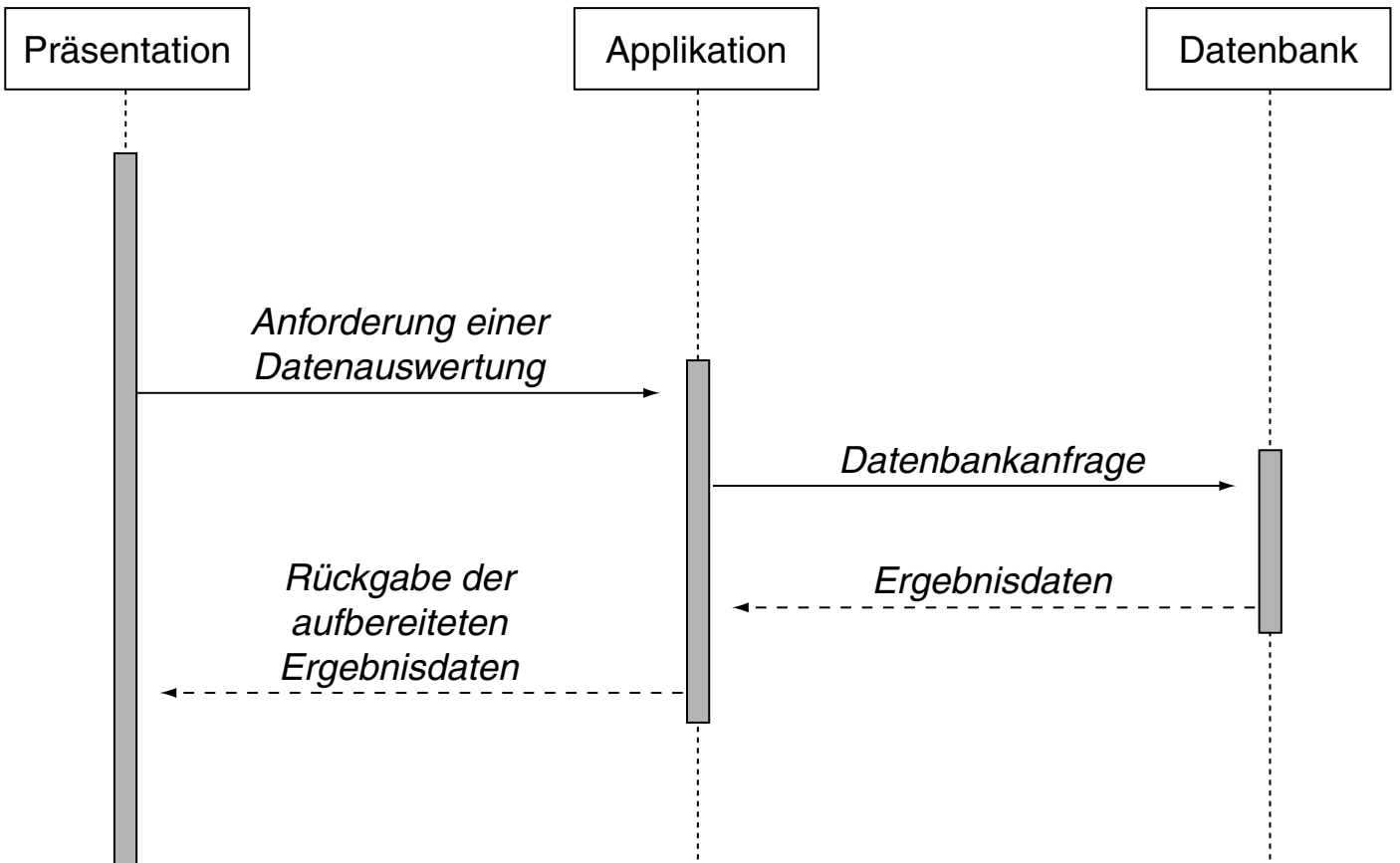
Kapitel WT:VI

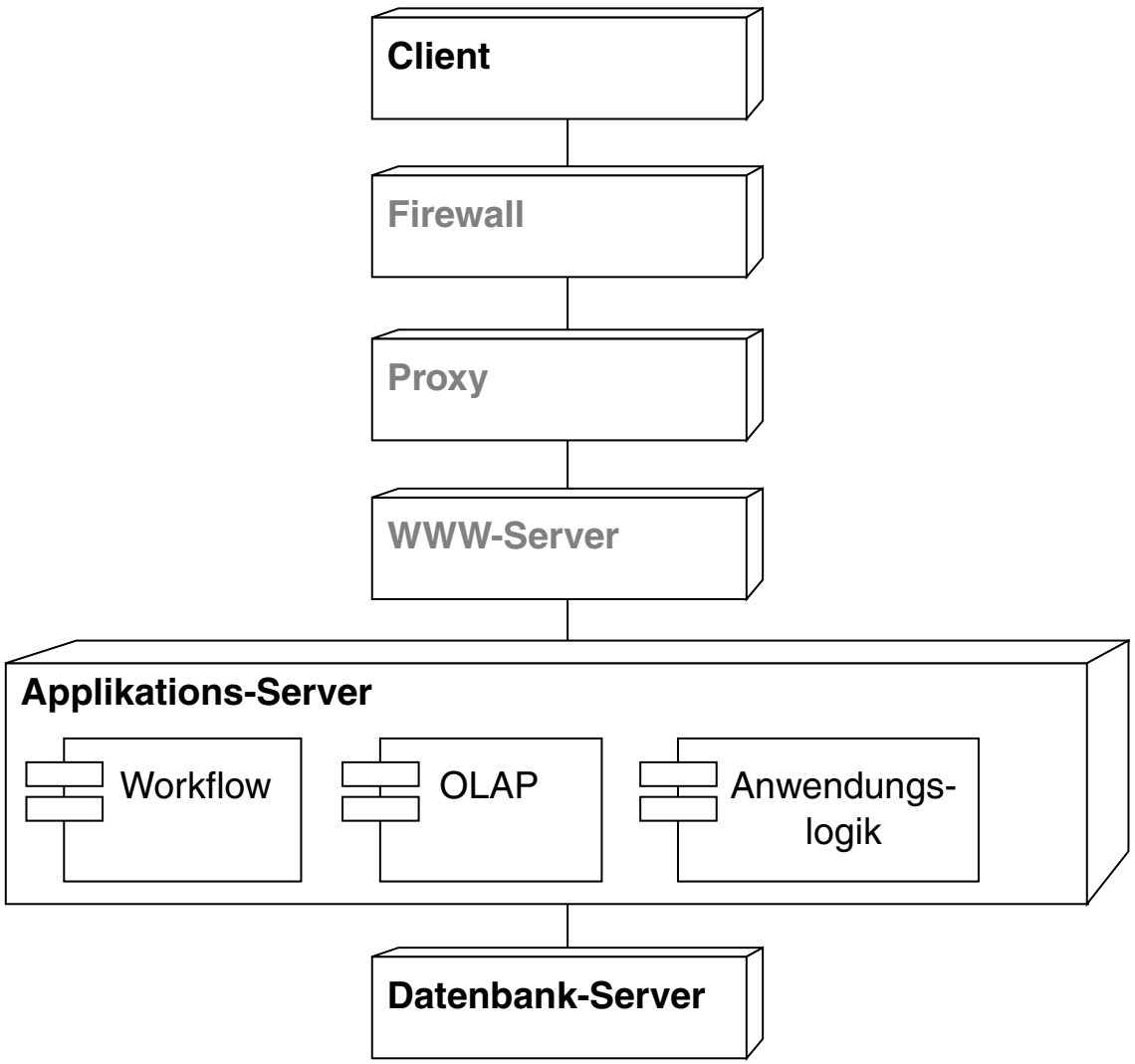
Client-Server-Architekturen

3-Tier Architektur: Sequenzdiagramm



Client-Server-Architekturen

3-Tier Architektur: Deployment-Diagramm



Client-Server-Architekturen

Architekturmuster	Applikationslogik	Präsentationsschicht
Thin Client	im Applikationsserver, z.B. JSP + JavaBeans mit Tomcat, J2EE-EnterpriseJavaBeans	HTML + JavaScript im Webbrowser

Client-Server-Architekturen

Architekturmuster	Applikationslogik	Präsentationsschicht
Thin Client	im Applikationsserver, z.B. JSP + JavaBeans mit Tomcat, J2EE-EnterpriseJavaBeans	HTML + JavaScript im Webbrowser
Portal	im Portal-Applikationsserver	Portlets im Browser

Client-Server-Architekturen

Architekturmuster	Applikationslogik	Präsentationsschicht
Thin Client	im Applikationsserver, z.B. JSP + JavaBeans mit Tomcat, J2EE-EnterpriseJavaBeans	HTML + JavaScript im Webbrowser
Portal	im Portal-Applikationsserver	Portlets im Browser
Rich Thin Client	sowohl im Client als auch im Applikationsserver	Java-AWT-GUI in speziellem Java Applet im Webbrowser

Client-Server-Architekturen

Architekturmuster	Applikationslogik	Präsentationsschicht
Thin Client	im Applikationsserver, z.B. JSP + JavaBeans mit Tomcat, J2EE-EnterpriseJavaBeans	HTML + JavaScript im Webbrowser
Portal	im Portal-Applikationsserver	Portlets im Browser
Rich Thin Client	sowohl im Client als auch im Applikationsserver	Java-AWT-GUI in speziellem Java Applet im Webbrowser
Rich Fat Client	clientseitiges Framework (plattformabhängig)	z.B. Java-SWT, JFace-GUI

Client-Server-Architekturen

Architekturmuster	Applikationslogik	Präsentationsschicht
Thin Client	im Applikationsserver, z.B. JSP + JavaBeans mit Tomcat, J2EE-EnterpriseJavaBeans	HTML + JavaScript im Webbrowser
Portal	im Portal-Applikationsserver	Portlets im Browser
Rich Thin Client	sowohl im Client als auch im Applikationsserver	Java-AWT-GUI in speziellem Java Applet im Webbrowser
Rich Fat Client	clientseitiges Framework (plattformabhängig)	z.B. Java-SWT, JFace-GUI
Managed Client	Client-Java-Middleware mit Client-Applikationsserver für Softwareverteilung, etc.	clientseitiges Framework

Client-Server-Architekturen

Architekturmuster	Applikationslogik	Präsentationsschicht
Thin Client	im Applikationsserver, z.B. JSP + JavaBeans mit Tomcat, J2EE-EnterpriseJavaBeans	HTML + JavaScript im Webbrowser
Portal	im Portal-Applikationsserver	Portlets im Browser
Rich Thin Client	sowohl im Client als auch im Applikationsserver	Java-AWT-GUI in speziellem Java Applet im Webbrowser
Rich Fat Client	clientseitiges Framework (plattformabhängig)	z.B. Java-SWT, JFace-GUI
Managed Client	Client-Java-Middleware mit Client-Applikationsserver für Softwareverteilung, etc.	clientseitiges Framework
OS-dependent Fat Client	überwiegend im Client: Desktop-Applikation	Windows- oder Linux-GUI

[torsten-horn.de]

Client-Server-Architekturen

Architekturmuster	Applikationslogik	Präsentationsschicht
Thin Client		
Portal		
Rich Thin Client		
Rich Fat Client		
Managed Client		
OS-dependent Fat Client		

datenintensiv (*data shipping*)

auftragsbezogen (*operation shipping*)

[torsten-horn.de]

Client-Server-Architekturen

Implementierung von Architekturmustern

- Ajax.
  - dynamisches Web für Thin Clients
- RPC, XML-RPC, DCOM, Java RMI.
  - Clients, die in der Lage sind, ein High-Level-Protokoll über TCP/IP abzuwickeln
  - Kommunikation zwischen Server-Anwendungen
- Web-Services.
  - flexible RPC-Technologie zur Client-Anbindung
  - Komposition komplexer Anwendungen durch Service-Verschaltung zwischen verschiedenen Servern
- CORBA.
  - industrielle, hoch-professionelle Technologie zur Kopplung von Server-Anwendungen
  - Anbindung von clientseitigen Frameworks
- Message-oriented-Middleware MOM.
  - professionelle Technologie zur asynchronen Kopplung von Server-Anwendungen

Bemerkungen:

- Verschiedene der Technologien, mit denen sich ein bestimmtes Architekturmuster implementieren lässt, werden mit dem Begriff „Middleware“ in Verbindung gebracht. Middleware – auch als „Architectural Glue“ bezeichnet – realisiert die Infrastruktur für und zwischen Komponenten. Es gibt verschiedene Kategorien von Middleware, je nach Granularität und Art der Komponenten.
- Middleware ist Software, welche die Erstellung verteilter Anwendungen dadurch vereinfacht, dass sie standardisierte Mechanismen zur Kommunikation von verteilten Komponenten zur Verfügung stellt.

