DATA-DRIVEN ACTIONS FOR NETWORK FORMS

Inventors: Amol S. Kelkar, Redmond, WA (US); Danny van Velzen, Redmond, WA (US); David Airapetyan, Kirkland, WA (US); Jonathan E. Rivers-Moore, Bellevue, WA (US); Ranjan Aggarwal, Redmond, WA (US)

Correspondence Address: MICROSOFT CORPORATION ONE MICROSOFT WAY REDMOND, WA 98052 (US)

Assignee: Microsoft Corporation, Redmond, WA (US)

Appl. No.: 12/694,836

Filed: Jan. 27, 2010

Related U.S. Application Data

Division of application No. 11/095,254, filed on Mar. 30, 2005, now Pat. No. 7,673,228.

Publication Classification

Int. Cl.
G06F 17/21 (2006.01)

U.S. Cl. 715/223; 715/221; 715/230

ABSTRACT

Systems and/or methods are described that enable a data-driven action associated with altering a data instance of a network form without altering and/or having access to the data instance. These systems and/or methods, in one embodiment, render a result of a data-driven action for a network form without transforming the network form's data instance. In another embodiment, the systems and/or methods map a data-driven action to a view-centric logical representation of the electronic form.

100

106

104

102

108

110

112

114

116

118
COMPUTER-READABLE MEDIA NETWORK FORM DATA INSTANCE SCHEMA

PROCESSOR(S) 108

COMPUTER-READABLE MEDIA DATA-CENTRIC ACTIONS NETWORK BROWSER 110 EVENT LOG

VIEW TEMPLATE VIEW-CENTRIC ACTIONS ACTOR APPLICATION INITIAL DATA

Fig. 1
Fig. 2

202 Find Navigation Path Mapping Data-Driven Action To Data Instance

204 Build View Template Path Corresponding to Navigation Path

206 Replace Navigation Path With View Template Path

208 Annotate View Template
Merchant
Seller of Quality Products for Twelve Years

PURCHASE ORDER

Name: 
Phone: 

<table>
<thead>
<tr>
<th>Item Id.</th>
<th>Quantity</th>
<th>Unit Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Grand Total 0

Fig. 3
402 Receive Choice To Edit Network Form

404 Receive Information Sufficient To Display Network Form

406 Render Chosen Network Form

408 Receive Edit To Network Form

410 Determine If Action Associated With Edit

412 Determine Whether Action Affects View

414 Perform Action

416 Update Data For Form

418 Render Result(s)

Fig. 4
**Merchants**

Seller of Quality Products for Twelve Years

**PURCHASE ORDER**

<table>
<thead>
<tr>
<th>Item Id</th>
<th>Quantity</th>
<th>Unit Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1756</td>
<td>17</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Grand Total 0

**Fig. 5**
Merchant
Seller of Quality Products for Twelve Years

PURCHASE ORDER

Name: Dave
Phone: 425-555-1234

<table>
<thead>
<tr>
<th>Item Id</th>
<th>Quantity</th>
<th>Unit Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1756</td>
<td>17</td>
<td>299.99</td>
<td>5099.83</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Grand Total 5099.83

Fig. 6
DATA-DRIVEN ACTIONS FOR NETWORK FORMS

PRIORITY

[0001] This application is a divisional of and claims priority under 35 U.S.C. §121 to U.S. patent application Ser. No. 11/065,254 filed Mar. 30, 2005, the disclosure of which is incorporated by reference herein in its entirety.

BACKGROUND

[0002] Electronic forms are commonly used to collect information. Electronic forms may be used locally or over a communication network, such as an intranet or the Internet. For electronic forms used locally, a user’s computer locally accesses view information and data information about an electronic form. With this view information, the user’s computer may enable the user to view and enter data into the electronic form. With this data information, the user’s computer may enable data actions for the electronic form, like validating data entered into a field, adding data from multiple fields and populating the result into another field, and the like.

[0003] For electronic forms used with a network, a user’s computer communicates with a network computer to enable use of the electronic form. In so doing, often the user’s network browser receives view information but not data information from the network computer. This view information may be used by the user’s network browser to permit the user to view and enter information into the electronic form.

