NOTES & DOMINO ENTWICKLERCAMP

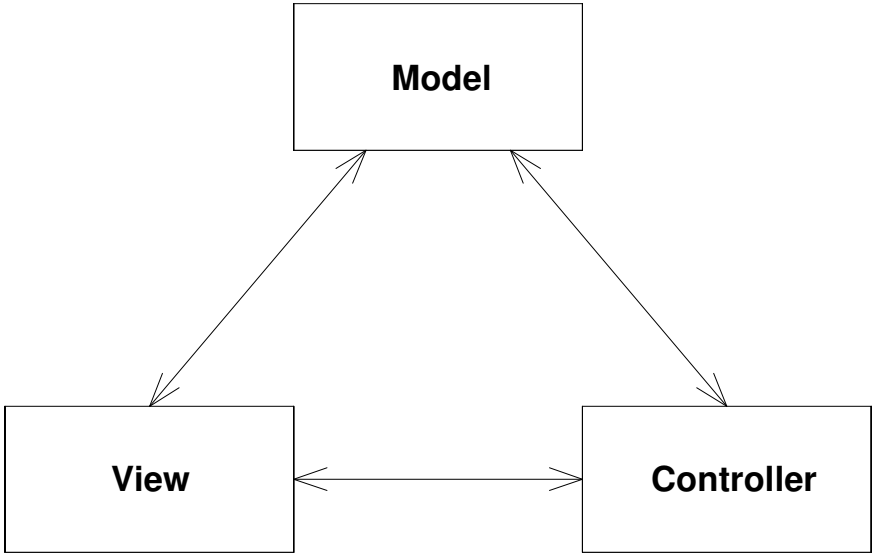
OOP in LotusScript für Web- und Notes-Anwendungen

25. Februar 2008

Bernd Hort
assono GmbH
bhort@assono.de
http://www.assono.de
+49/4101/48747

Agenda

- Vorstellung
- Model View Controller Pattern
- Basisklassen
- Eingabevalidierung Notes Client
- Eingabevalidierung Web Browser im Backend
- Eingabevalidierung Web Browser im Frontend
- Zusammenfassung
- Fragen & Antworten



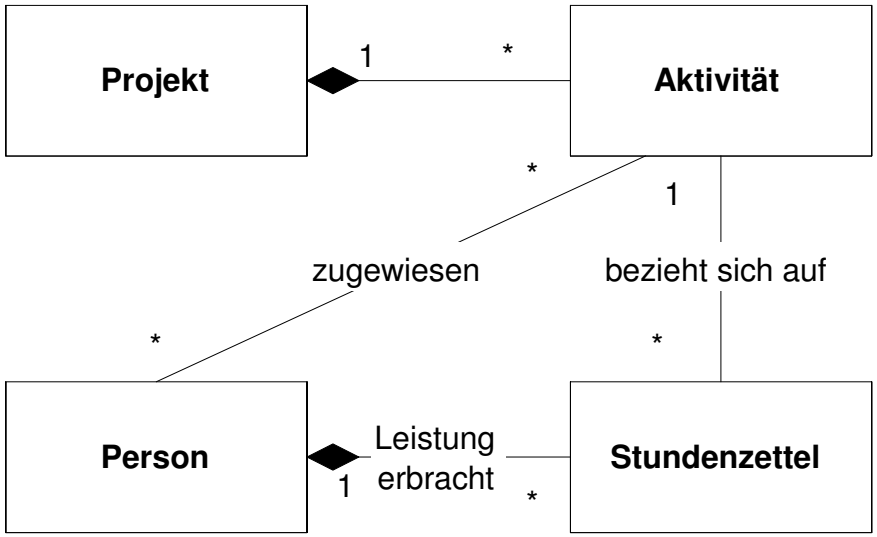
Vorstellung

- Bernd Hort
- Diplom-Informatiker
- Lotus Notes Anwendungsentwicklung seit 1995
- IBM Certified Application Developer - Lotus Notes and Domino 7
- IBM Certified System Administrator – Lotus Notes and Domino 7
- IBM Certified Instructor SA & AD – Lotus Notes and Domino 7
- Sprecher Lotusphere 2008

Certified for IBM Lotus e-business software

Beispiel-Anwendung

- Simple Projektverwaltung
- Bestandteile
  - Projekt
  - Aktivität
  - Person
  - Stundenzettel

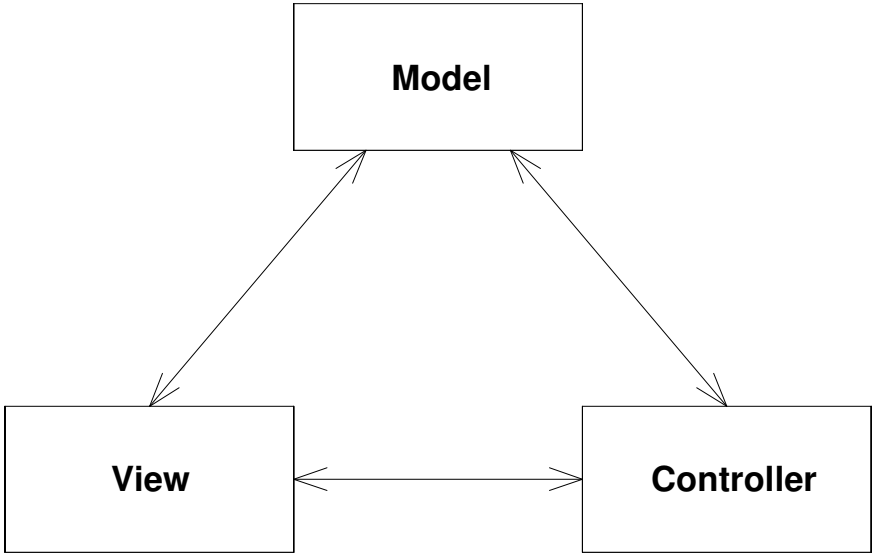


Demo Beispiel-Anwendung

Beispiel-Anwendung

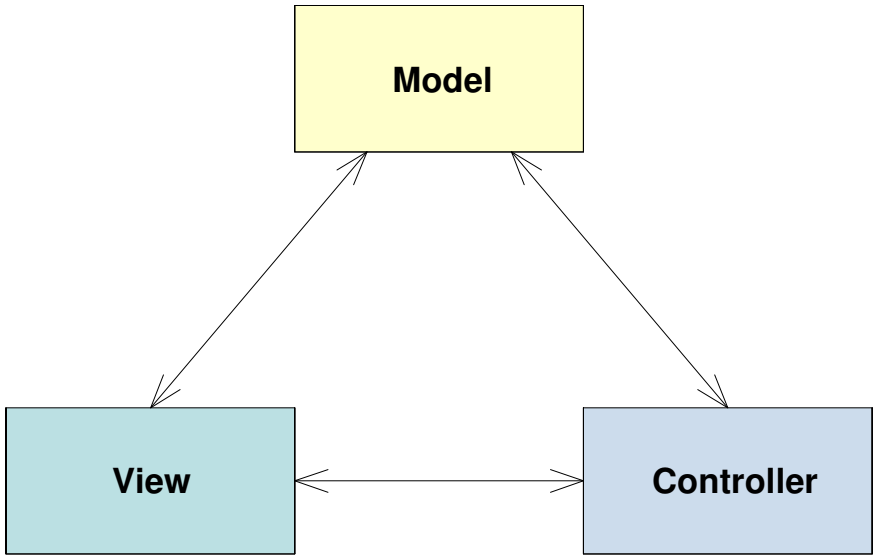
Agenda

- Vorstellung
- **Model View Controller Pattern**
- Basisklassen
- Eingabevalidierung Notes Client
- Eingabevalidierung Web Browser im Backend
- Eingabevalidierung Web Browser im Frontend
- Zusammenfassung
- Fragen & Antworten



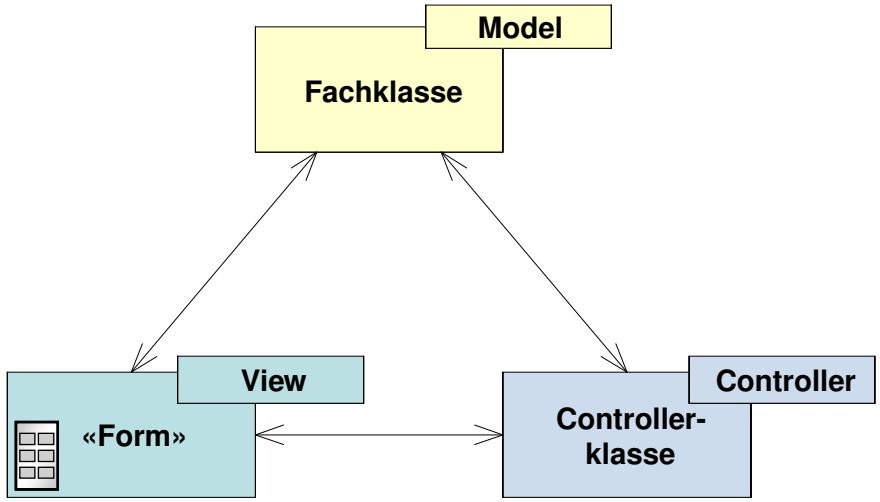
Model View Controller Pattern

- Ursprünglich aus Smalltalk
- Allgemeines OO-Prinzip für die Entwicklung von GUIs
- Trennung von Fachklassen und deren Darstellung
  - *Model* – Fachklasse
  - *View* – Darstellung
  - *Controller* – Benutzerinteraktion
- Die Fachklasse weiß nichts von seiner Darstellung!