Ajax
Einführung

Ajax = Asynchronous JavaScript and XML

Charakteristika:

- das synchrone Request-Response-Paradigma wird aufgebrochen
- Web-Seiten müssen nicht als Ganzes ersetzt, sondern können teilweise überladen werden. Schnittstelle: DOM-API
- auf klar definierten, offenen Standards basierend
- Browser- und plattformunabhängig
- es wird keine Art von „Ajax-Server“ benötigt, sondern auf bekannten Web-Server-Technologien aufgesetzt

Anwendung [Demo] :

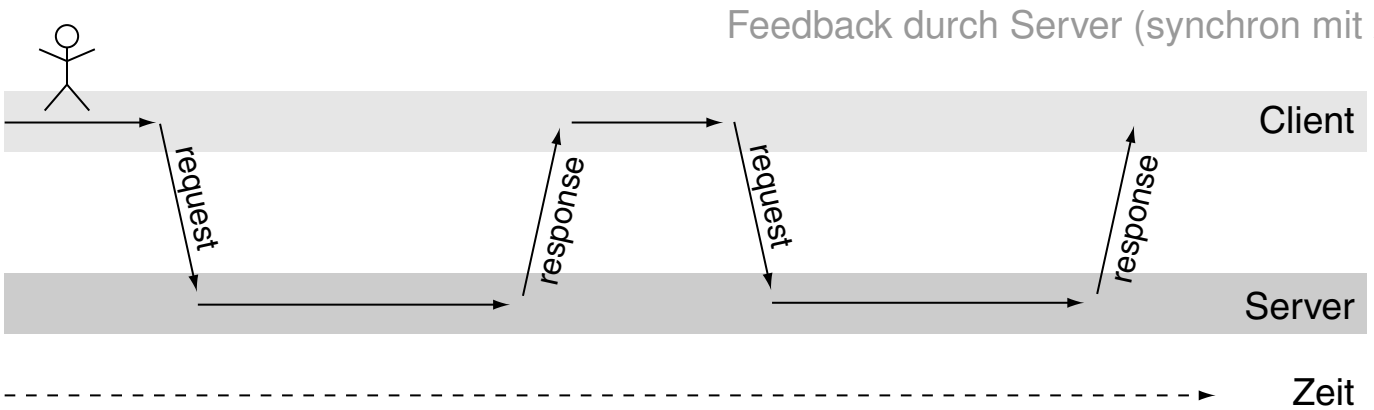
- Realisierung interaktiver Thin Clients
- standardisierte, einfache graphische Bedienelemente

Bemerkungen:

- Die Kernidee von Ajax besteht darin, einen HTTP-Request nebenläufig auszuführen und das Ergebnis des Requests in den DOM-Seitenbaum des Browsers einzufügen.
- Teilweise wird Ajax als reine Client-Technologie bezeichnet. [apache.org]
- Tatsächlich steht bei Ajax die Art und die Abwicklung der Kommunikation zwischen Client und Server im Vordergrund. Somit kann man Ajax als eine Technologie zur Umsetzung eines Architekturmusters verstehen: *"Ajax isn't a technology, it's more of a pattern – a way to identify and describe a useful design technique."* [McCarthy 2005, IBM]

Ajax

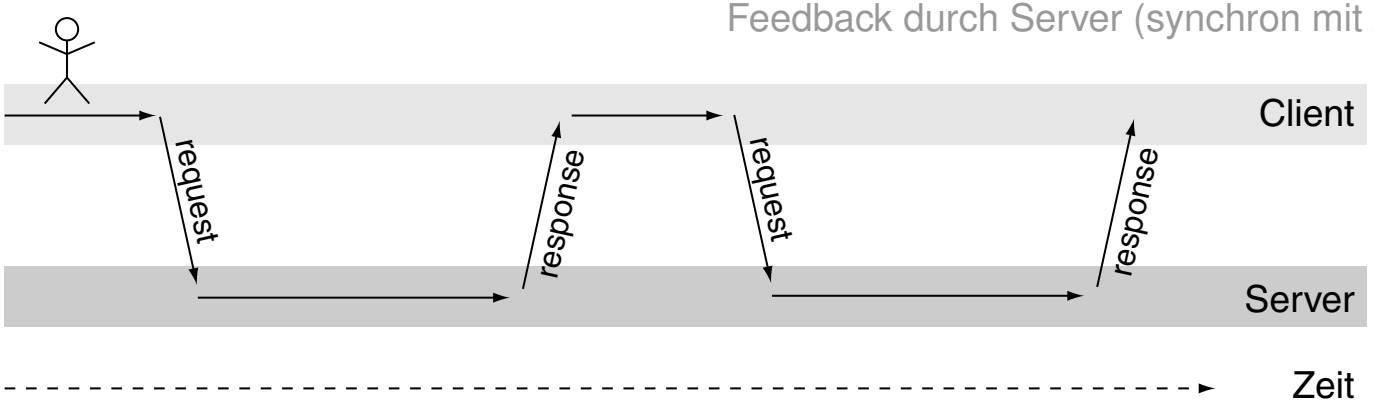
Einführung (Fortsetzung)



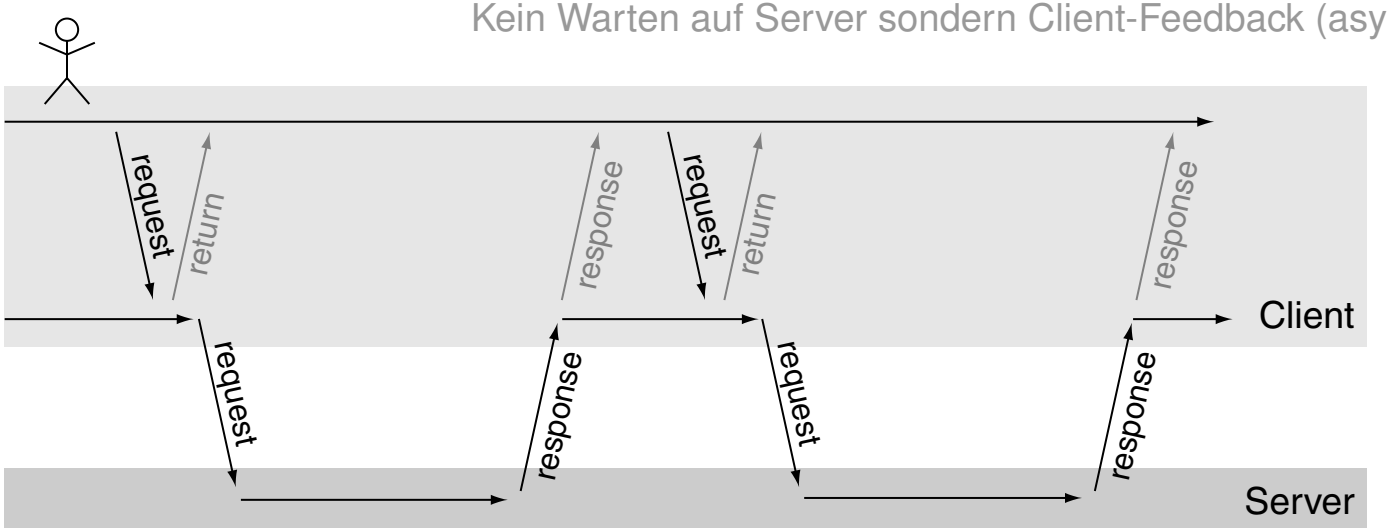
Ajax

Einführung (Fortsetzung)

Feedback durch Server (synchron mit Antwort)



Kein Warten auf Server sondern Client-Feedback (asynchron)



Ajax

Bestandteile einer Ajax-Anwendung

1. Event-Handler.

2. Callback-Funktion.

3. Server-Funktion.

Ajax

Bestandteile einer Ajax-Anwendung

1. Event-Handler.
   - realisiert als JavaScript-Funktion im Client, typischerweise im HTML-Dokument
   - wird bei entsprechender Anwenderaktion aufgerufen
   - instanziiert (bei jedem Aufruf) ein `XMLHttpRequest`-Objekt
   - meldet eine Callback-Funktion im `XMLHttpRequest`-Objekt an
   - ruft die zur Anwenderaktion gehörende Server-Funktion auf

2. Callback-Funktion.

3. Server-Funktion.

Ajax

Bestandteile einer Ajax-Anwendung

1. Event-Handler.
   - realisiert als JavaScript-Funktion im Client, typischerweise im HTML-Dokument
   - wird bei entsprechender Anwenderaktion aufgerufen
   - instanziiert (bei jedem Aufruf) ein `XMLHttpRequest`-Objekt
   - meldet eine Callback-Funktion im `XMLHttpRequest`-Objekt an
   - ruft die zur Anwenderaktion gehörende Server-Funktion auf

2. Callback-Funktion.
   - realisiert als JavaScript-Funktion im Client, typischerweise im HTML-Dokument
   - wird bei jedem `readyState`-Event vom `XMLHttpRequest`-Objekt aufgerufen
   - filtert bezüglich des letzten `readyState` (4) und HTTP-O.K.-Status-Code (200)
   - verarbeitet die zurückgegebene XML-Datei der Server-Funktion oder ruft für die Verarbeitung eine weitere JavaScript-Funktion auf und manipuliert den DOM