[0004] To enable data actions for the electronic form, the user’s network browser often needs to communicate with the network computer for each data action. A user may, for instance, enter a price for a product in one data-entry field of a network form and a quantity of these products into another data-entry field. A data action may calculate a total price by multiplying the product’s price by its quantity. To present this total price in the form, often the user’s network browser sends the price and the quantity to the network computer and waits for the network computer to calculate the total, create new view information for the entire form but now including this total, and send it to the network browser. The network browser may then present the total to the user based on this new view information.

[0005] Communicating with the network computer, however, may slow the user’s network browser enough to negatively affect the user’s editing experience. In some cases, the network computer may take an appreciable amount of time to receive the request from the browser, perform the data action, and send new view information to the browser. Also, communicating with the network computer may be slow because of the network or how the browser communicates with it (e.g., with a dial-up modem), thereby potentially affecting a user’s editing experience.

[0006] In addition, having the network computer perform data actions expends some of the network computer’s resources. Expending these resources may slow or inhibit the network computer’s ability to service other users.

[0007] Accordingly, this invention arose out of concerns associated with performing data actions for network forms.

SUMMARY

[0008] Systems and/or methods (“tools”) are described that enable a data-driven action associated with altering a data instance of a network form without altering and/or having access to the data instance.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 illustrates an exemplary operating environment in which various embodiments can operate.

[0010] FIG. 2 is a flow diagram of an exemplary process for creating a view-centric data-driven action.

[0011] FIG. 3 illustrates an exemplary view of a network form.

[0012] FIG. 4 is a flow diagram of an exemplary process for performing a data-driven action associated with altering a data instance of a network form and/or rendering a result of the data-driven action.

[0013] FIG. 5 illustrates the view of FIG. 3 with edits.

[0014] FIG. 6 illustrates the view of FIG. 5 with an additional edit and results from performing a data-driven action.

[0015] The same numbers are used throughout the disclosure and figures to reference like components and features.

DETAILED DESCRIPTION

Overview

[0016] The following disclosure describes one or more tools enabling a data-driven action associated with altering a data instance of a network form without altering and/or having access to the data instance. In one embodiment, a tool performs a data-driven action without local access to a network form’s data instance and without communicating non-locally to access the data instance, such as by a roundtrip with a network computer having the data instance.

[0017] In another embodiment, a tool renders a result of a data-driven action for a network form without transforming the network form’s data instance. The tool may do so without necessitating access to and/or an alteration of the data instance. This may permit a user to edit a control in a network form and view the results of a data-driven action driven by the user’s edit without requiring that the tool access or alter the network form’s data instance. This may permit a user to avoid the form and view the result of the action driven by that edit without waiting for a roundtrip to a network computer having the data instance. Not having to perform a roundtrip may improve a user’s editing experience by making it smoother, faster, and/or more seamless. Not having to perform a roundtrip to view an action’s result may also reduce the resources needed by the network computer servicing the network form.

[0018] In still another embodiment, a tool builds a relation mapping a data-driven action to an electronic form’s view template. The tool may do so by transforming an action’s data-centric map of an electronic form’s data instance to a view-centric map of the electronic form’s view template. This view-centric map may be used, for example, to enable a tool to perform the data-driven action without needing access to the network form’s data instance.

Exemplary Operating Environment

[0019] Before describing the tools in detail, the following discussion of an exemplary operating environment is provided to assist the reader in understanding where and how the tools may be employed. The description provided below constitutes but one example and is not intended to limit application of the tools to any one particular operating environment.

[0020] FIG. 1 illustrates one such operating environment generally at 100 comprising a computer 102, a communication network 104, and a network computer 106. The computer is shown comprising one or more processors 108 and computer-readable media 110. The processor(s) are capable of accessing and/or executing the computer-readable media. The computer-readable media comprises a network browser.
The dashed lines between the received media and media of the network computer show that these indicated media may be downloaded by the computer from the network computer.

