ExtJS

Kayartaya Vinod

Who am I?

- Vinod Kumar K
- Graduated in 1994 (B.Sc, Sahyadri Science College, Shimoga)
- Learnt BASIC in 1989 and started teaching C and Foxpro in 1994
- Owned a training institute between 1999 and 2002
- Over 20 years of experience in Software Training and Development
- Mostly work with Java based technologies and web frameworks (both client and server)
  - Also code in PHP, C#, ASP.Net and more
- Share my training resources at http://vinod.co

Introduction

- ExtJS is a java-script framework that enables developers to develop Rich Internet Applications
- It is a fully featured client UI library
- It is capable of creating dynamic, fluidly laid out user interfaces

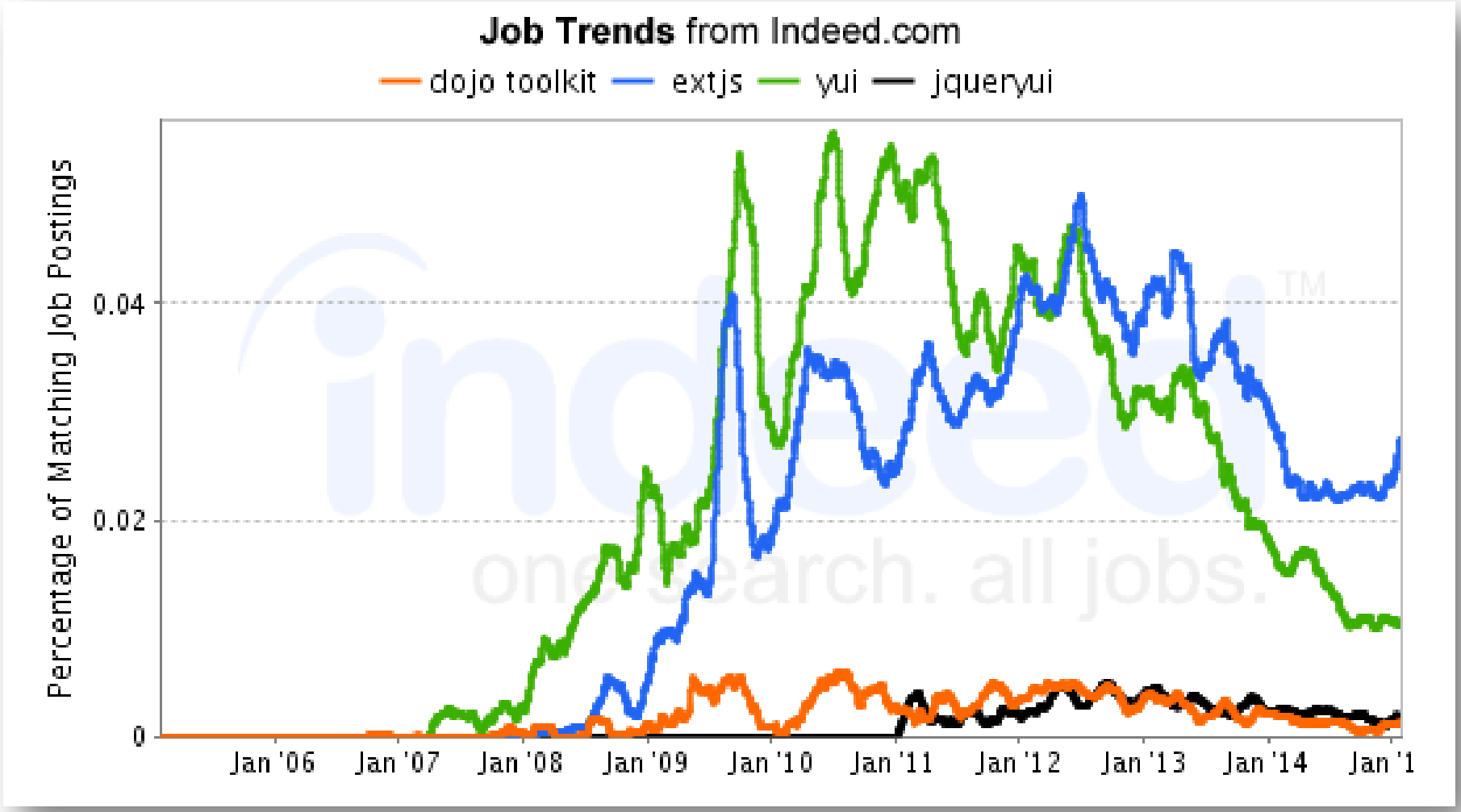
Introduction

- ExtJS has a huge collection of controls
  - Ranging from textboxes to highly sophisticated UI Controls
- UI components can be bound to structured data which itself is linked to server-side data sources