3. Server-Funktion.

Ajax

Bestandteile einer Ajax-Anwendung

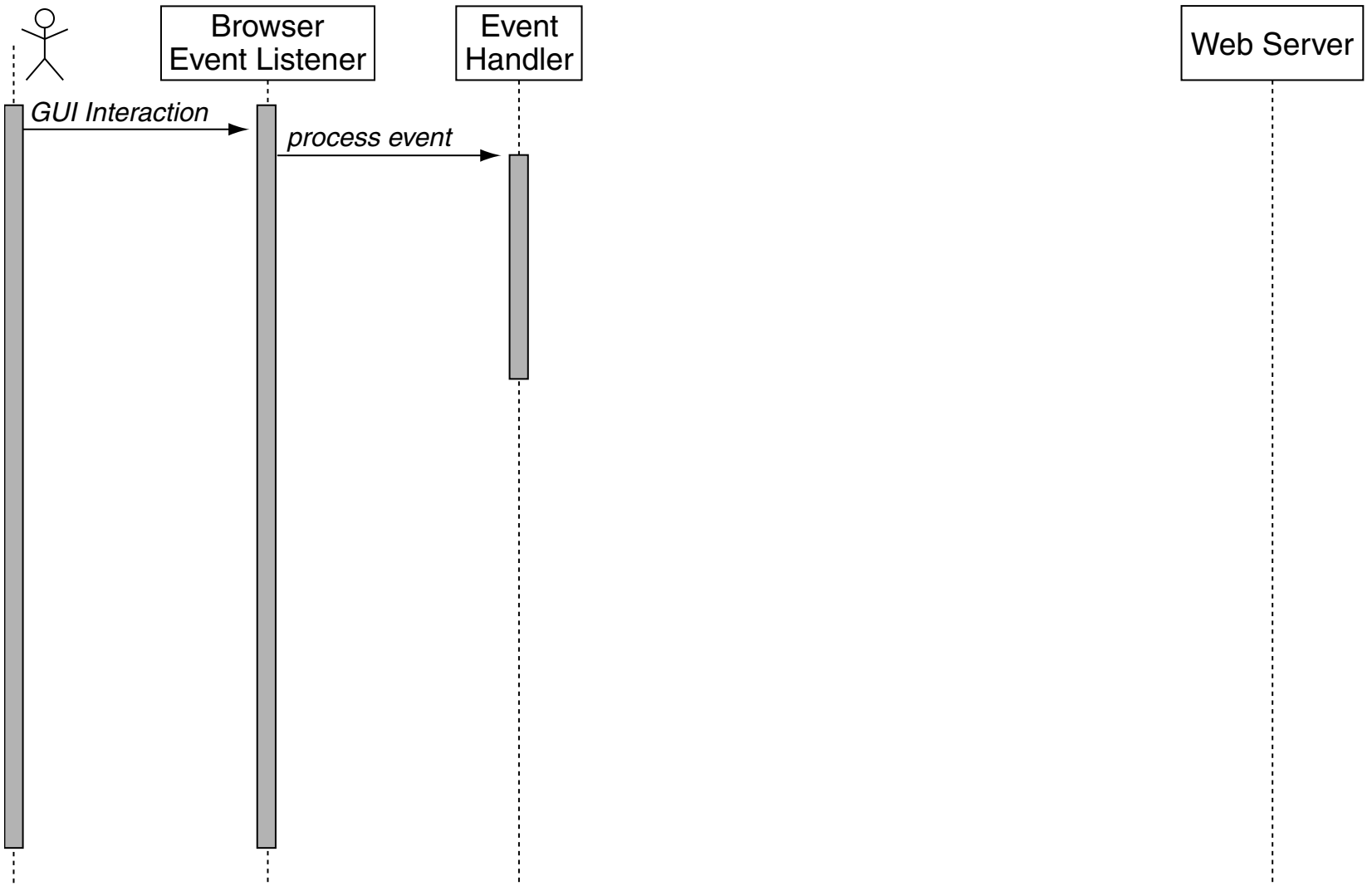
1. Event-Handler.
   - realisiert als JavaScript-Funktion im Client, typischerweise im HTML-Dokument
   - wird bei entsprechender Anwenderaktion aufgerufen
   - instanziiert (bei jedem Aufruf) ein `XMLHttpRequest`-Objekt
   - meldet eine Callback-Funktion im `XMLHttpRequest`-Objekt an
   - ruft die zur Anwenderaktion gehörende Server-Funktion auf

2. Callback-Funktion.
   - realisiert als JavaScript-Funktion im Client, typischerweise im HTML-Dokument
   - wird bei jedem `readyState`-Event vom `XMLHttpRequest`-Objekt aufgerufen
   - filtert bezüglich des letzten `readyState` (4) und HTTP-O.K.-Status-Code (200)
   - verarbeitet die zurückgegebene XML-Datei der Server-Funktion oder ruft für die Verarbeitung eine weitere JavaScript-Funktion auf und manipuliert den DOM

3. Server-Funktion.
   - wird mittels Standardtechnologie (CGI, PHP-Script, Servlet, etc.) auf einem Web-Server zur Verfügung gestellt
   - generiert eine XML-Datei als Rückgabewert

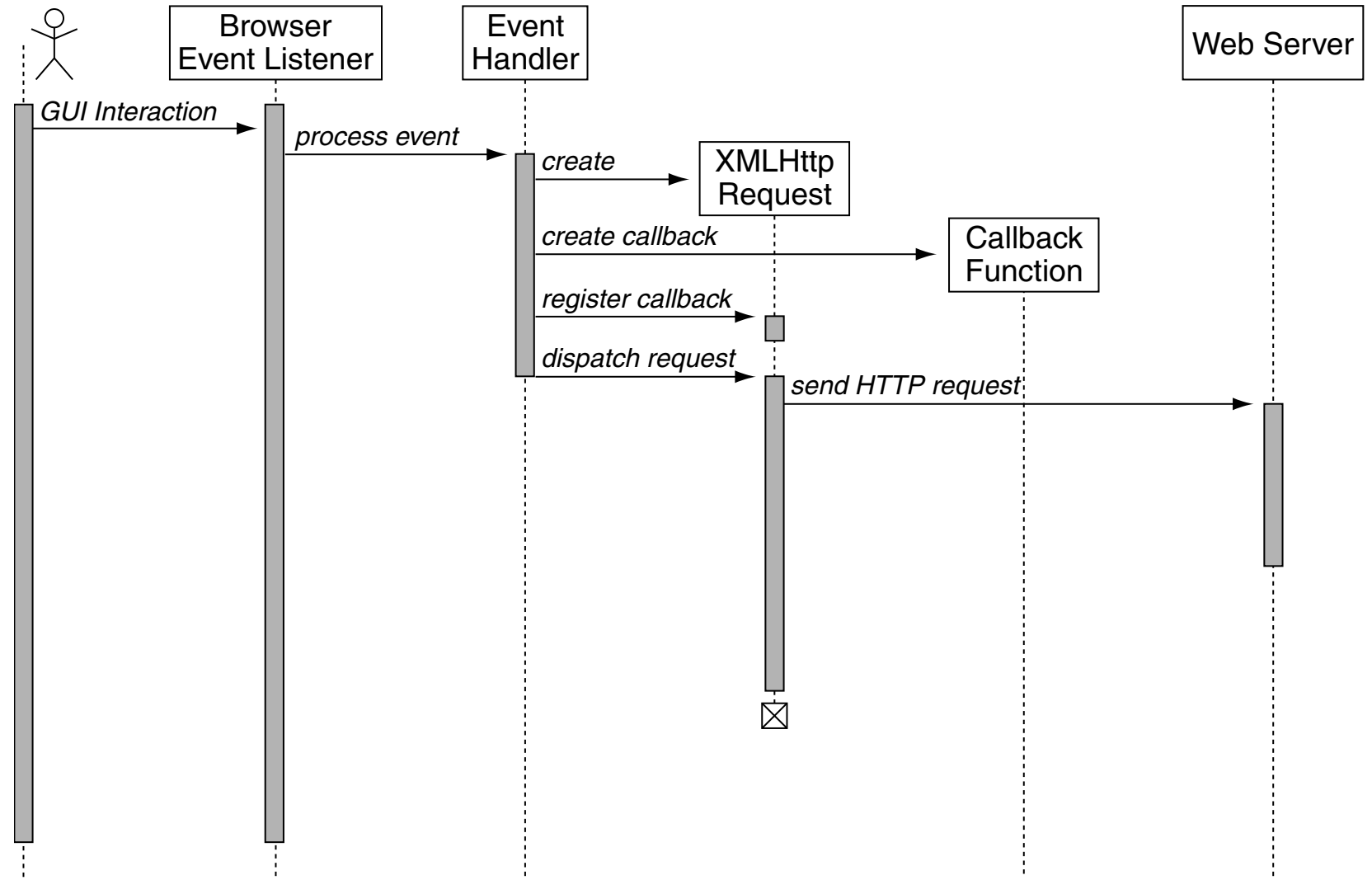
Ajax

Ablauf einer Ajax-Interaktion



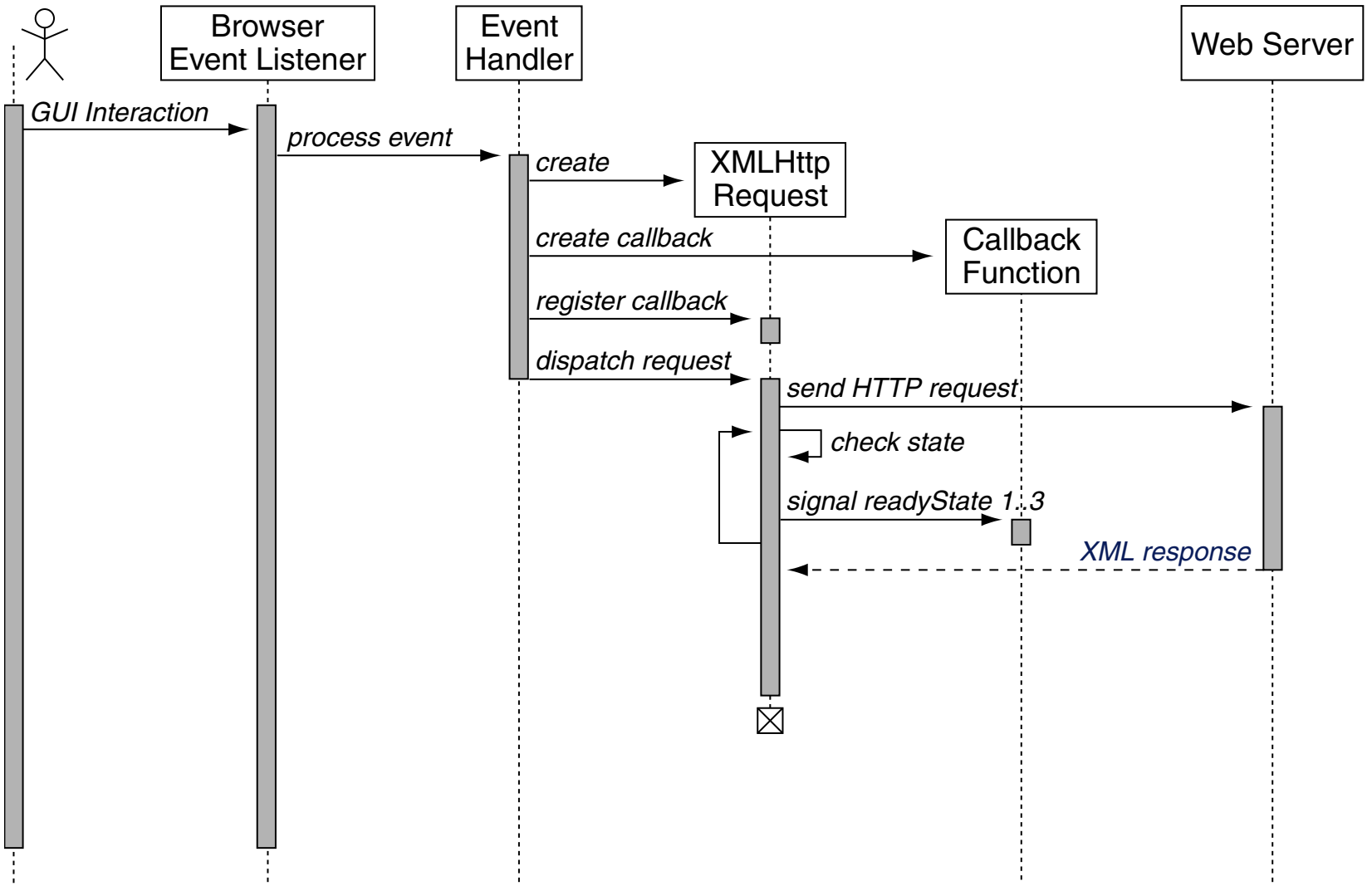
Ajax

Ablauf einer Ajax-Interaktion (Fortsetzung)



