der parallelen Verarbeitung
    async.start(new Runnable() {

        @Override
        public void run() {
            ServletResponse response = async.getResponse();
            ServletRequest request = async.getRequest();
            // Requestverarbeitung ...
            async.complete();
        }
    });
}
```

```
@WebServlet(..., asyncSupported=true)
public class AjaxServlet extends HttpServlet {...}
```

Umsetzung in der Servlet API [2|4]

Asynchrone Requestverarbeitung

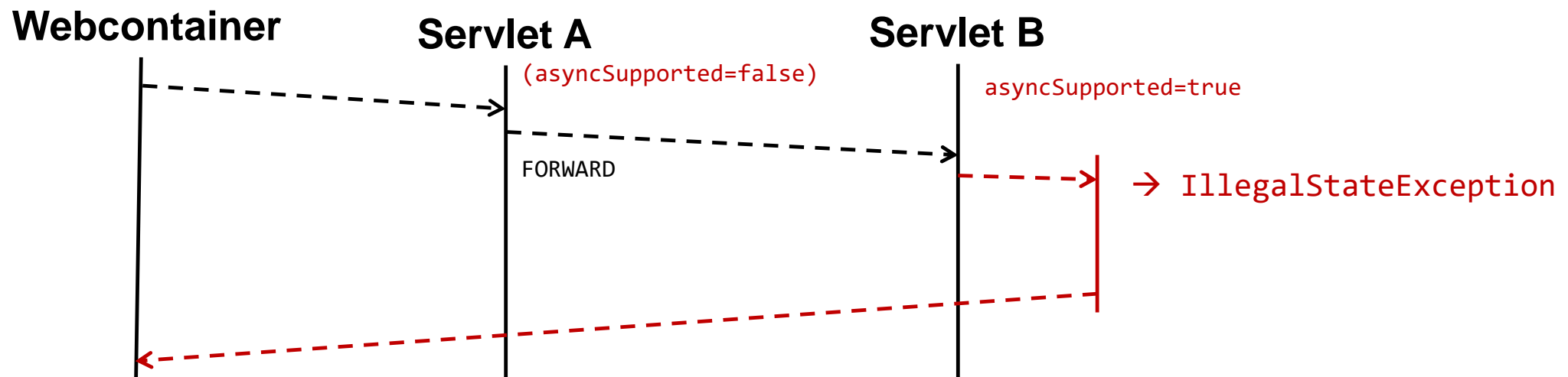
■ Weitere Optionen

```
final AsyncContext async = request.startAsync();  
// Timeout in ms (0: kein Timeout, Standardwert durch Container belegt !!!)  
async.setTimeout(TimeUnit.MINUTES.toMillis(2));  
// Event Handling (Callbacks)  
async.addListener(new AsyncListener() {  
  
    public void onStartAsync(AsyncEvent event) throws IOException { /* ... */}  
  
    public void onComplete(AsyncEvent event) throws IOException { /* ... */}  
  
    public void onTimeout(AsyncEvent event) throws IOException { /* ... */}  
  
    public void onError(AsyncEvent event) throws IOException { /* ... */}  
  
});  
// Starten der parallelen Verarbeitung  
async.start(...);
```

Umsetzung in der Servlet API [3|4]

Asynchrone Requestverarbeitung

- Motivation: Freigabe des Webcontainer-Threads
 - Bei inperformanten Operationen
 - Bei dauerhaft offenen Verbindungen (HTTP 1.1, „Comet“)
- Problemstellung: Servlet-Verkettungen



Umsetzung in der Servlet API [4|4]

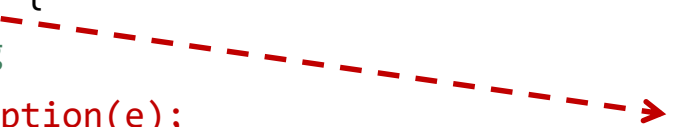
Asynchrone Requestverarbeitung

■ Problemstellung: Exception Handling

!!! Keine (deklarativen) Error Pages !!!!

```
async.start(new Runnable() {  
  
    @Override  
    public void run() { !!! Keine Checked Exceptions erlaubt !!!  
        try {  
            // Requestverarbeitung ...  
            PrintWriter out = response.getWriter();  
            // Antwortgenerierung ...  
        } catch (IOException e) {  
            // Ausnahmebehandlung  
            throw new RuntimeException(e);  
        }  
        async.complete();  
    }  
});
```

```
async.addListener(new AsyncListener() {  
  
    @Override  
    public void onError(AsyncEvent event)  
        throws IOException {  
        /* ... */  
    }  
});
```



Zusammenfassung

Weitere Informationen

- Servlet API als Basis weiterer Technologien
 - JAX-RS (REST Services)
 - Asynchrone Services
 - Asynchrone Clients
- Servlet API 3.1
 - Non-blocking I/O
- Integration der Concurrency Utilities in Java EE
 - Managed Thread Factory
 - Managed (Scheduled) Executor Services
 - Context Service