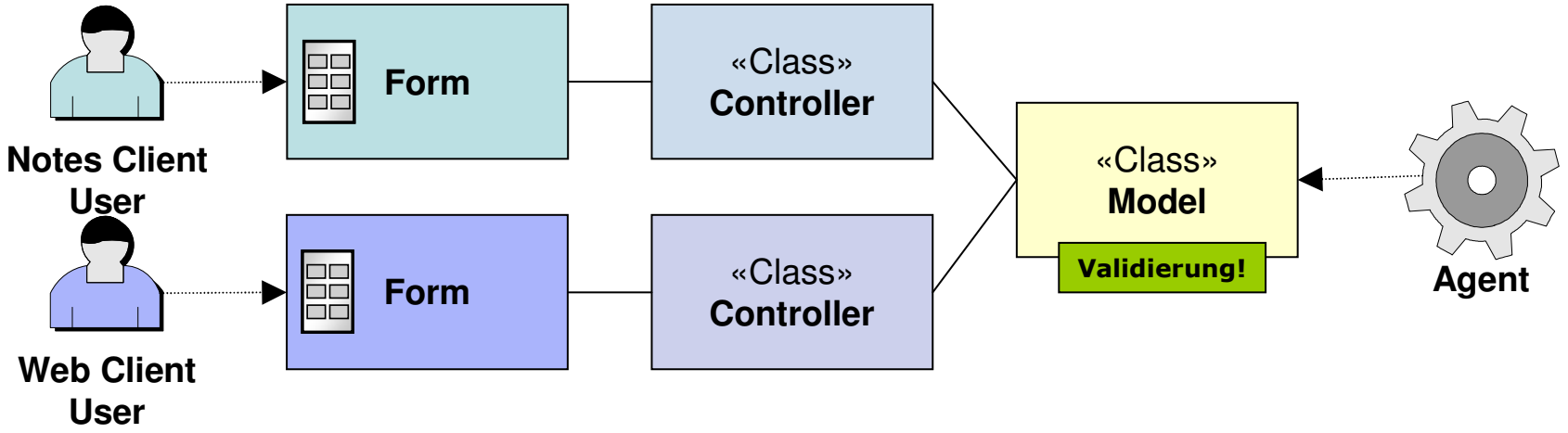
Model View Controller in Notes

- Die Maske stellt die Daten dar und nimmt die Benutzereingabe entgegen.
- Alle fachlichen Anforderungen werden in der Fachklasse implementiert.
- Die Controllerklasse ist das Verbindungsstück zwischen der Fachklasse und der Maske.



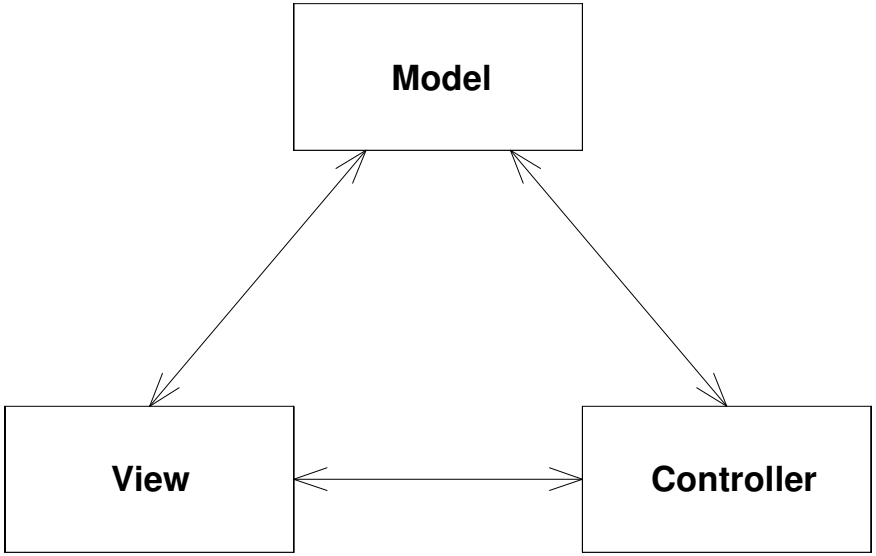
Umsetzung aller fachlichen Anforderungen in der Fachklasse

- Jegliche fachliche Anforderung sollte auch in der Fachklasse umgesetzt werden.
- Das beinhaltet insbesondere auch alle Eingabevalidierungen und Plausibilitätsüberprüfungen.
- Die Fachklasse wird so implementiert, dass sie keine UI-Methoden verwendet.
- Somit können die gleichen Überprüfungsroutinen sowohl für Agents als auch für Benutzereingaben verwendet werden.



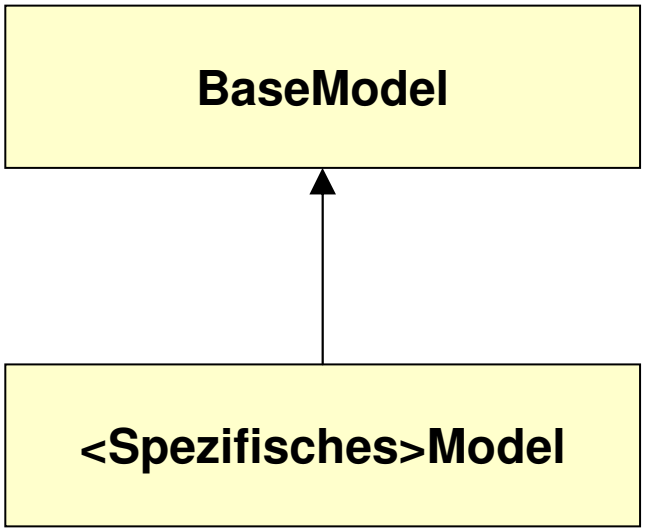
Agenda

- Vorstellung
- Model View Controller Pattern
- **Basisklassen**
- Eingabevalidierung Notes Client
- Eingabevalidierung Web Browser im Backend
- Eingabevalidierung Web Browser im Frontend
- Zusammenfassung
- Fragen & Antworten



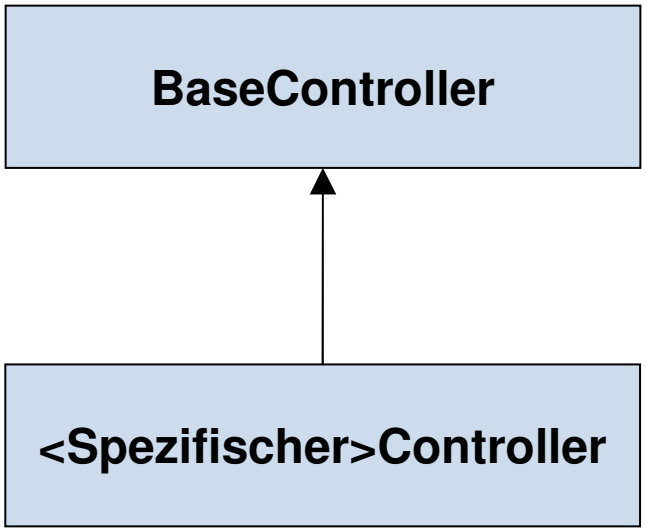
«Class» BaseModel

- Basis für alle fachlich motivierten Klassen
- Die meisten Methoden sind nur Schnittstellen-Definitionen, d.h. der Methodenrumpf ist leer
- Keine UI-Methoden bzw. –Klassen erlaubt
- Die eigentliche Fachklasse wird davon abgeleitet



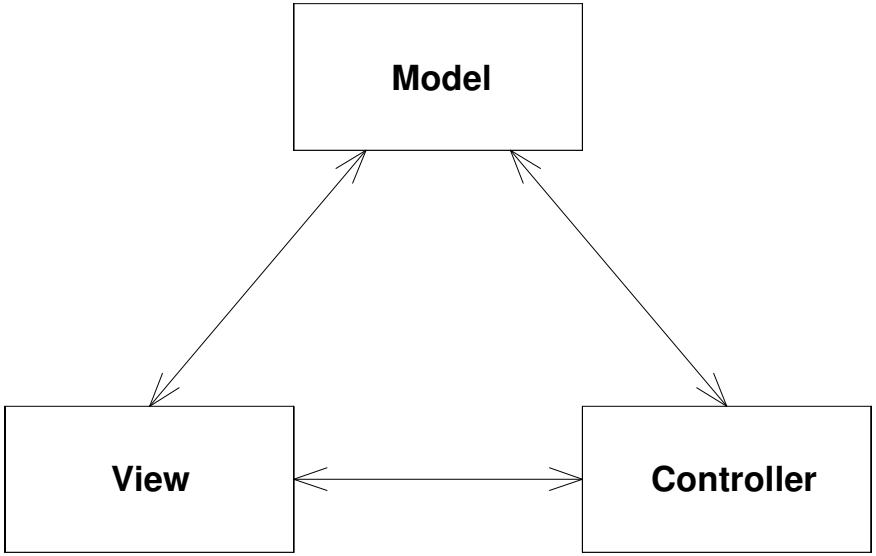
«Class» BaseController

- Schnittstelle zwischen Fachklasse und Maske
- Abfangen aller Events in der Maske