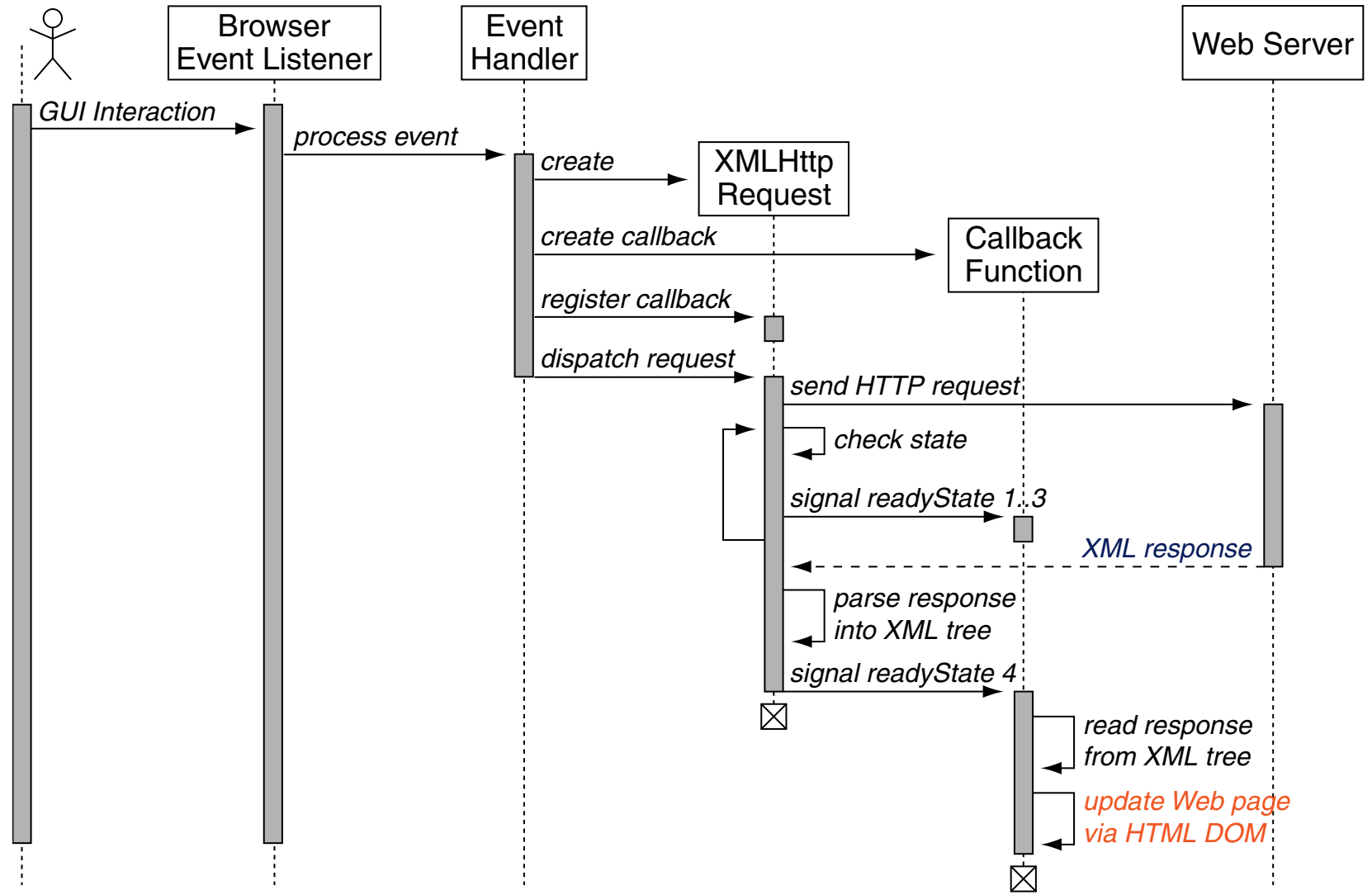
Ajax

Ablauf einer Ajax-Interaktion (Fortsetzung)



Ajax

Ablauf einer Ajax-Interaktion (Fortsetzung)



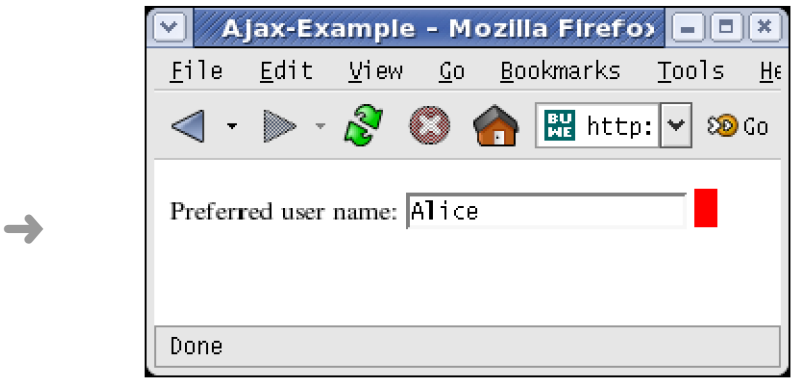
Ajax

Beispiel: Überwachung von Eingabefeld

[Demo]

Ajax

Beispiel: Überwachung von Eingabefeld



[Demo]

HTML-Code:

```
...
<form action="">
  Preferred user name:
  <input name="username" type="text"
    onkeyup="myEventHandler('http://.../check-username.php?q=' +
       this.value, processResponseXMLUsername)" />

</form>
```

Ajax

Beispiel: Überwachung von Eingabefeld

[Demo]

HTML-Code:

```
<style type="text/css">
  span.available{background-color: green;}
  span.unavailable{background-color: red;}
</style>
...
<form action="">
  Preferred user name:
  <input name="username" type="text"
    onkeyup="myEventHandler('http://.../check-username.php?q=' +
        this.value, processResponseXMLUsername)" />
  <span class="available" id="nameCheck">     </span>
</form>
```

Ajax

Beispiel: Überwachung von Eingabefeld (Fortsetzung)

Generischer JavaScript-Code für Event-Handler:

```
function myEventHandler(url, processResponseXMLFunction) {
  var req = createXMLHttpRequest();
  if(req) {
    req.onreadystatechange =    // Create and register callback.
       createCallbackFunction(req, processResponseXMLFunction);
    req.open("GET", url, true); // Dispatch request.
    req.send(null);             // Send HTTP request.
  }
}
```

Ajax

Beispiel: Überwachung von Eingabefeld (Fortsetzung)

Generischer JavaScript-Code für Event-Handler:

```
function myEventHandler(url, processResponseXMLFunction) {
  var req = createXMLHttpRequest();
  if(req) {
    req.onreadystatechange =    // Create and register callback.
       createCallbackFunction(req, processResponseXMLFunction);
    req.open("GET", url, true); // Dispatch request.
    req.send(null);             // Send HTTP request.
  }
}

function createXMLHttpRequest() {
  var req = false;
  if(window.XMLHttpRequest) {
    // XMLHttpRequest object for non-Microsoft browsers.
    req = new XMLHttpRequest();
  } else if(window.ActiveXObject) {
    // XMLHttpRequest object for newer Microsoft IE.
    req = new ActiveXObject("Msxml2.XMLHTTP");
  }
  return req;
}
```

Ajax

Beispiel: Überwachung von Eingabefeld (Fortsetzung)

Generischer JavaScript-Code der Callback-Funktion:

```
function createCallbackFunction(req, processResponseXMLFunction) {
  return function() {
    // Check whether readyState is complete.
    if(req.readyState == 4) {
      // Check whether server response is O.K.
      if(req.status == 200) {
        processResponseXMLFunction(req.responseXML);
      } else { alert("HTTP error: "+req.status); }
    }
  }
}
```

Ajax

Beispiel: Überwachung von Eingabefeld (Fortsetzung)

Generischer JavaScript-Code der Callback-Funktion:

```
function createCallbackFunction(req, processResponseXMLFunction) {
  return function() {
    // Check whether readyState is complete.
    if(req.readyState == 4) {
      // Check whether server response is O.K.
      if(req.status == 200) {
        processResponseXMLFunction(req.responseXML);
      } else { alert("HTTP error: "+req.status); }
    }
  }
}
```

Spezifischer JavaScript-Code der Callback-Funktion zur Verarbeitung der XML-Antwortdatei:

```
function processResponseXMLUsername(xmltree) {
  var result = xmltree.documentElement.getElementsByTagName(
    'result')[0].firstChild.data;
  var message = document.getElementById('nameCheck');
  if(result == 1){ message.className = 'unavailable'; }
  else{ message.className = 'available';
  }
}
```

Bemerkungen:

- Das Beispiel beschreibt ein wiederverwendbares Pattern für Ajax-Anwendungen: der generische JavaScript-Code kann unverändert in jedem HTML-Dokument zum Einsatz kommen. Lediglich zur Verarbeitung der XML-Antwort ist eine spezielle JavaScript-Funktion zu definieren. Diese Funktion zusammen mit der Server-Funktion bilden die Argumente von `myEventHandler()`, der beliebigen Events im HTML-Dokument zugeordnet werden kann.
- Die Art der Verwendung von `createCallbackFunction` hat Ähnlichkeit zum Template-Method-Pattern.
- Der Rückgabewert von `createCallbackFunction` ist ein Funktionsliteral, das eine anonyme Funktion definiert.
- In der anonymen Funktion wird das zweite Argument von `createCallbackFunction`, `processResponseXMLFunction`, als Funktionsname in der Position eines Funkionsaufrufs verwendet wird.