The network computer is shown comprising computer-readable media 118. This computer-readable media comprises a network form 120 comprising a data instance 122, a schema 124 governing the network form, and data-centric data-driven actions 126. The network form's data instance is a particular data instance of the network form; thus, if the structure or data of the network form is changed (such as a user entering information into a view of the network form), the network form's data instance is altered at some point in time to reflect the change. The data-centric data-driven actions are actions driven by data changes in the data instance and are mapped to the data instance.

The view information is information sufficient for the network browser to display an editable view of the network form. This view information may also be sufficient for the network browser and/or the actor application to display an alteration to the editable view resulting from execution of an action. In one embodiment, the view information comprises HyperText Machine Language (HTML) resulting from a transformation declared in XML Stylesheet Language Transformation (XSLT) of an Extensible Markup Language (XML) embodiment of the data instance. This HTML is usable by the network browser to display the network form. Also in this embodiment, the network form's schema comprises XML Schema (XSD). This schema may comprise data-centric actions to validate the network form.

The view template comprises a view-centric logical representation of the network form. The view template's structure is, in one embodiment, one in which view information may be mapped sufficient for the actor application and/or the browser to build a view of the network form. The view template's structure may also be, in another embodiment, be one in which actions are related sufficient for the actor application to perform actions in response to a user's interaction with a portion of the view template.

The view-centric actions comprise data-driven actions, which may be identified with or mapped to a position or node in the view template.

The actor application, in conjunction with or separate from the network browser, is capable of performing data-driven actions without altering and/or accessing the network form's data instance 122.

The initial data comprises stored data, such as default data. Default data may comprise, for example, values for simple fields (like date, currency, and name fields), and a number of repetitions for lists, tables, and rows. Other stored data may comprise data previously entered into the network form, such as when a form is altered, submitted, and later re-opened for viewing or further editing.

Various embodiments of these elements, and particularly an exemplary process for creating view-centric data-driven actions, are set forth in greater detail below. After describing this process, exemplary tools enabling these and other actions are described.

Data-Driven Actions

Many data-driven actions for electronic forms are related to a form's data instance, such as by being mapped to the form's data instance. But this relation may require accessing the data instance, which may be undesirable for electronic forms accessed over a network.

An exemplary process 200 for creating a view-centric data-driven action is shown in FIG. 2 and described below. This process creates a view-centric data-driven action by transforming an existing data-centric data-driven action having a relation to a network form's data instance to a data-driven action having a relation to the network form's view template. Process 200 is illustrated as a series of blocks representing individual operations or acts performed by elements of the operating environment 100 of FIG. 1, such as actor application 134. This and other processes described herein may be implemented in any suitable hardware, software, firmware, or combination thereof; in the case of software and firmware, these processes may represent a set of operations implemented as computer-executable instructions stored in computer-readable media 118 and executable by processor(s) of network computer 106 and/or in computer-readable media 110 and executable by processors 108.

To illustrate the process 200 a purchase order 300 showing an example of network form 120 is set forth in FIG. 3. In the purchase order, a view of controls are shown, here for a name field 302, a phone field 304, item fields 306a and 306b, quantity fields 308a and 308b, unit price fields 310a and 310b, total fields 312a and 312b, and a grand total field 314. The purchase order's schema, data instance, and view template are helpful in describing the process and so are set forth below.

The schema governing the data instance of the network form may be represented as:

```
root
  name
  phone
  orders
    order
      itemid
      quantity
      unitPrice
      total
```

The data instance governed by this schema may be represented as:

```
<root>
  <name/>
  <phone/>
  <orders>
    <order>
      <itemid/>
      <quantity/>
    </order>
  </orders>
</root>
```
And, the view template for this form may be represented as:

V1 - root
T1 - name
T2 - phone
R1 - orders/order
T1 - itemid
T2 - quantity
T3 - price
T4 - total
T3 - total

Returning to FIG. 2, at block 202 a tool (e.g., actor application 134) finds a relation mapping a data-driven action (e.g., one of the data-centric data-driven actions 126 of FIG. 1) to data instance 122. This relation may comprise a navigation path, such as an XPath expression.