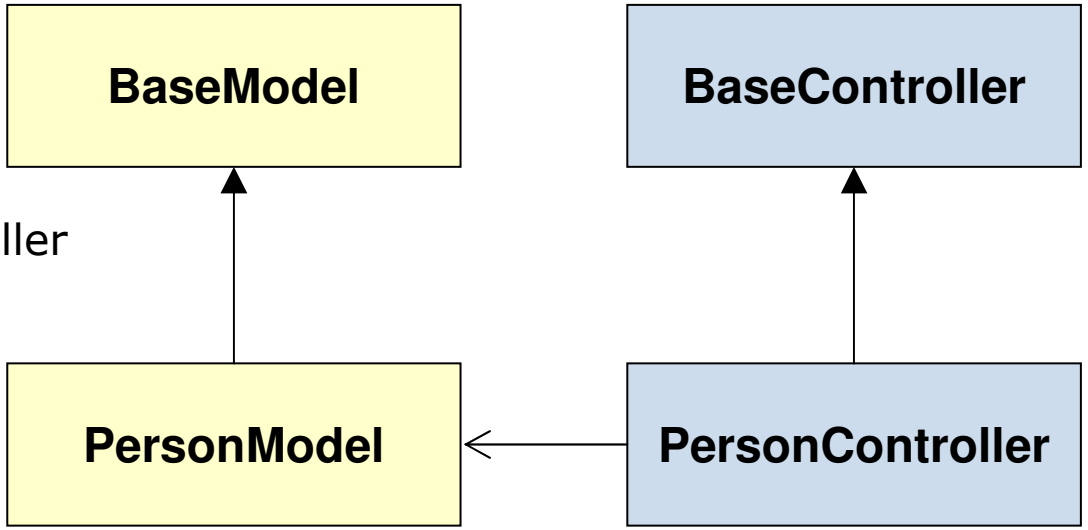
Agenda

- Vorstellung
- Model View Controller Pattern
- Basisklassen
- **Eingabevalidierung Notes Client**
- Eingabevalidierung Web Browser im Backend
- Eingabevalidierung Web Browser im Frontend
- Zusammenfassung
- Fragen & Antworten



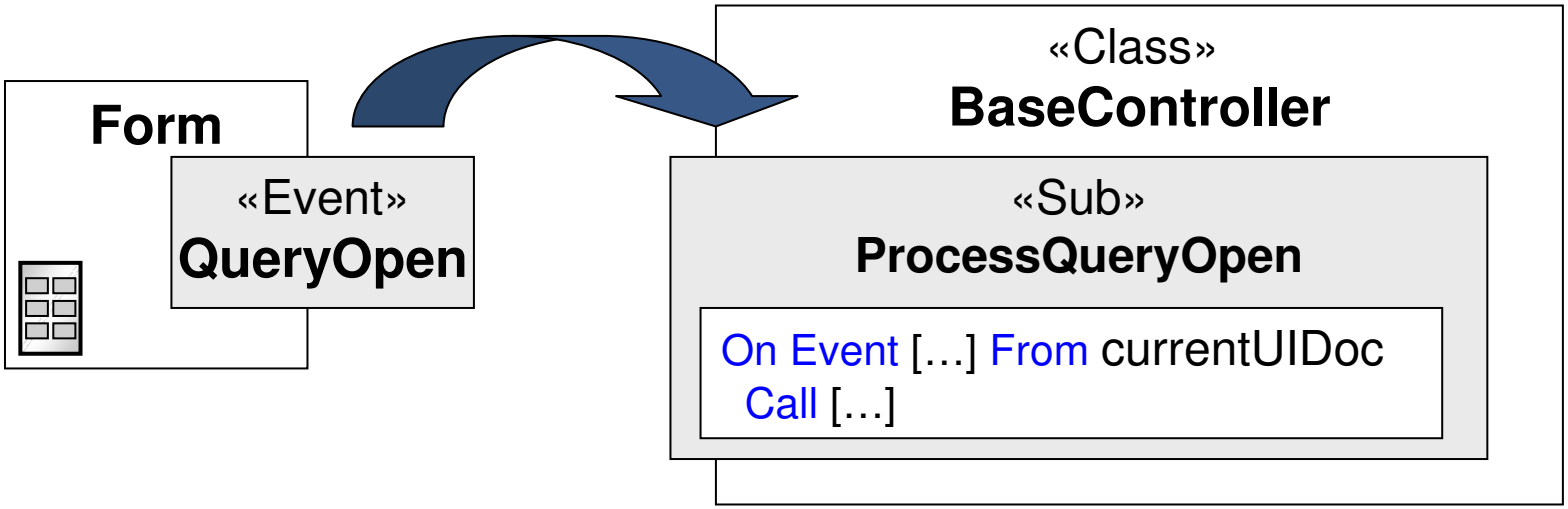
Person

- Fachliche Aspekte in «Class» PersonModel
- Verbindung zur Maske für den Notes Client über «Class» PersonController



Form-Event-Handling in Controller Class im Notes Client

- Ziel ist es, möglichst wenig Code in der Maske selber zu haben.



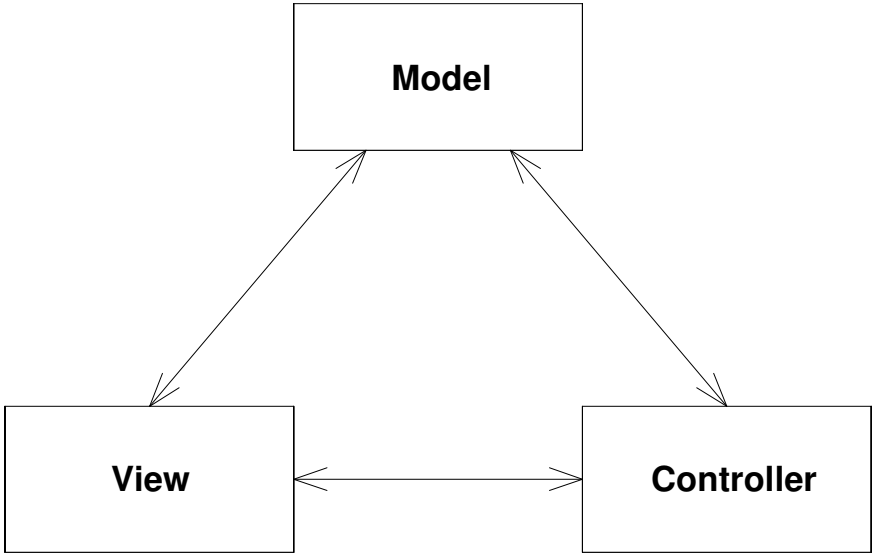

```
Sub Queryopen(Source As Notesuidocument, Mode As Integer, Isnewdoc As Variant,
Continue As Variant)
        Call CreateProjectController(source, mode, isNewDoc, continue)
End Sub
```

Zuordnung Event-Handling

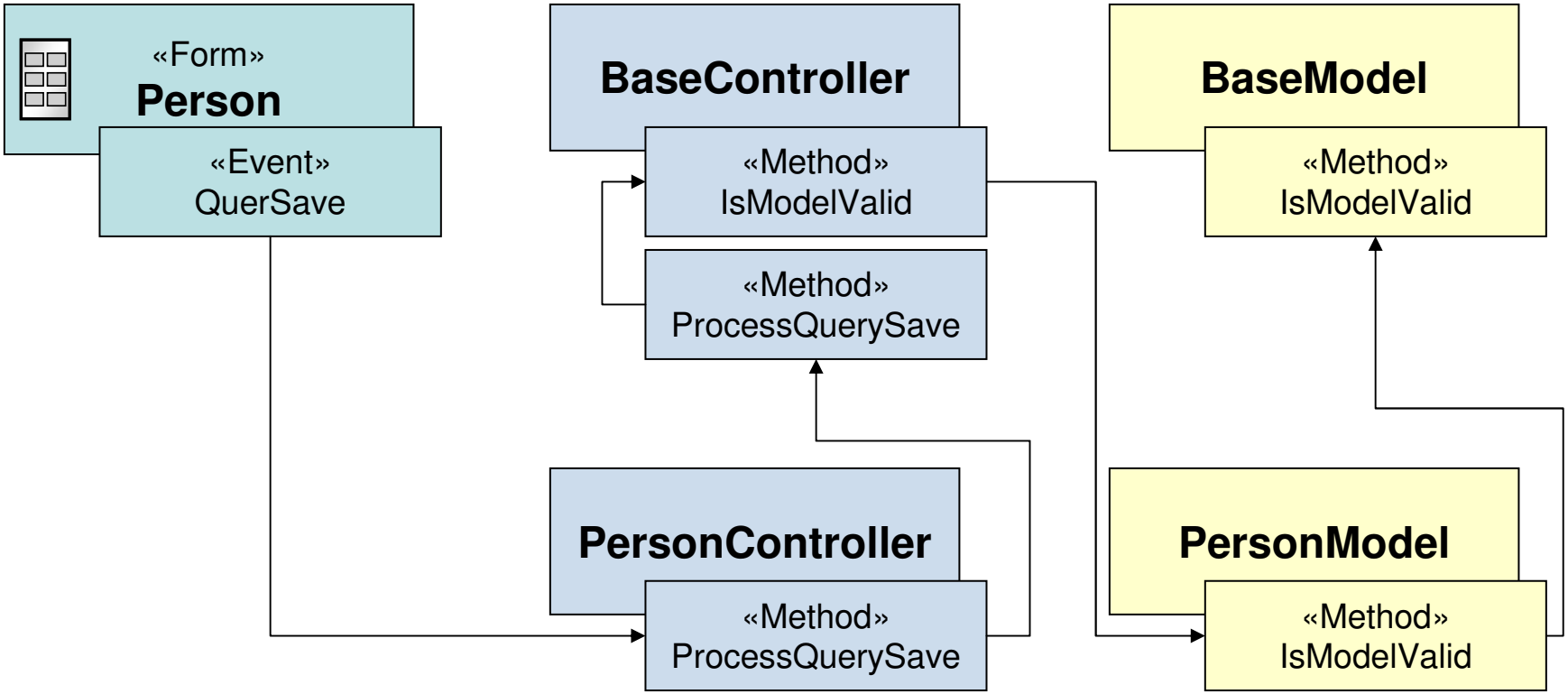
```
Public Sub ProcessQueryOpen(source As NotesUIDocument, mode As Integer,
    isNewDoc As Variant, continue As Variant)

    [...]
        ' register all event handlers
        On Event PostOpen From currentUIDoc Call processPostOpen
         [...]
        On Event QuerySave From currentUIDoc Call processQuerySave
         [...]
        On Event QueryClose From currentUIDoc Call processQueryClose
    [...]

End Sub ' BaseController.ProcessQueryOpen
```