Ajax

Beispiel: Überwachung von Eingabefeld (Fortsetzung)

Server-Funktion `check-username.php` generiert XML-Antwortdatei [Demo: 1, 2] :

```
<?php
```

```
?>
<response>
  <result> <?php                                    ?> </result>
</response>
```

Ajax

Beispiel: Überwachung von Eingabefeld (Fortsetzung)

Server-Funktion `check-username.php` generiert XML-Antwortdatei [Demo: 1, 2] :

```
<?php
header('Content-Type: text/xml');
echo '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>';
?>
<response>
  <result> <?php                                    ?> </result>
</response>
```

Ajax

Beispiel: Überwachung von Eingabefeld (Fortsetzung)

Server-Funktion `check-username.php` generiert XML-Antwortdatei [Demo: 1, 2] :

```
<?php
header('Content-Type: text/xml');
echo '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>';
function nameInUse($q) {
  if (isset($q)){
    switch(strtolower($q)) {
      case 'alice' :
        return '1';
        break;
      case 'bob' :
        return '1';
      ...
      default:
        return '0';
    }
  }else{ return '0'; }
}
?>
<response>
  <result> <?php echo nameInUse($_GET['q']) ?> </result>
</response>
```

Ajax

Quellen zum Nachlernen und Nachschlagen im Web

- Google. *Google Web Toolkit GWT.*
  code.google.com/webtoolkit
- T. Horn. *Technische Kurzdokumentationen.*
  www.torsten-horn.de/techdocs
- D. McLellan. *Very Dynamic Web Interfaces.*
  www.xml.com/pub/a/2005/02/09/xml-http-request.html
- P. McCarthy. *Ajax for Java developers: Build dynamic Java applications.*
  www.ibm.com/developerworks/java/library/j-ajax1
- P. McCarthy. *Ajax for Java developers: Exploring the Google Web Toolkit.*
  www.ibm.com/developerworks/java/library/j-ajax4
- W3 Schools. *Ajax Tutorial.*
  www.w3schools.com/ajax

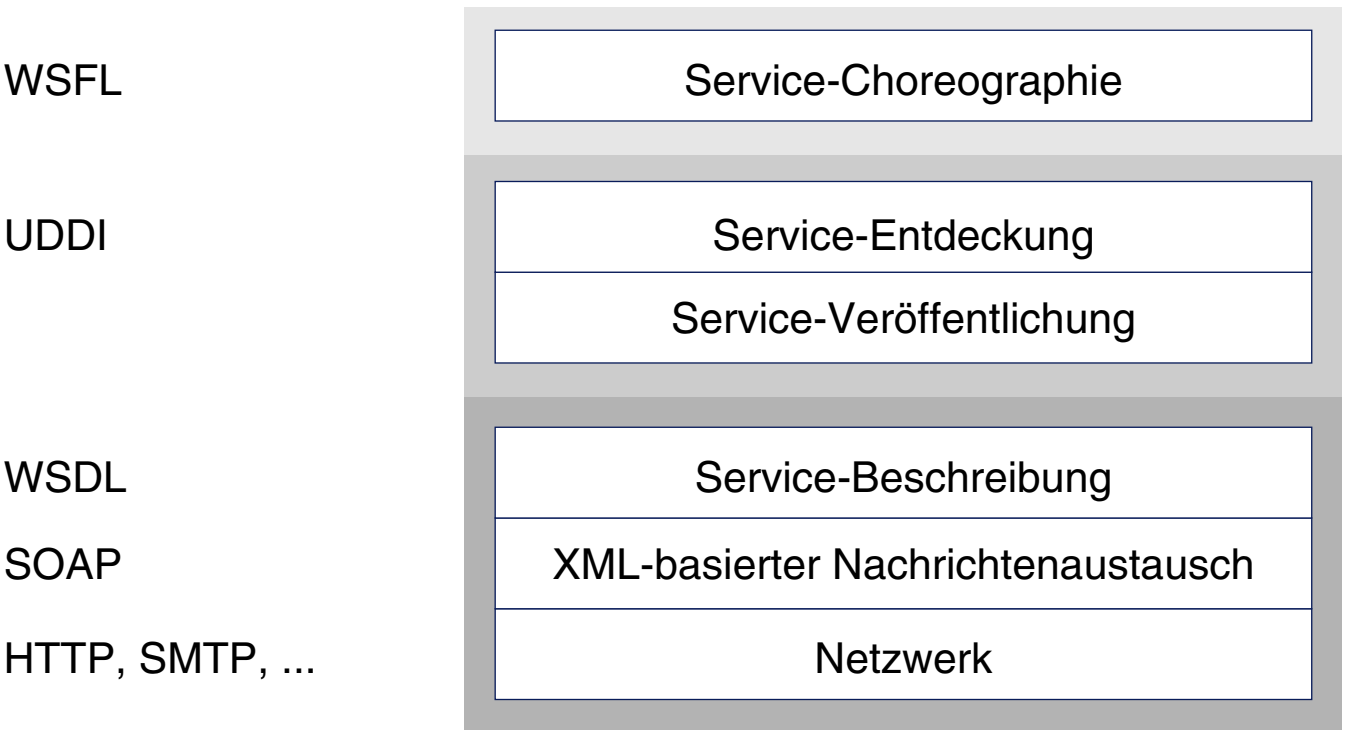
Kapitel WT:VI (Fortsetzung)

Web-Services

Web-Services ermöglichen die Abwicklung von Dienstleistungen und Geschäften über das Internet.

Web-Services

Web-Services ermöglichen die Abwicklung von Dienstleistungen und Geschäften über das Internet.



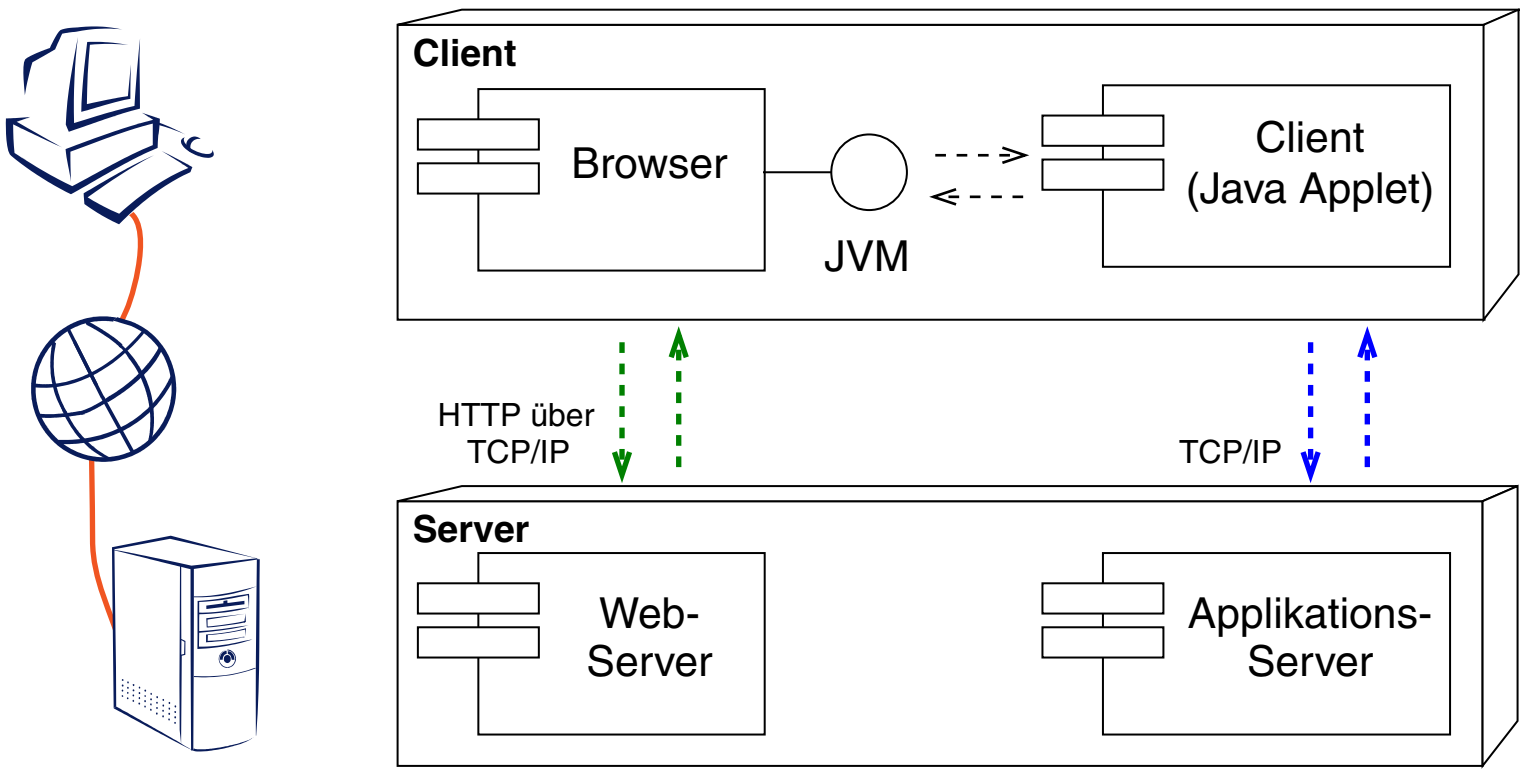
Web-Service-Definition aus Protokollsicht [Kilgore 2002]:

Web-Service = HTTP + XML + SOAP + WSDL (+ UDDI + WSFL)