Continuing the illustrated embodiment, assume that the data instance has two data-centric data-driven actions. These actions may be represented as:

1. target="root/orders/order/total"
   expression="../quantity * ..unitPrice"

2. target="root/total"
   expression="sum(../orders/order/total)"

The first action is structured to multiply the data in the data instance’s quantity node by the data in the data instance’s unit price node. The second action is structured to sum all of the data in the data instance’s total nodes that are child nodes to the orders node.

The navigation paths may be extracted from these data-centric data-driven actions, which in this case are XPath expressions. The tools extract the following XPaths:

1. /root/orders/order/total
2. /root/total

As shown, these XPaths map to the data instance. The first maps to the two total nodes that are children to the “orders” node (rendered as the total fields 312a and 312b in FIG. 3). The second maps to the total node that is a child of just the root node (rendered as the grand total field 314 in FIG. 3).

At block 204, the tool builds a view template path corresponding to the navigation path. The tool may correlate the navigation path to the view template to transform the navigation path into the view template path. The tool may also compare the portion of the data instance mapped by the navigation path to the structure of the view template. If the tool finds a node of the view template that corresponds to the mapped portion of the data instance, the tool may build a view template path mapping this node.

Continuing the illustrated embodiment, the tool determines which view template node corresponds to the data instance node mapped by the navigation path. The view template may be represented as:

V1 - root
T1 - name
T2 - phone
R1 - orders/order
T1 - itemid
T2 - quantity
T3 - price
T4 - total
T3 - total

The tool determines that the data instance node mapped by the navigation path of “/root/orders/order/total”, which is:

V1/R1/T4

Similarly, the tool determines that the data instance node for the second data-driven action corresponds to:

V1/T3

Alternatively, the tool transforms the navigation path by determining the node of the view template corresponding to the schema node mapped by the navigation path. In some cases the structure of the schema governing the electronic form is more easily or accurately parsed, thereby making the tool’s determination easier or more accurate. As shown above, correlating the schema node of “/root/orders/order/total” may be correlated to:

V1 - root
R1 - orders/order
T4 - total

to find “V1/R1/T4” of the view template.

In some cases, no view template node corresponds with the data instance node mapped by the navigation path. One such case is where a data instance node to which an action is mapped is not capable of affecting the form’s view. If a node of a data instance (or schema) has a related action
that alters data in a node that is not viewable, for instance, then that action may not have a corresponding node in the view template.

If the tool determines that the mapped-to portion of the data instance does not have a corresponding node in the view template, the tool does not proceed. If it determines otherwise, it proceeds to block 206. In the illustrated embodiment, both of the data-centric data-driven actions are capable of affecting the view and have a corresponding node in the view template. The tool builds view template paths to these corresponding nodes.

At block 206, the tool replaces the navigation path mapping the data-driven action to the view instance with a view template path mapping the data-driven action to the view template.

Continuing the illustrated embodiment, the navigation paths for the data-driven actions are:

```
/root/orders/order/total
```

and

```
/root/total
```

The tools replace these navigation paths with view template paths. Thus, the data-centric data-driven actions are transformed to view-centric data-driven actions, such as:

```
target="V1/R1/T4" expression="../T2 *../T3 0061 and 0062 target="V1/T3' expression="sum (. . . /R1/T4)"
```

```
V1 - root
T1 - name
T2 - phone
R1 - orders/order
T1 - itemid
T2 - quantity

data_action =

{ target = ..T4 expression = Multiply (Select(..T2), Select(..T3))

T4 - total

```

This annotated view template indicates that data for the node “V1/R1/T2” (shown with the quantity field 308a) and “V1/R1/T3” (shown with the quantity field 310b) may be multiplied and the result placed in the “V1/R1/T4” node (shown with the total field 312a). The view template also indicates the same for each iteration of these nodes (e.g., fields 308b, 310b, and 312b).

The view template also indicates that data for each of nodes “V1/R1/T4” (shown as total fields 312a and 312b) may be summed and the result placed in another node, that of “V1/T3” (shown with grand total field 314).