Agenda

- Vorstellung
- Model View Controller Pattern
- Basisklassen
- Eingabevalidierung Notes Client
- Eingabevalidierung Web Browser im Backend
- Eingabevalidierung Web Browser im Frontend
- Zusammenfassung
- Fragen & Antworten



Eingabevalidierung im Notes Client

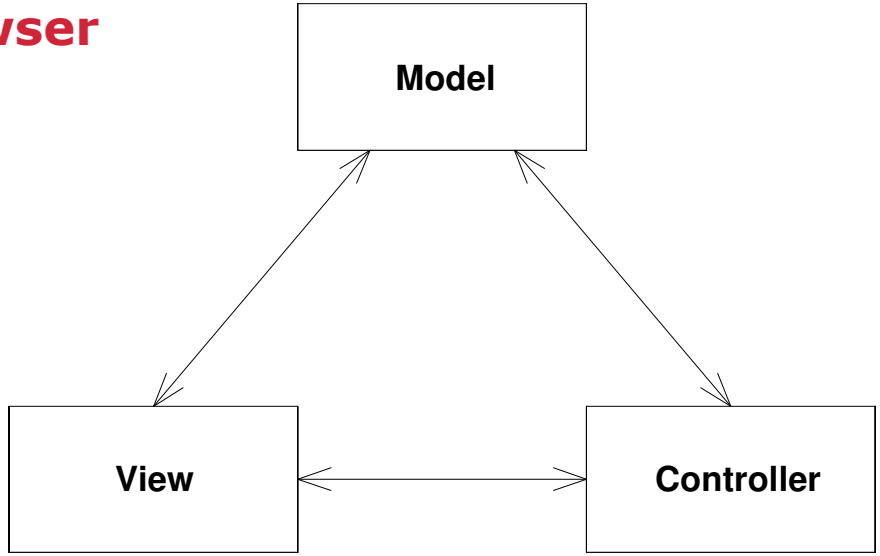


Demo Beispiel-Anwendung

«Form» Person – Notes Client

Agenda

- Vorstellung
- Model View Controller Pattern
- Basisklassen
- Eingabevalidierung Notes Client
- **Eingabevalidierung Web Browser im Backend**
- Eingabevalidierung Web Browser im Frontend
- Zusammenfassung
- Fragen & Antworten

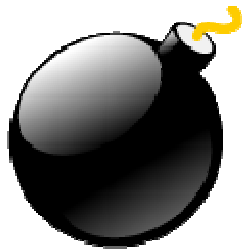


Warum Eingabevalidierung im Backend?

- Um Roundtrips zum Server zu vermeiden, sollte zunächst im Frontend via JavaScript die Eingabe validiert werden.

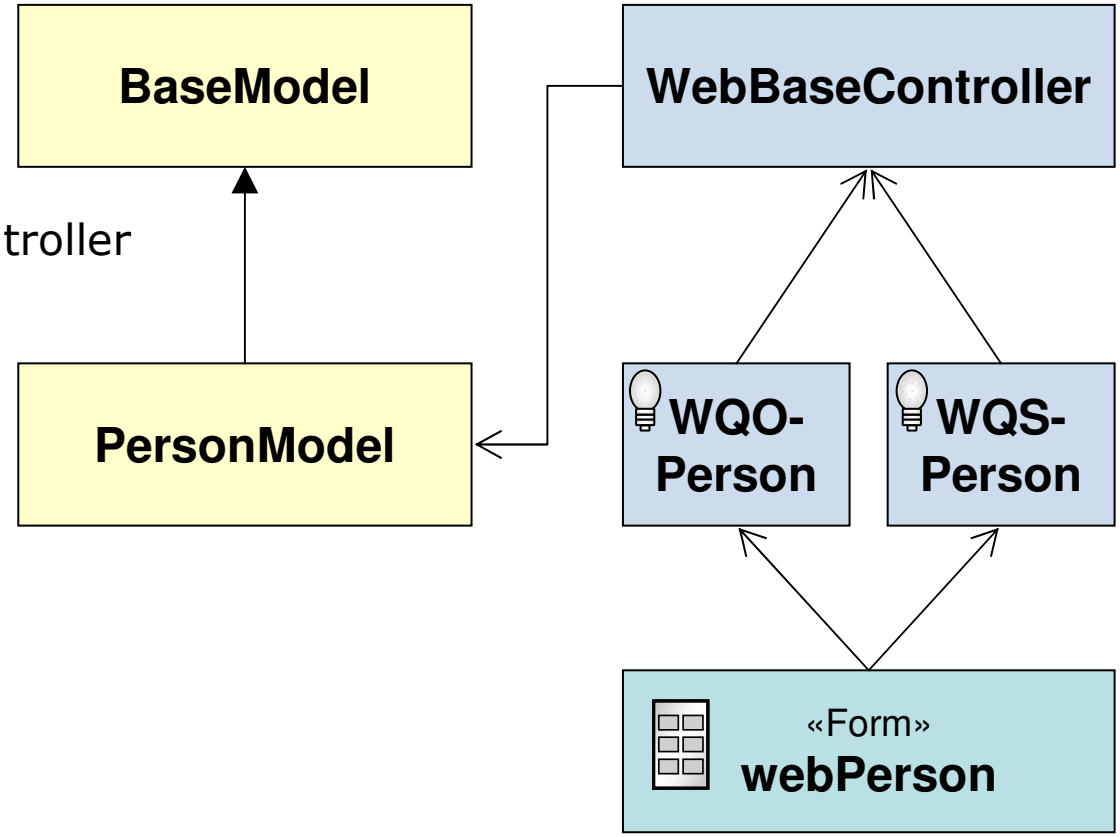
- Anwender können JavaScript ausgeschaltet haben.
- Mit Browsererweiterungen (Greasemonkey u.ä.) kann der Anwender eigenes JavaScript zur Ausführung bringen und somit die Security aushebeln.