Web-Services

Rich Client Anbindung

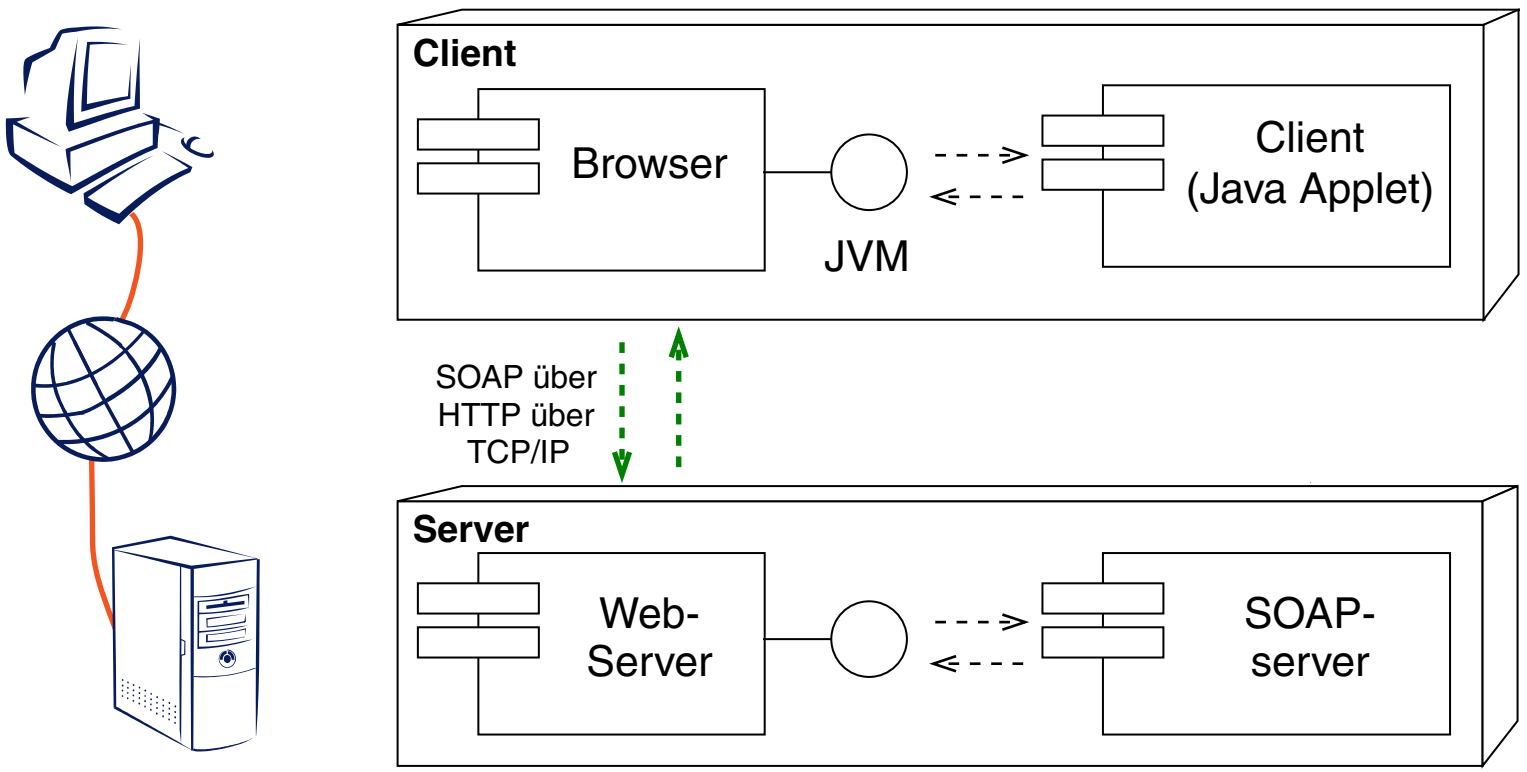
Verwendung von eigener TCP/IP-Verbindung und proprietärem Protokoll:



Web-Services

Rich Client Anbindung (Fortsetzung)

Verwendung von SOAP via Standard-HTTP-Protokoll zum Web-Server:

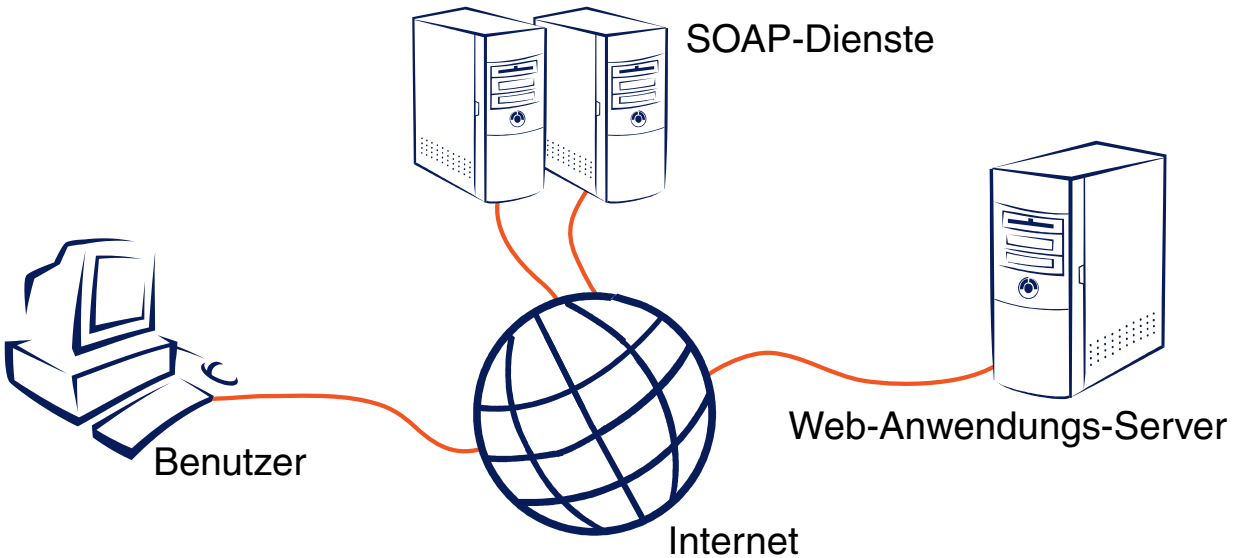


Bemerkungen:

- Wenn man Web-Services entwickelt, muss man von der klassischen Client-Server-Sicht umdenken; insbesondere muss man sich bei SOAP zwei Server vorstellen. [Lohrer 2003]

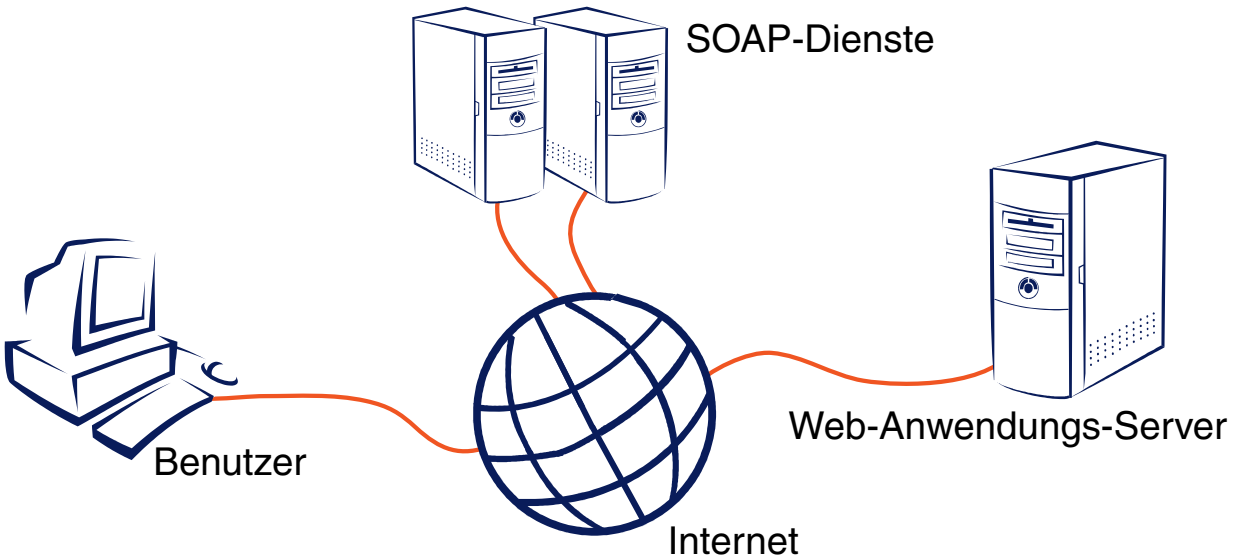
Web-Services

Szenario 1: Simulation integriert im CAD-Dokument



Web-Services

Szenario 1: Simulation integriert im CAD-Dokument



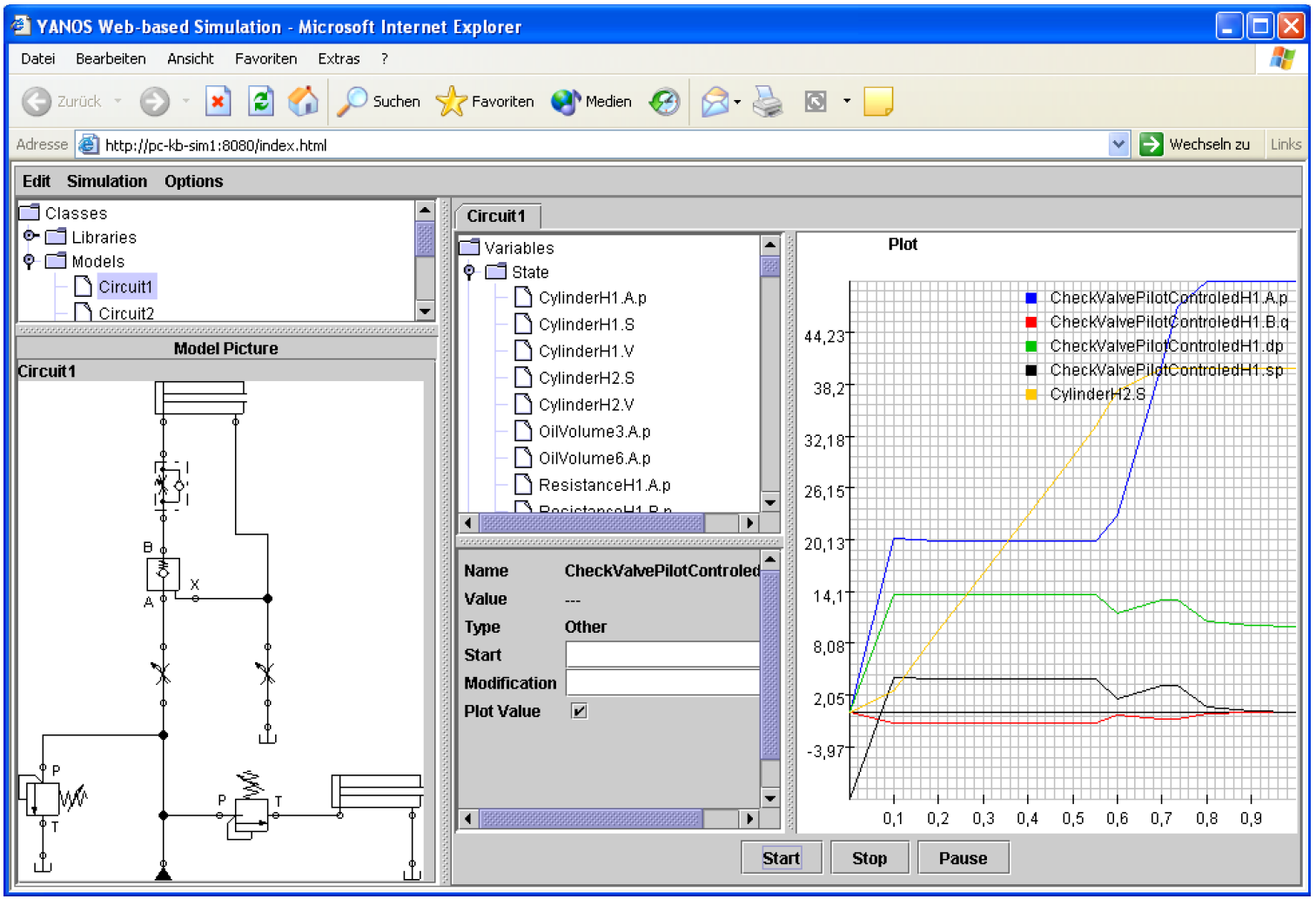
Der Web-Anwendungs-Server

- bereitet mittels SOAP-Dienst A das Modell zur Simulation auf,
- lässt über SOAP-Dienst B die Simulation durchführen,
- lässt über SOAP-Dienst C die Simulationsergebnisse aufbereiten,

und präsentiert das Ergebnis dem Anwender.