Performing a Data-Driven Action For A Network Form

As part of an exemplary process 400 shown in FIG. 4, the tools perform data-driven actions associated with altering a data instance of a network form without altering and/or having access to the data instance. Also in this process, a tool renders a result of a data-driven action for a network form without transforming the network form’s data instance. Process 400 is illustrated as a series of blocks representing individual operations or acts performed by elements of the operating environment 100 of FIG. 1, such as actor application 134.

At block 402, network browser 112 receives a user’s choice to edit a network form. Assume, by way of example, that the user is browsing forms available over a network and selects to edit purchase order 300 of FIG. 3.

At block 404, the network browser receives information sufficient to enable computer 102 to display and receive edits to the network form. This information may comprise computer-readable media from network computer 106 of FIG. 1, such as view information 128, view template 130, view-centric actions 132, actor application 134, and initial data 136. With the view information and view template, the actor application and/or the network browser displays an editable view of the form.

In one embodiment, the actor application builds an editable view of the form by mapping pieces of the view information to the view template. These pieces are viewable pieces, such as HTML, which the actor application may concatenate into an overall view of the form. These view pieces may comprise, for instance, renderable HTML for data-entry fields, buttons, and other controls.

At block 406, the computer renders the network form chosen by the user. In the illustrated embodiment, the actor application renders the purchase order of FIG. 3.

At block 408, the network browser receives an edit to a control of the network form. The user may edit the form by entering data into a field, selecting a button, and the like.

Continuing the illustrated embodiment, the network browser receives data entered into name field 302. This data is pushed into the view, shown in FIG. 5.

At block 410, the actor application determines if an action is associated with the edit. In one embodiment, the actor application analyzes actions to determine if any are associated with (e.g., map to) the node edited by the user. In
another embodiment, the actor application navigates the view template to find actions associated with the edit (if there are any). If there are no data-driven actions associated with the edit, the actor application returns to block 408 to receive additional edits from the user. If there are data-driven actions, the actor application proceeds to block 412 (or skips to block 414).

[0075] Continuing the illustrated embodiment, the actor application navigates the view template annotated with actions (shown above) to determine if any action is associated with “V1/T1” node (the “name node”), rendered as the name field 302 in FIGS. 3 and 5. This node does not have an action associated with it. The actor application then returns to block 408. The network browser receives other edits from the user; entry of the user’s phone number into the phone field 304, shown in FIG. 5, and an item identifier into item node 306a, also shown in FIG. 5. These entries are also not associated with a data-driven action.

[0076] The user next enters a quantity into the quantity field 308a, received at block 408, and shown in FIG. 5. The actor application determines if an action is associated with this edit at block 410.

[0077] Continuing the illustrated embodiment, the actor application navigates the view template to determine if an action is associated with the received edit. The annotated view template may be represented as:

```xml
V1 - /root
T1 - name
T2 - phone
R1 - orders/order
T1 - itemid
T2 - quantity
data_action =
  {target = ..T4
   expression
   = Multiply(Select(/..T2), Select(/..T3))}
```

[0078] The actor application navigates this view template and determines that the following data-driven action is associated with the user’s edit to the quantity node:

```xml
data_action =
  {target = ..T4
   expression
   = Multiply(Select(/..T2), Select(/..T3))}
```

[0080] The actor application may determine if the action will affect the view by navigating the view template to determine which nodes of the view template may be changed by performing the action.

[0081] In the ongoing embodiment, the actor application determines that the data-driven action associated with the quantity node multiplies data in the “V1/R1/T2” node by the data in the “V1/R1/T3” node, and places this result in the target node “V1/R1/T4”. Thus, the data “$17” in quantity field 308a is multiplied by no data (zero) in the unit price field 310a, the result of which (“$0”) is to be placed in the total field 312a. In this case, the result of performing the action does not change the view. The total field 312a already has a zero. In this case the actor application may forgo performing the action associated with this quantity node and/or other actions, such as updating data for the form (i.e., do not perform block 416) and rendering the results of performing this action (i.e., do not perform block 418). If the actor application forgoes performing the action and/or these other actions, it returns to block 408.