- **Deshalb immer auch eine Eingabevalidierung im Backend vornehmen.**

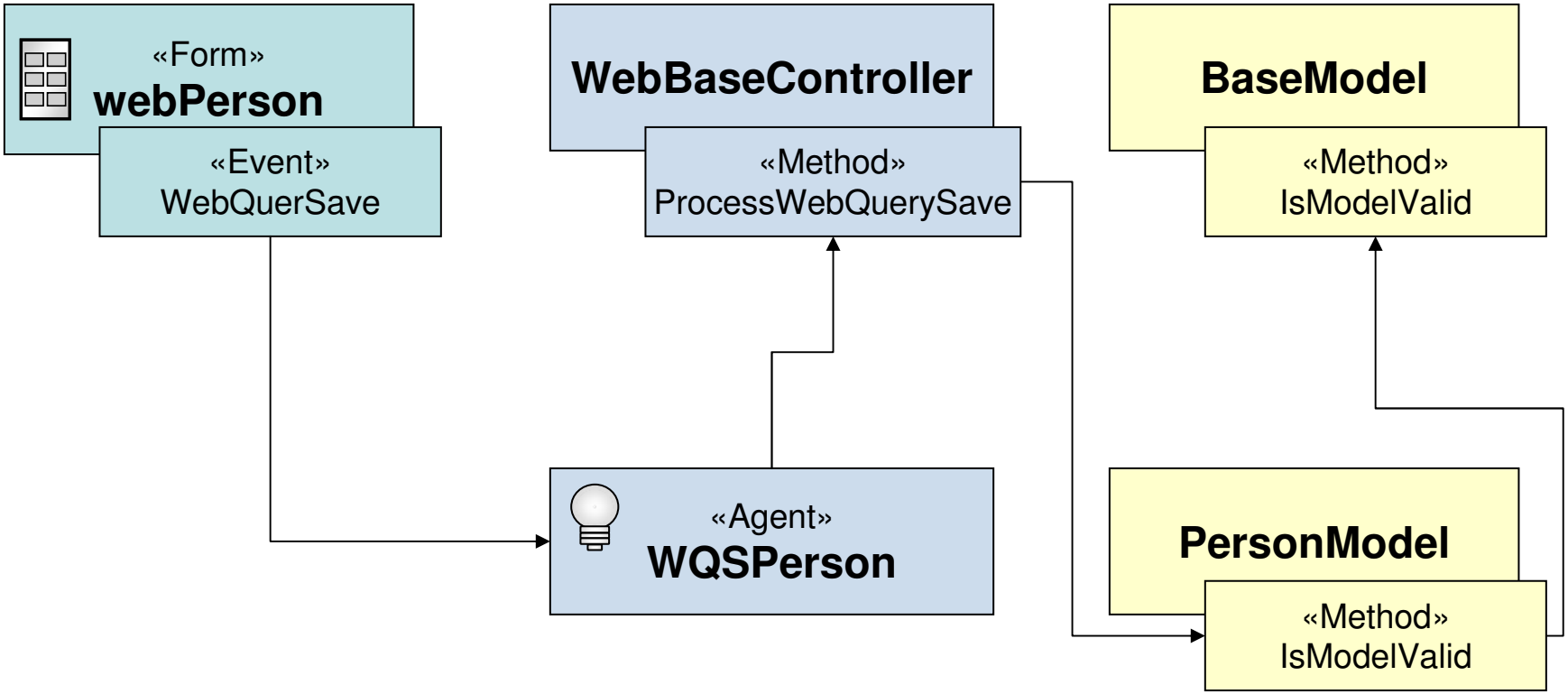


Backend Web Events

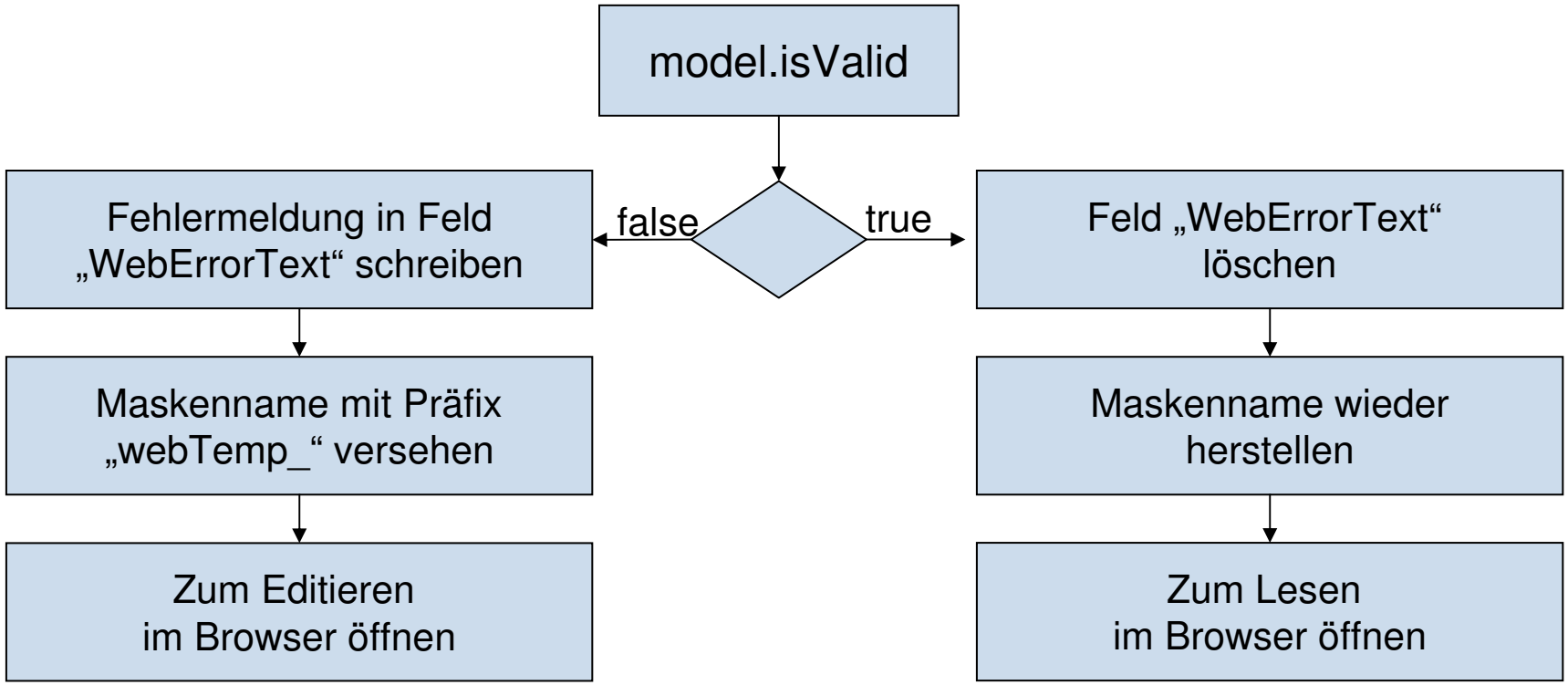
- Fachliche Aspekte in «Class» PersonModel
- Verbindung zur Maske für den Web Browser über «Class» WebBaseController
- Aufruf
  - «Agent» WQOPerson (WebQueryOpen)
  - «Agent» WQSPerson (WebQuerySave)



Eingabevalidierung im Browser via Backend



Ablauf ProcessWebQuerySave

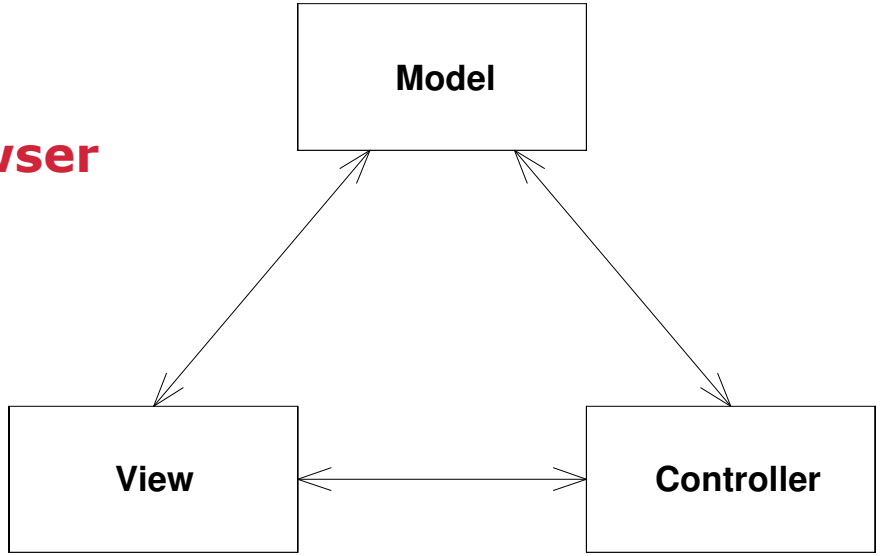


Demo Beispiel-Anwendung

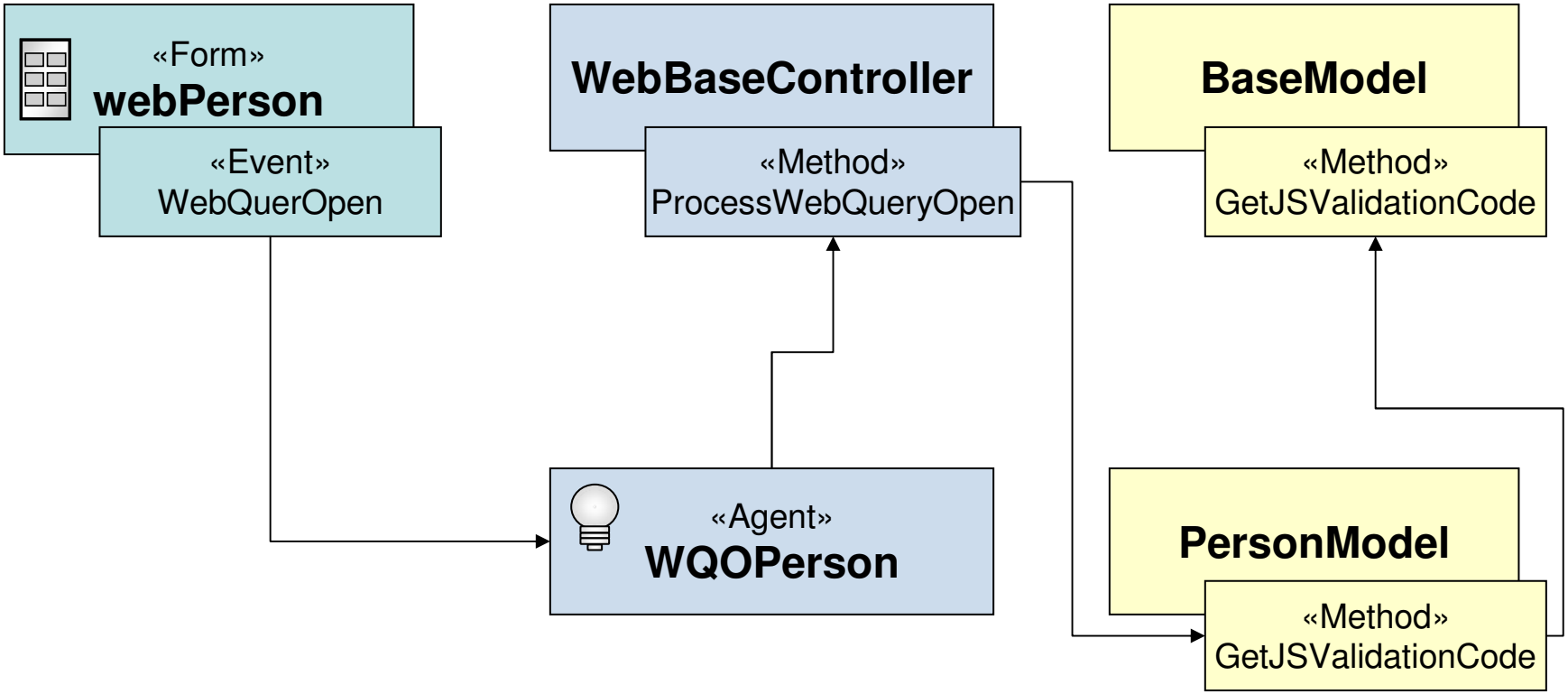
«Form» Person – Web Browser Validierung via Backend