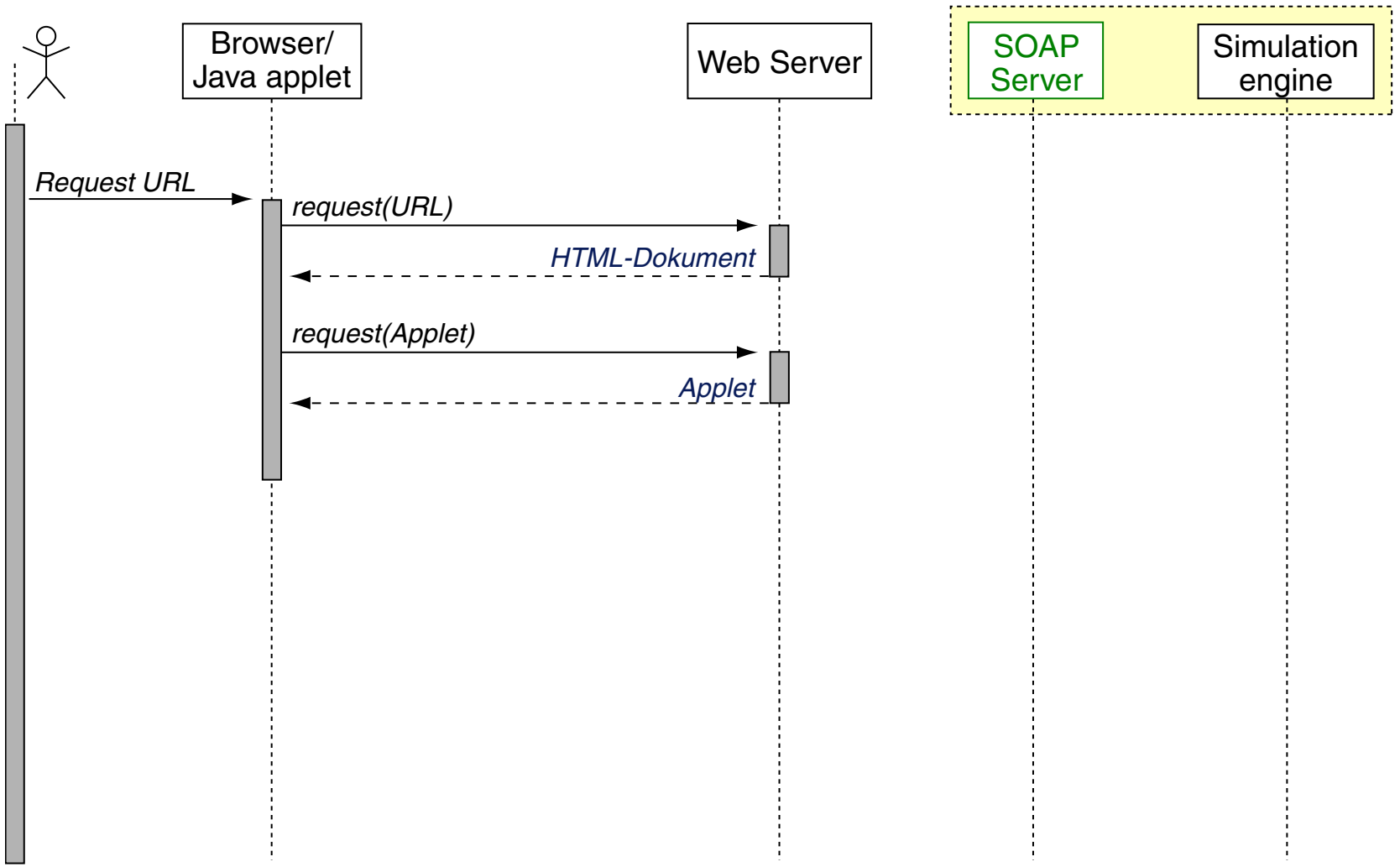
Web-Services

Szenario 1: Simulation integriert im CAD-Dokument (Fortsetzung)



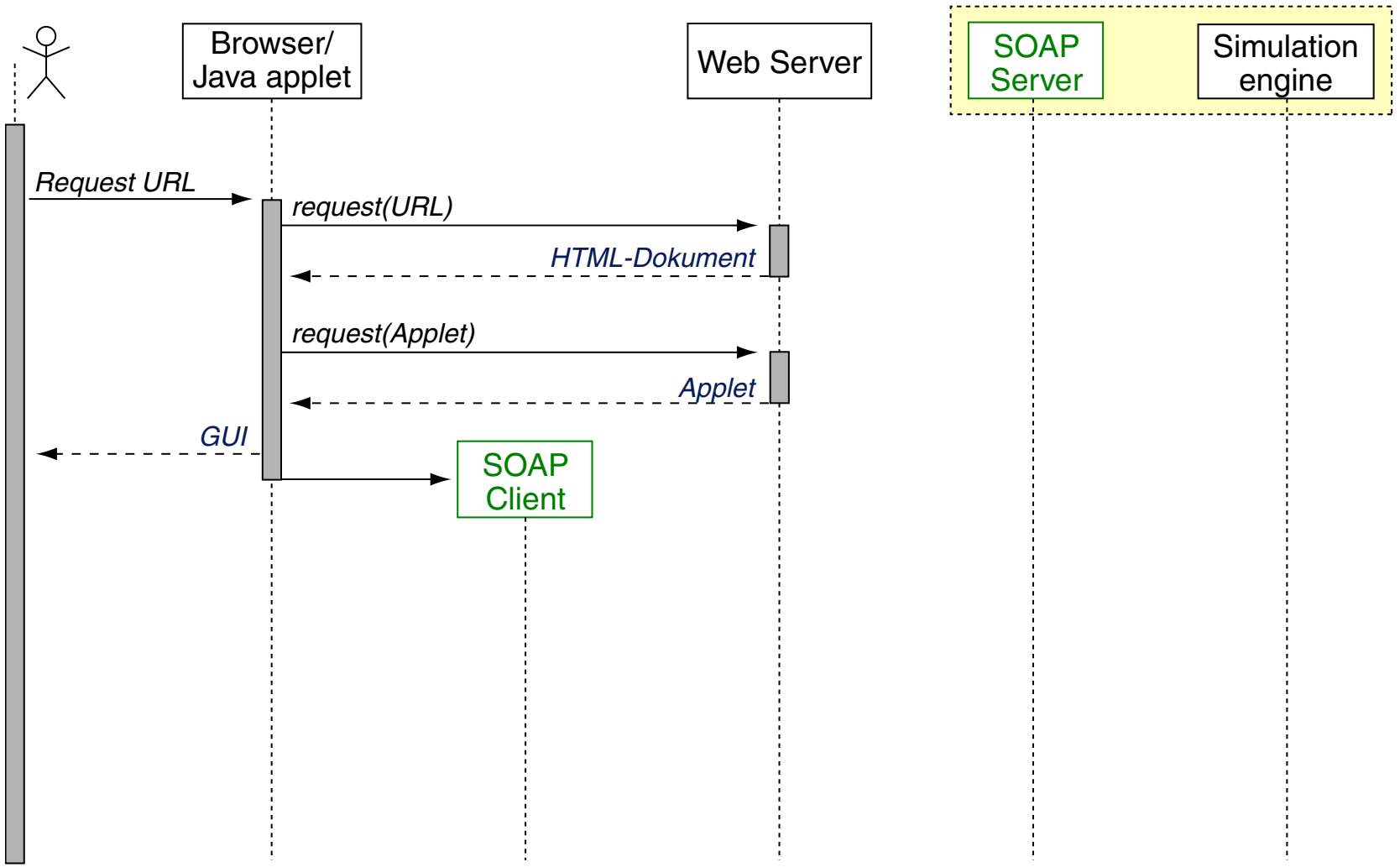
Web-Services

Szenario 1: Simulation integriert im CAD-Dokument (Fortsetzung)



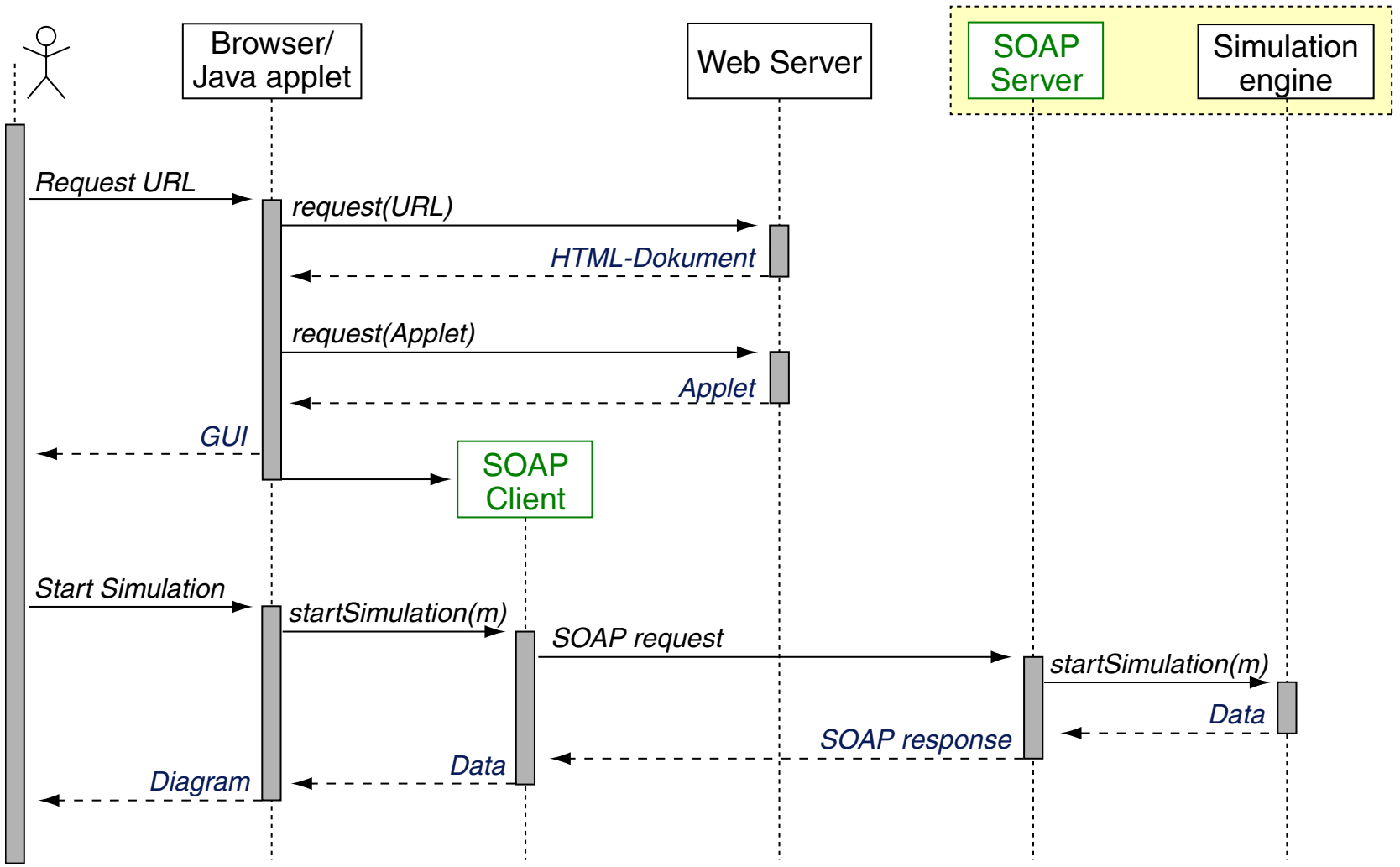
Web-Services

Szenario 1: Simulation integriert im CAD-Dokument (Fortsetzung)