[0082] Returning to block 408, the network browser receives another edit, this time entry of a unit price to the unit price field 310a. The price is shown in FIG. 6 at field 310a (“$299.99”).

[0083] The actor application then determines at block 410 that the action is associated with the edit. Continuing the illustrated embodiment, the actor application navigates the view template annotated with actions (shown above) and determines that an action is associated with “V1/R1/T3” node (the “price” node).

[0084] Navigating the view template given above, the actor application determines that the following data-driven action is associated with the user’s edit to the price node:

```xml
data_action =
  {target = ..T4
   expression
   = Multiply(Select(/..T2), Select(/..T3))}
```

[0085] The actor application then determines that the action will affect the view. It does so by navigating the view template to determine which nodes of the view template may be changed by performing the action.

[0086] In the ongoing embodiment, the actor application determines that the data-driven action associated with the quantity node multiplies data in the “V1/R1/T2” node by the data in the “V1/R1/T3” node, and places this result in the target node “V1/R1/T4”. Thus, the data “$17” shown in quantity field 308a is multiplied by “$299.99” shown in unit price
field 310a, the result of which may be different that the zero currently rendered in the total field 312a.

[0087] At block 414 the actor application performs the action without accessing and/or altering the form’s data instance. In this exemplary process, the actor application uses the view template rather than a data instance, thereby freeing the actor application from having to access or alter the form’s data instance. The actor application may perform the data-driven action, in some cases, without any non-local communication, thereby potentially improving the user’s editing experience. In one embodiment, the actor application performs the action by executing an expression and locating a node of the view template into which the result may be rendered.

[0088] In the ongoing embodiment, the actor application multiplies the data entered into the quantity field “17”, by the data entered into the unit price field “299.99”, for a result of “5099.83”.

[0089] At block 416, the actor application updates the data for the form. This data for the form may be locally stored, which may permit fewer roundtrips to the network computer. In the ongoing embodiment, the actor application updates the zero for the first total node (rendered as the total field 312a) with the result “5099.83”. The actor application may proceed directly to block 418 to render this result in the view, but may also first determine if this data change triggers any other actions. If so, the actor application may perform all of the triggered actions (some actions may trigger another action, which triggers another action, and so forth) before rendering the different actions’ results.

[0090] Similarly to block 408, 410, 412, and 414, the actor application may proceed to receive data (e.g., a result from a performed action), determine in the node into which the data is received is associated with an action, determine if that action may affect the view, and perform the action. Thus, the actor application in the ongoing embodiment receives the result of the action performed above to the total node. The actor application may then determine whether or not the node to which this “edit” is made (e.g., a result automatically received from performance of the action) has another action associated with it. If so, it may then determine whether or not performing that other action will affect the view; and if so, may then perform the other action.

[0091] In the ongoing embodiment, the actor application determines that the result for the total node has another associated action. This action may be represented as:

```
data_action = {
    target = T3
    expression = Sum(Select(../R1, T4))
}
```

[0092] Thus, the total node (“V1/R1/T4”) has an associated data-driven action. This action sums the data of the total node and all other total nodes. The result of this summation is then targeted for the “V1/T3” total node (rendered as the grand total field 314).

[0093] The actor application next determines that performing this summation action may affect the view. The actor application performs this action, summing the data of the first and second total nodes, which are “5099.83” and “0”, for a result of “5099.83”.

[0094] The actor application may record this data entered into and resulting from actions, such as in a locally stored hierarchically structured data tree or event log 114. The data tree, with the data and results of the ongoing embodiment, may be represented as:

```
(V1)
  (Dave, T1)
  (425-555-1234, T2)
  (R1, T3)
  (1756, T1)
  (17, T2)
  (299.99, T3)
  (5099.83, T4)
```

[0095] At block 418 the actor application and/or network browser renders the result of the data-driven action(s) performed without transforming the form’s data instance. The actor application may also, in another embodiment, render changes to the view without accessing or altering the data instance.