Agenda

- Vorstellung
- Model View Controller Pattern
- Basisklassen
- Eingabevalidierung Notes Client
- Eingabevalidierung Web Browser im Backend
- **Eingabevalidierung Web Browser im Frontend**
- Zusammenfassung
- Fragen & Antworten



JavaScript aus «Class» BaseModel generieren

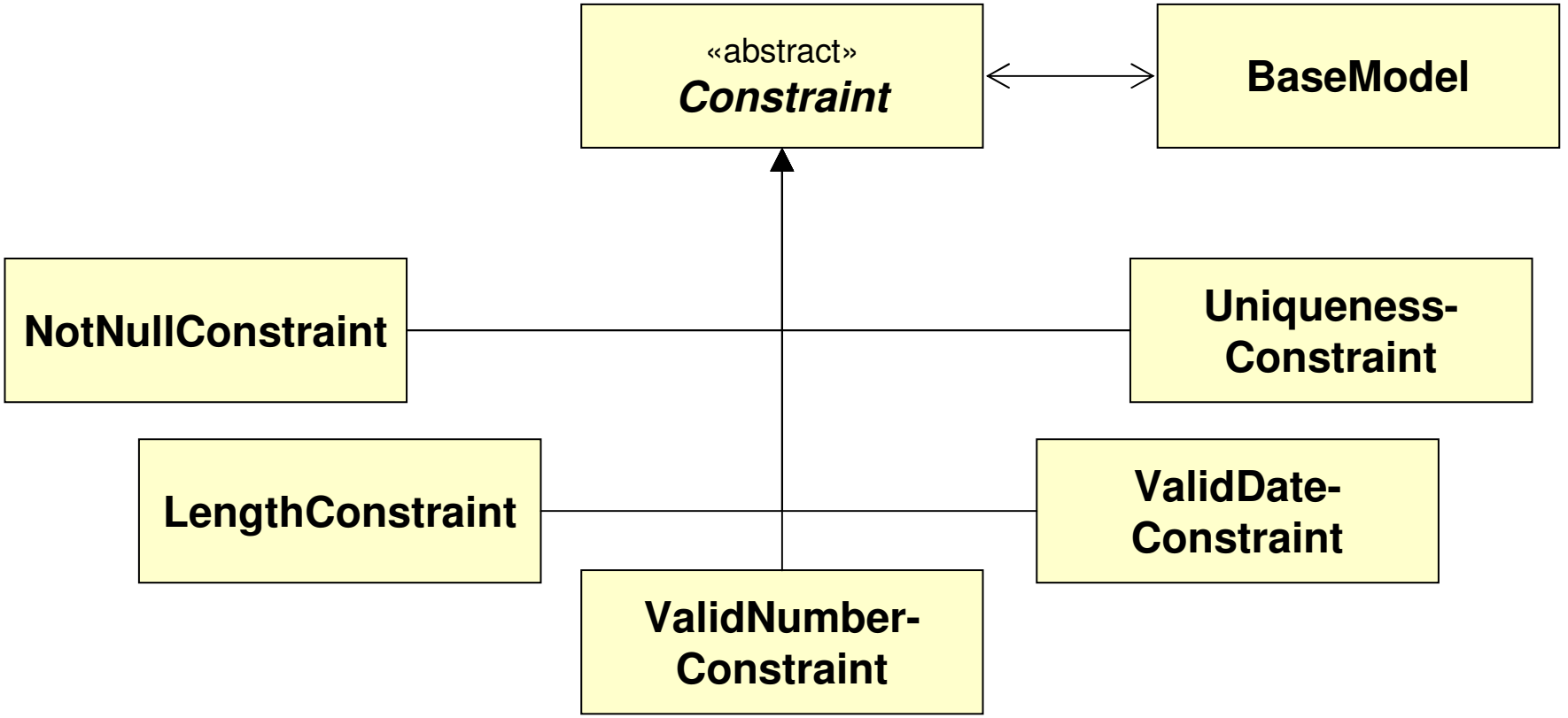


Problem: Zwei Implementierungen für gleiche Überprüfung

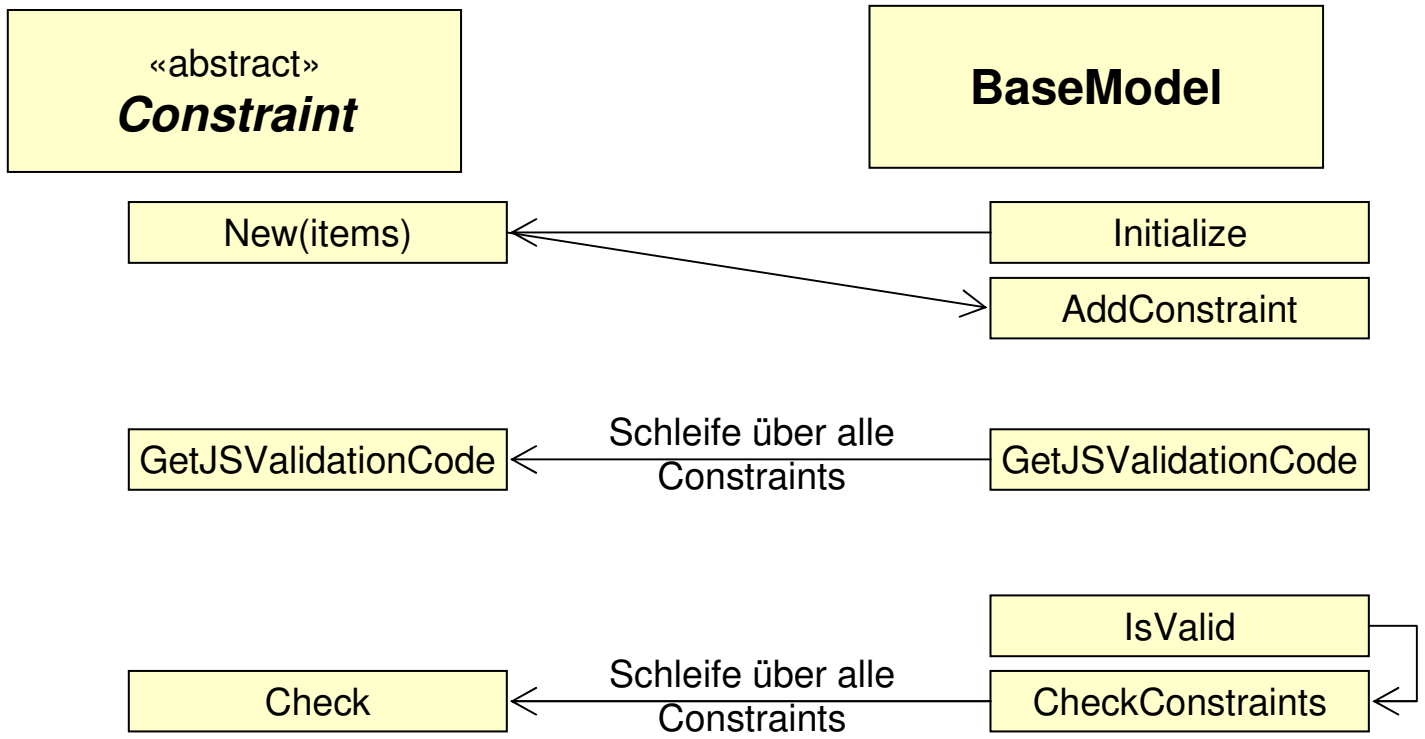
- Doppelte Umsetzung der fachlichen Anforderung
  - LotusScript für Notes Client und Web Browser Backend
  - JavaScript für Web Browser Frontend