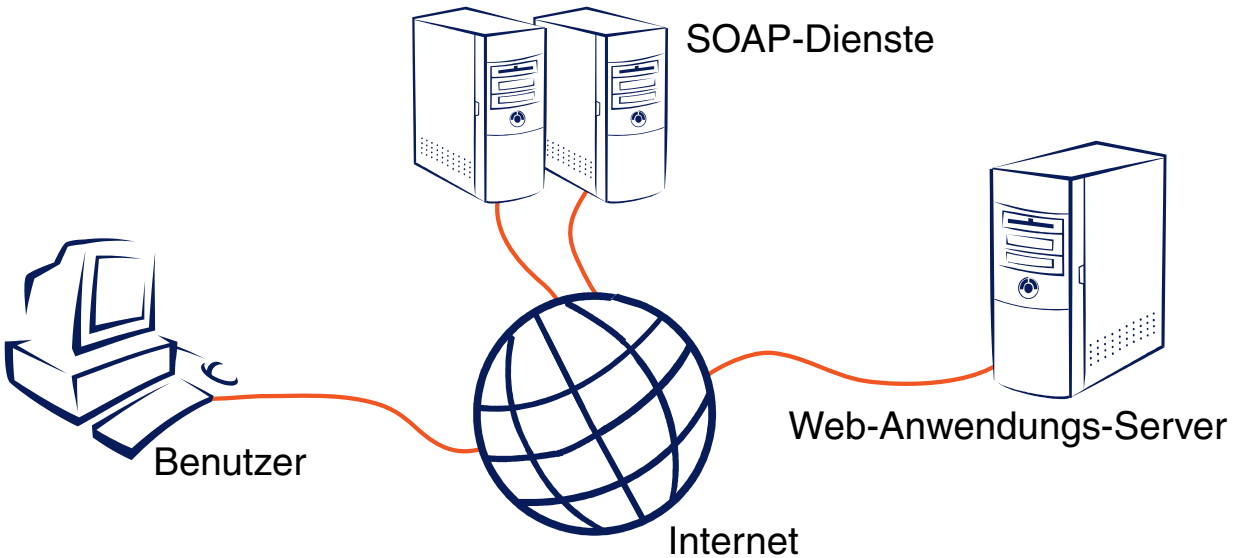
Web-Services

Szenario 1: Simulation integriert im CAD-Dokument (Fortsetzung)



Web-Services

Szenario 2: Übersetzung eines Textdokuments



Der Web-Anwendungs-Server

- überprüft über einen SOAP-Dienst X die Kreditkartennummer,
- erfragt beim SOAP-Dienst Y aktuelle Währungsumrechnungskurse,
- lässt über SOAP-Dienst A den ASCII-Text extrahieren,
- fordert über SOAP-Dienst B eine Übersetzung des Textes an,
- lässt über SOAP-Dienst C ein neues Dokument erstellen,

und präsentiert das Ergebnis dem Anwender.

Web-Services

Automatisierungsaspekte

- SOAP ist ein Mechanismus für entfernte Funktionsaufrufe, codiert in XML.
- SOAP ist unabhängig vom Transportprotokoll, meistbenutzt ist HTTP.

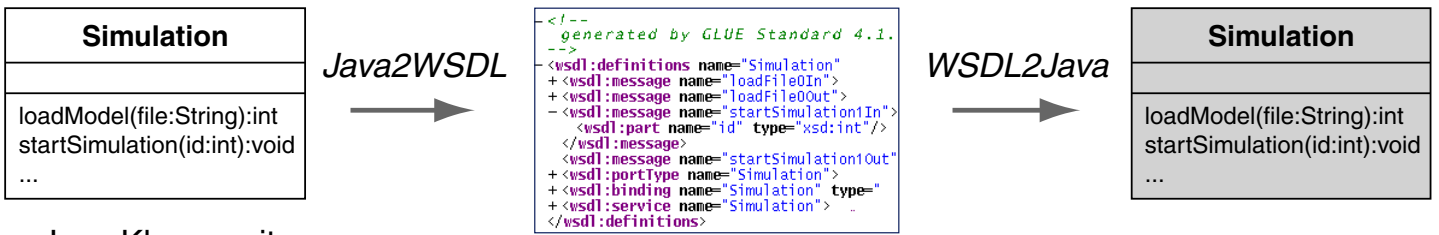
Vergleich mit anderen High-Level-Protokollen:

	RPC	RMI	DCOM	propiertäres TCP/IP	CORBA	SOAP
plattformunabhängig		+		+	+	+
herstellerunabhängig	o				+	+
sprachunabhängig	o		+	+	+	+
Browser-integrierbar		+		+	+	+
Protokollgenerierung			o		o	+
öffentliche Interfaces					o	+
Firewall-verträglich						+

Web-Services

Automatisierungsaspekte (Fortsetzung)

Bei der Anwendungsentwicklung für SOAP-Protokolle ist ein hoher Automatisierungsgrad möglich:



Java-Klasse mit Methoden

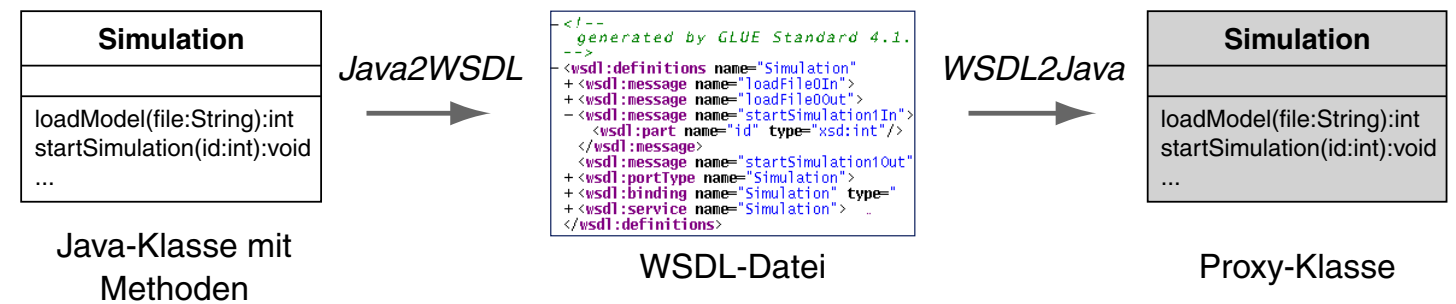
WSDL-Datei

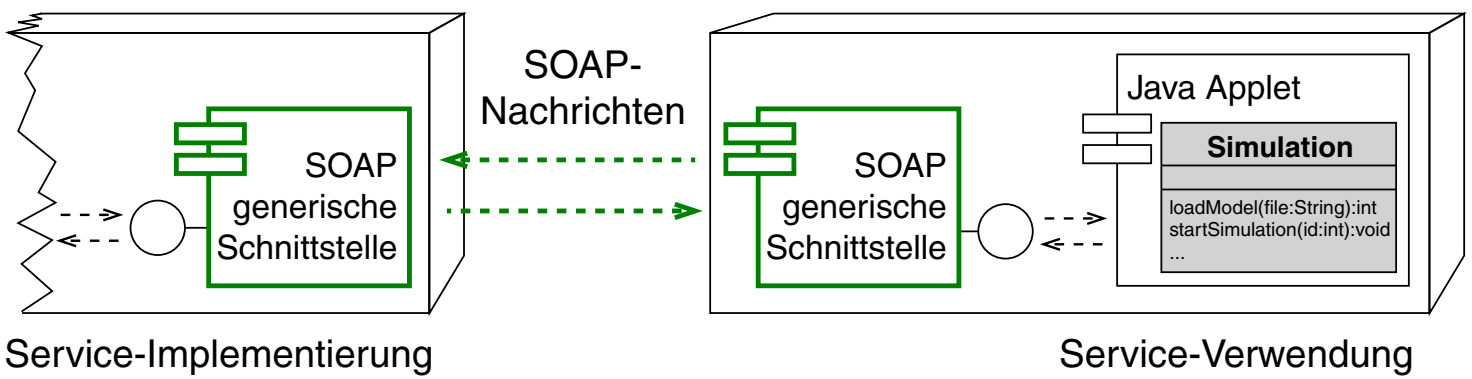
Proxy-Klasse

Web-Services

Automatisierungsaspekte (Fortsetzung)

Bei der Anwendungsentwicklung für SOAP-Protokolle ist ein hoher Automatisierungsgrad möglich:

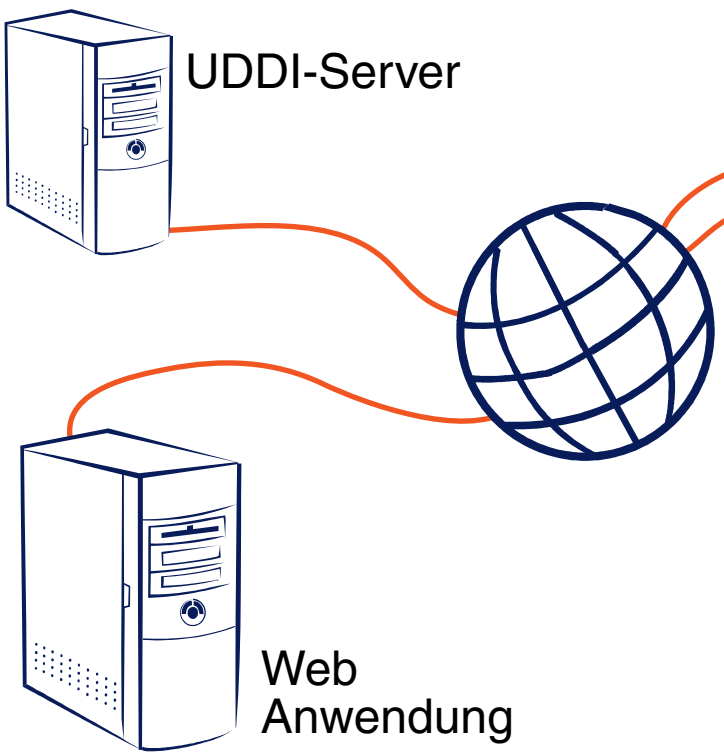




Web-Services

Automatisierungsaspekte (Fortsetzung)

Vollautomatische Kommunikation zwischen Applikationen via Internet (Stichwort: „Semantic Web“) auf Basis von UDDI: Universal Description, Discovery, and Integration (of web services).



1. Anfrage: Wer hat Dienst für Aufgabe?
2. Antwort: URL der WSDL-Datei
3. Anforderung der WSDL-Datei
4. Anwendung des Dienstes via SOAP