[0096] The actor application may keep track of which controls in the view need to be updated by marking nodes associated with these controls, and then rendering each of these nodes for new data once all of the actions have been performed (but with results being rendered).

[0097] Alternatively, the actor application may re-render all of the form, though this may take additional time or resources compared with selective re-rendering.

[0098] To selectively re-render parts of the view, the actor application may determine which of the pieces of information from view information 128 of FIG. 1 are associated with the nodes of the view template that have been changed, such as by marking these nodes dirty. For the above view template, the marked nodes are “V1/R1/T4” and “V1/T3”. The actor application may then push the results into these pieces of the view or re-render each of these pieces with the results added.

[0099] As shown in FIG. 6, the view of the purchase order shows the user the results of data-driven actions. These results, shown in the total field 312a and the grand total field 314, are accurate renderings of how the view may look if the data-driven actions were instead associated with the data instance. Thus, this view may be identical to the view shown by placing the results of these actions into a data instance, transforming the data instance, and rendering the transformation.

[0100] In some cases, however, a data-driven action may require communication across a network, such as to access a data instance for the form or a database having data needed to perform an action (e.g., validation of entered data against data in a database). In these cases, the actor application may access the network, thereby forgoing many of the blocks of process 400.

[0101] The actor application and/or network browser may record the user’s edits and/or the results of performing various actions. This record of edits and/or results may be sent to
network computer 106, such as when sending it will not appreciably hinder a user’s editing experience, or on submital by the user.

[0102] The network computer, responsive to receiving this event log, alters the data instance of the network form. It may do so infrequently, thereby potentially reducing the resources needed to service the network form.

[0103] For the ongoing illustration, the actor application records the user’s edits to the view into event log 114 and communicates this event log to the network computer. The network computer updates data instance 122 with information from the event log. The data instance represented above at the beginning of the description relating to FIG. 2 may be changed by the network computer using the event log, in this case to:

```xml
<root>
  <name>Dave</name>
  <phone>425-555-1234</phone>
  <orders>
    <order>
      <itemId>1756</itemId>
      <quantity>17</quantity>
      <unitPrice>299.99</unitPrice>
      <total>5099.83</total>
    </order>
    <order>
      <itemId>1756</itemId>
      <quantity>17</quantity>
      <unitPrice>299.99</unitPrice>
      <total>5099.83</total>
    </order>
  </orders>
</root>
```

CONCLUSION

[0104] Systems and/or methods are described that enable a data-driven action associated with altering a data instance of a network form without altering and/or having access to the data instance. Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

1. One or more computer-readable media having computer-readable instructions therein that, when executed by a network computing device, cause the network computing device to perform acts comprising:
   - obtaining a first relation for a data-driven action that maps the data-driven action to a data instance of a network form maintained by the network computing device;
   - acquiring a second relation to map the data-driven action to a view-centric logical representation of the electronic form;
   - sending the view-centric logical representation of the network form to a client device to enable the client device to render a result of the data-driven action

2. One or more computer-readable media as recited in claim 1, further having computer-readable instructions therein that, when executed by the network computing device, cause the network computing device to perform acts comprising generating the view-centric logical representation based on the first relation and the second relation.

3. One or more computer-readable media as recited in claim 2, wherein the view-centric logical representation is configured to enable the client device to render a result of the data-driven action without accessing or altering the network form’s data instance that is maintained by the network computing device.

4. One or more computer-readable media as recited in claim 1, further having computer-readable instructions therein that, when executed by the network computing device, cause the network computing device to perform acts comprising:
   - receiving an event log that records the result of the data-driven action, the event log received by the network computing device from the client device subsequent to rendering of the result of the data-driven action by the client device; and
   - updating the network form’s data instance to reflect changes based on the result of data-driven action recorded in the event log.

5. One or more computer-readable media as recited in claim 1, further having computer-readable instructions therein that, when executed by a network computing device, cause the network computing device to perform acts comprising:
   - receiving the result of a data-driven action that is rendered at the client device using the view-centric logical representation of the network form; and
   - modifying the network form’s data instance to reflect the result of the data-driven action that is rendered at the client device using the view-centric logical representation.