- Lösung: Einführung einer «Class» Constraint zur Abdeckung der Standardfälle

«Class» Constraint



Zusammenspiel «Class» Constraint und «Class» BaseModel



Demo Beispiel-Anwendung

«Form» Person – Web Browser Validierung via Frontend

JavaScript für «Class» UniquenessConstraint

- Überprüfung auf Eindeutigkeit eines Schlüssels

- Parameter Feldname, Ansichtsname, Fehlertext, EvaluateFehlertext
- EvaluateFehlertext steuert, ob der Fehlertext Formelsprachenausdrücke enthält, die ausgewertet werden müssen

- Überprüfung in JavaScript erfordert Zugriff auf den Server
- Verwendung von AJAX-Techniken

Zwei Ansätze – Erster Ansatz: ReadViewEntries

- Wenn die Rückgabemeldung keinen Formelsprachenausdrück beinhaltet.
- Mit ?ReadViewEntries die Ansicht als XML zurück geben
- Mit &startkey={Schlüssel} nach dem Schlüssel suchen
- Mit &count=1 auf einen Rückgabewert beschränken
- Im zurück gelieferten XML die UniversalID überprüfen

ReadViewEntries

```
<viewentries toplevelentries="4">
    <viewentry position="1" unid="D98AAF8256C35239C125739A002F6D3A"
    noteid="902" siblings="4">
        <entrydata columnnumber="0" name="$1">
            <text>Hort, Bernd</text>
        </entrydata>
        <entrydata columnnumber="1" name="$3">
            <text>4711</text>
        </entrydata>
        <entrydata columnnumber="2" name="$4">
            <text>Bernd Hort/assono</text>
        </entrydata>
    </viewentry>
</viewentries>
```

Zwei Ansätze – Zweiter Ansatz: Agent WebCheckUniqueness

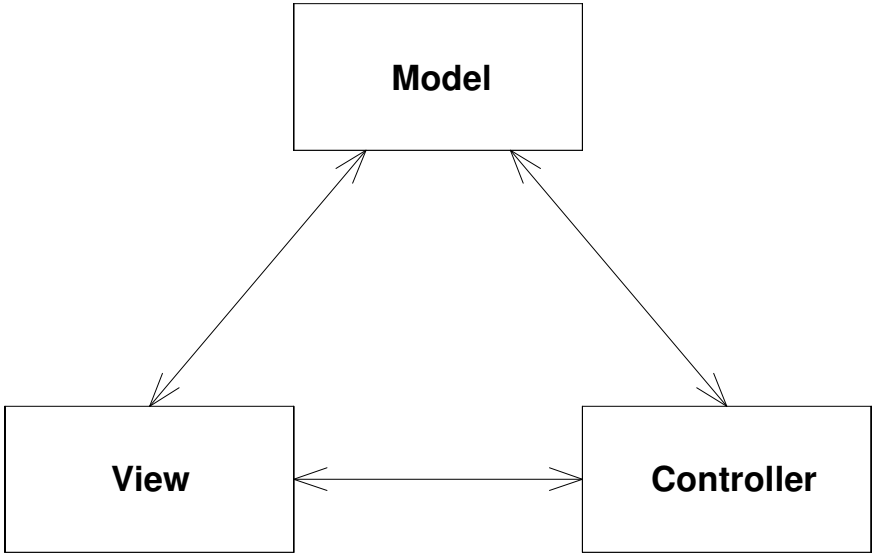
- Wenn die Rückgabemeldung einen Formelsprachenausdrück beinhaltet.
- Mit checkUniqueness?OpenAgent den Agenten aufrufen
- Mit &key={Schlüssel} nach dem Schlüssel suchen
- Mit &view={Ansichtsname} die Ansicht angeben
- Mit &unid={UniversalID} das aktuelle Dokument bestimmen
- Mit &msg={Rückgabemeldung} die Meldung übergeben
- Wenn der Schlüssel eindeutig wird „unique" zurück gegeben.
- Ansonsten der ausgewertete Formelsprachenausdruck

Demo Beispiel-Anwendung

«Form» Person – Web Browser
CheckUniqueness

Agenda

- Vorstellung
- Model View Controller Pattern
- Basisklassen
- Eingabevalidierung Notes Client
- Eingabevalidierung Web Browser im Backend
- Eingabevalidierung Web Browser im Frontend
- **Zusammenfassung**
- Fragen & Antworten

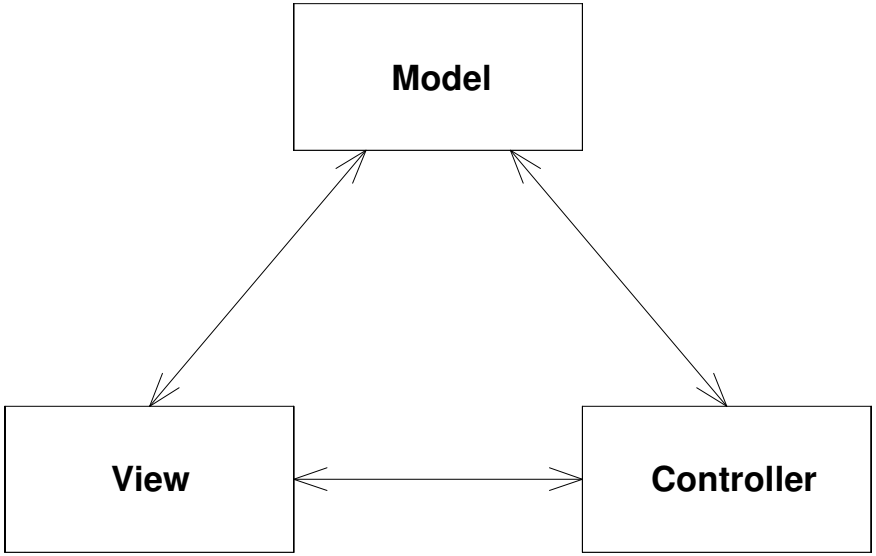


Zusammenfassung

- Die Kapselung aller fachlichen Aspekte in einer Klasse erhöht die Wartbarkeit, weil der relevante Code sich an einer Stelle befindet
- Der gleiche LotusScript-Code kann sowohl im Frontend als auch im Backend verwendet werden
- Für Routineüberprüfungen kann der JavaScript-Code automatisch generiert werden
- Effiziente Kombination von JavaScript und LotusScript möglich
- Das Framework ist leicht zu erweitern

Agenda

- Vorstellung
- Model View Controller Pattern
- Basisklassen
- Eingabevalidierung Notes Client
- Eingabevalidierung Web Browser im Backend
- Eingabevalidierung Web Browser im Frontend
- Zusammenfassung
- **Fragen & Antworten**



Zum guten Schluss…

- **Fragen?**
  - jetzt stellen oder später
    - E-Mail: bhort@assono.de
    - Telefon: 04101 / 487 47

- Folien und Beispiele unter http://www.assono.de/blog.nsf/d6plinks/EntwicklerCamp2008

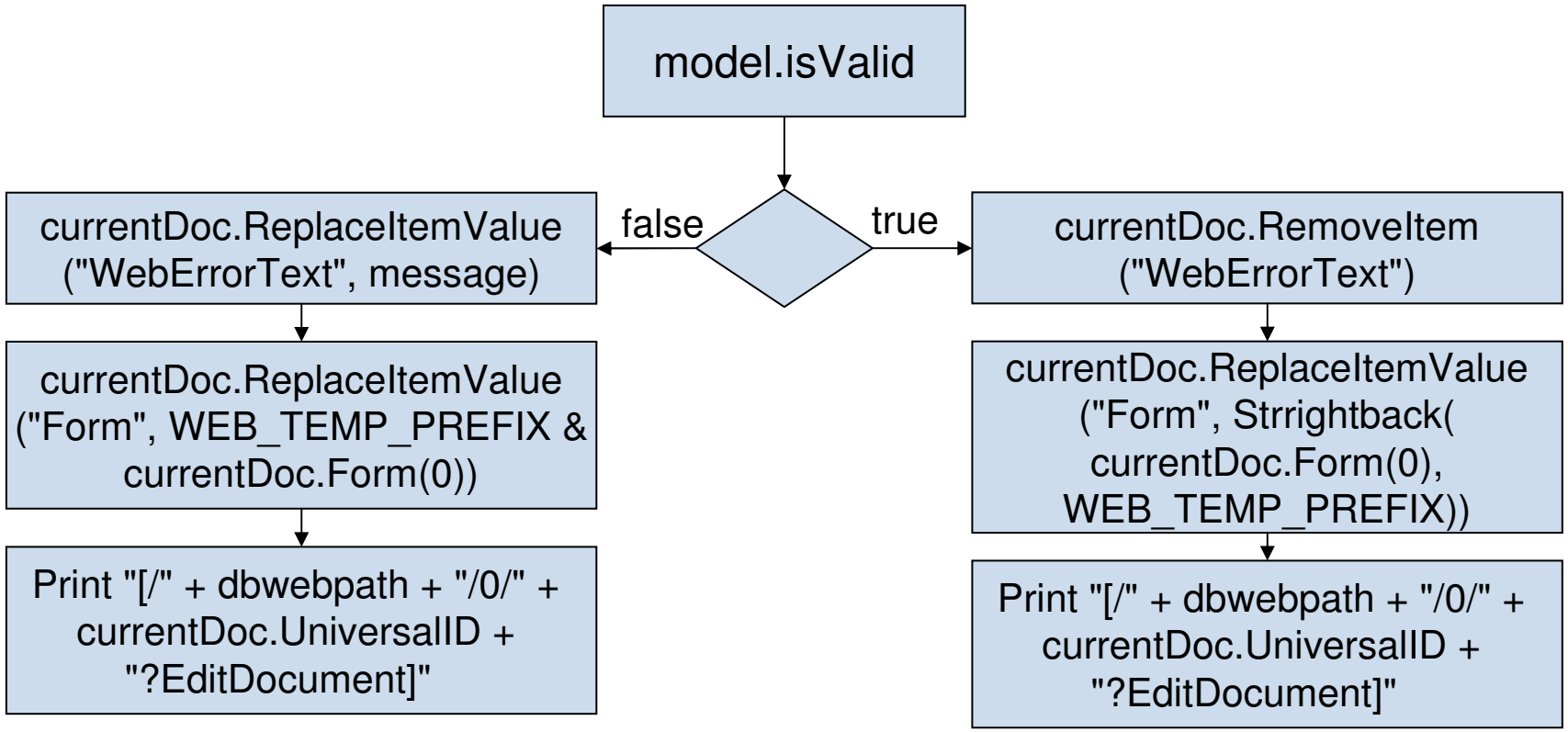
- In eigener Sache – wir suchen Verstärkung:

  IT-Berater (m/w)

  Details unter http://www.assono.de → Jobs

Folienbackup

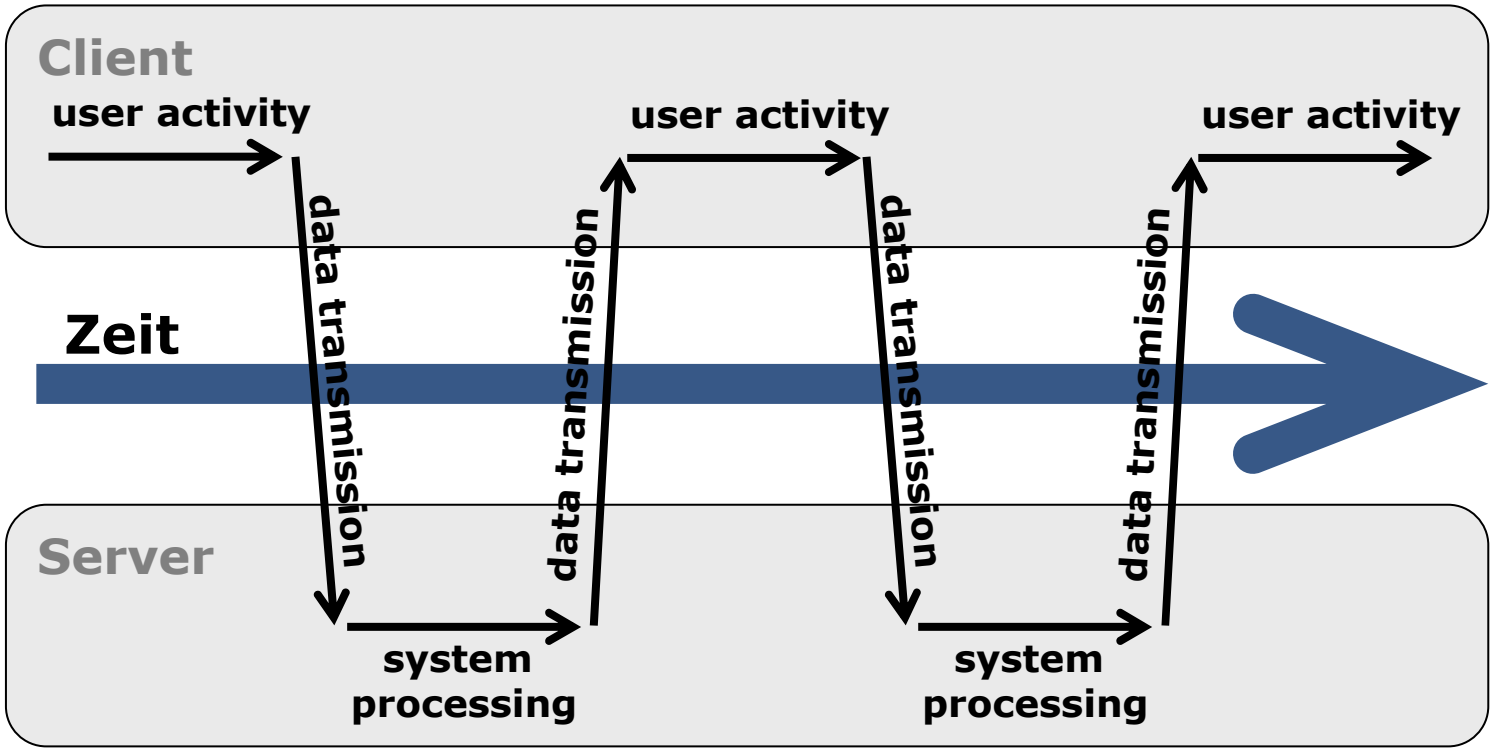
Ablauf ProcessWebQuerySave



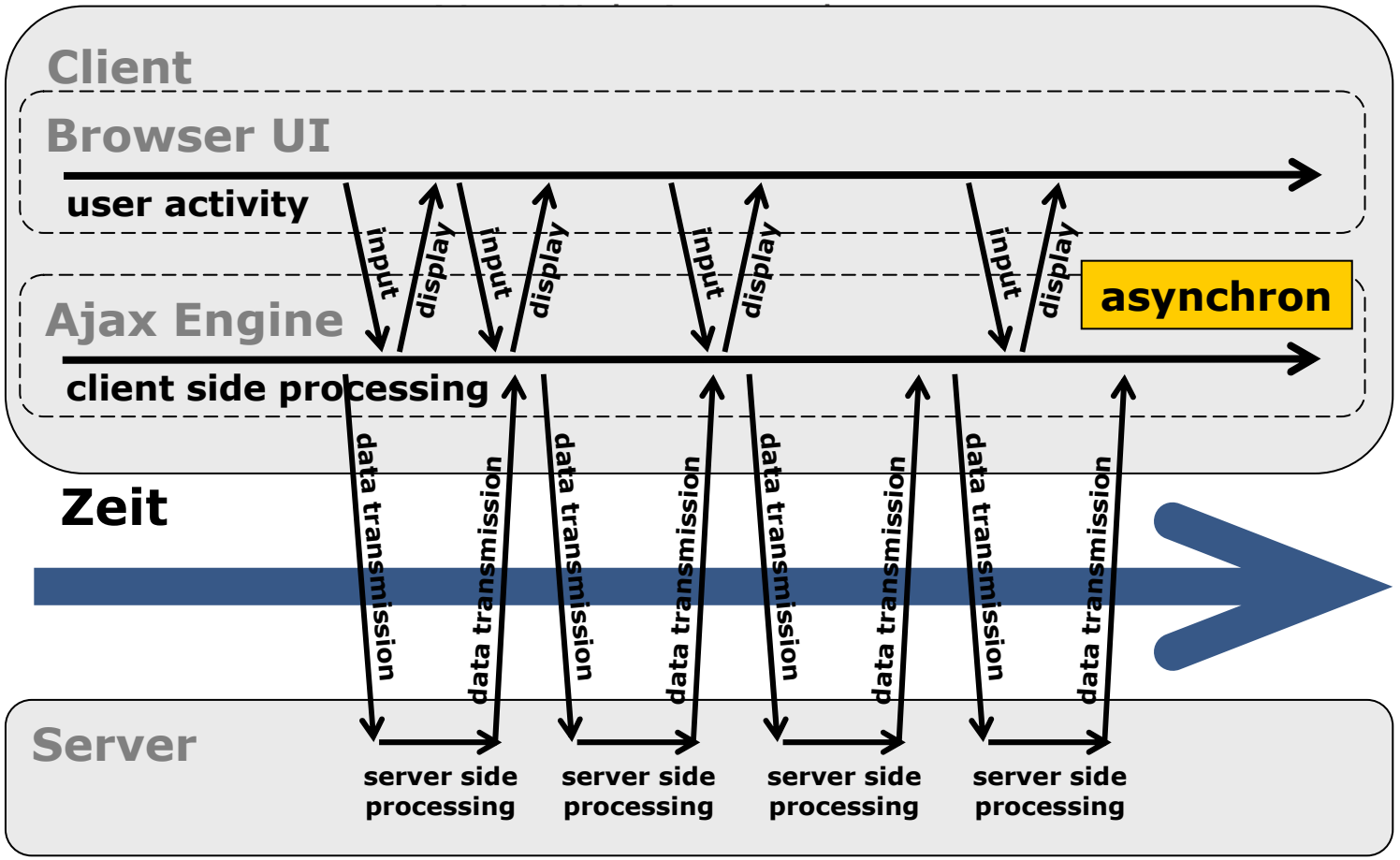
Ajax

- **Ajax** ['eɪdʒæks] ist ein Akronym für die Wortfolge **A**synchronous **J**avascript **a**nd **X**ML
- Keine neue Technology
- Dynamische Anpassung einer bereits geladenen Website mittels Javascript
- Grundlegende Änderung des Web Paradigmas

Klassische Web Anwendung



http://www.adaptivepath.com/publications/essays/archives/000385.php



http://www.adaptivepath.com/publications/essays/archives/000385.php