6. One or more computer-readable media as recited in claim 1, further having computer-readable instructions therein that, when executed by the network computing device, cause the network computing device to perform acts comprising causing replacement of the first relation with the second relation to generate the view-centric logical representation of the network form.

7. One or more computer-readable media as recited in claim 1, wherein the first relation comprises an eXtensible Markup Language (XML) path language (XPath) navigation path mapping the data-driven action to a node of the network form’s data instance.

8. One or more computer-readable media as recited in claim 1, further having computer-readable instructions therein that, when executed by the network computing device, cause the network computing device to perform acts comprising annotating the view-centric logical representation of the network form with the data-driven action based on the second relation.

9. One or more computer-readable media as recited in claim 1, further having computer-readable instructions therein that, when executed by the network computing device, cause the network computing device to perform acts comprising extracting the first relation from the data-driven action.

10. A method comprising using a network computing device configured to serve electronic forms to a client device to perform acts including:
   - obtaining a first relation for a data-driven action that maps the data-driven action to an electronic form’s data instance;
identifying a second relation to replace the first relation in a view-centric logical representation of the electronic form thereby mapping the data-driven action to the view-centric logical representation;
causing communication of the view-centric logical representation of the electronic form to a client device, the client device configured to enable the data-driven action locally using the view-centric logical representation in lieu of accessing the data instance over a network.
11. A method as recited in claim 10, wherein access to the electronic form’s data instance is controlled by the network computing device.
12. A method as recited in claim 10, wherein the network computing device is configured to replace the first relation with the second relation to generate the view-centric logical representation.
13. A method as recited in claim 10, wherein the network computing device is configured to cause communication of data describing the second relation to the client device in conjunction with the view-centric logical representation to enable mapping of the data driven action to the view-centric logical representation by the client device.
14. A method as recited in claim 10, further comprising:
receiving an event log that records a result of the data-driven action, the event log received by the network computing device from the client device subsequent to rendering of the result of the data-driven action by the client device; and
updating the electronic form’s data instance to reflect changes based on the result of data-driven action recorded in the event log.
15. A method as recited in claim 10, further comprising:
receiving a result of a data-driven action that is rendered at the client device using the view-centric logical representation of the electronic form; and
modifying the electronic form’s data instance to reflect the result of the data-driven action that is rendered at the client device using the view-centric logical representation.
16. A network computing device comprising:
one or more processors; and
memory storing instructions that, when executed by the one or more processors, cause the network computing device to perform acts including:
obtaining a view-centric logical representation of an electronic form having a data instance that is controlled under the influence of the network computing;
ascertaining a navigation path expression for a data-driven action that maps the data-driven action to the data instance of the electronic form;
causing replacement of the navigation path expression in the view-centric logical representation of the electronic form with a transformed path configured to map the data-driven action to the view-centric logical representation;
sending the view-centric logical representation of the electronic form having the transformed path to a client device, the client device configured to perform the data-driven action using the view-centric logical representation in lieu of accessing the data instance over a network.
17. A network computing device as recited in claim 16, further comprising instructions stored in the memory that, when executed by the one or more processors, cause the network computing device to perform acts including:
updating the data instance to reflect a result of the data-driven action responsive to a communication of the result to the client device at a time subsequent to rendering of the result of the data driven action at the client device.
18. A network computing device as recited in claim 16, further comprising instructions stored in the memory that, when executed by the one or more processors, cause the network computing device to perform acts including:
receiving a data log from the client device that describes multiple data-driven actions performed at the client device using the view-centric logical representation in lieu of accessing the data instance over a network; and
updating the data instance based on the received data log to reflect results of the multiple data-driven actions.
19. A network computing device as recited in claim 16, wherein the transformed path maps the data-driven action to a node of the view-centric logical representation into which the result of the data-driven action is rendered.
20. A network computing device as recited in claim 16, further comprising instructions stored in the memory that, when executed by the one or more processors, cause the network computing device to perform acts including:
annotating a node of the view-centric logical representation of the electronic form to indicate that a change to the node triggers the data-driven action.