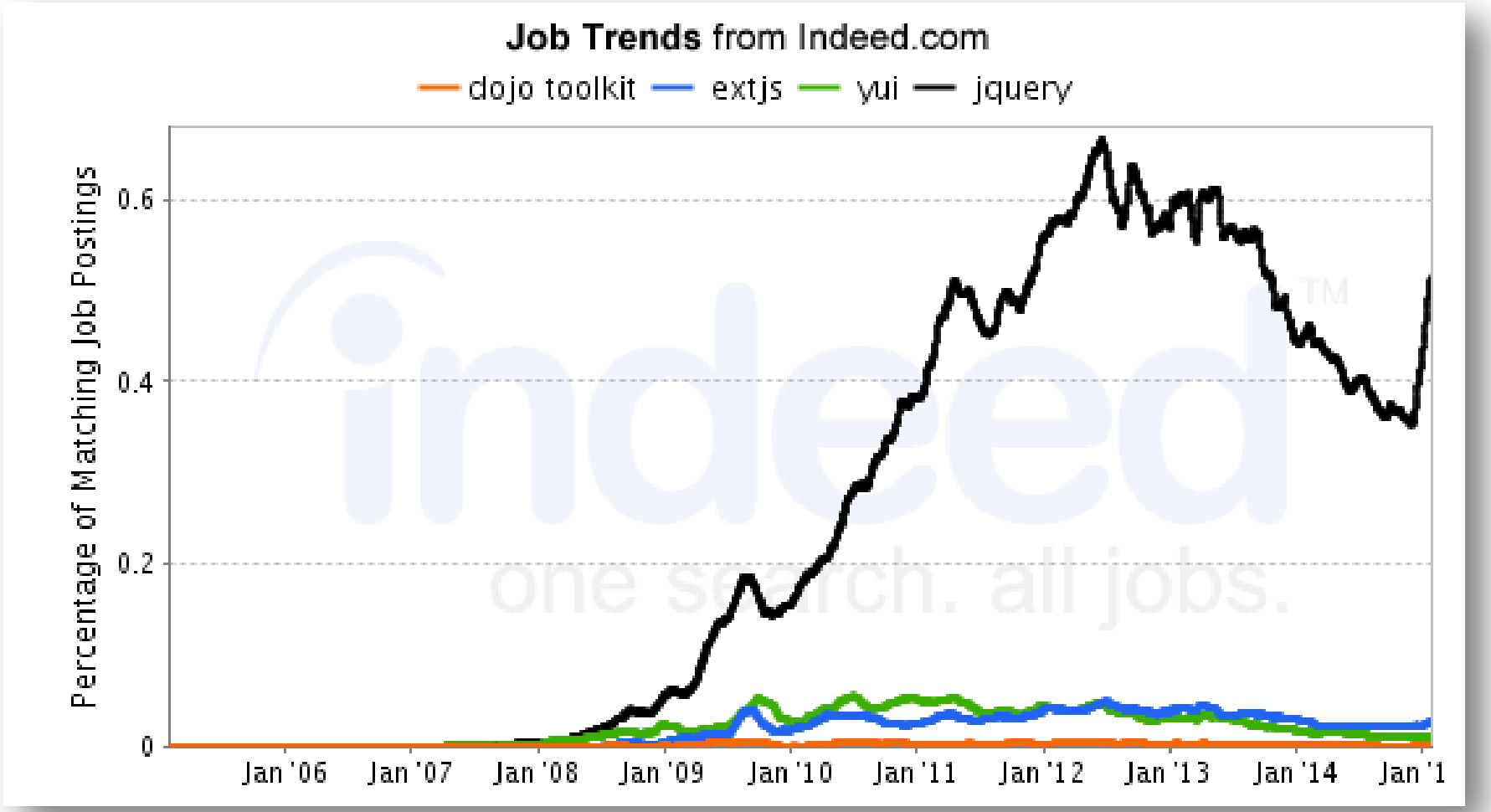
When to use

- Typically, we would use Ext in a website that requires a high level of user interaction
  - something more complex than your typical website

http://indeed.com/jobtrends



http://indeed.com/jobtrends



Easy application development

- Provids easy-to-use cross-browser compatible widgets such as windows, grids, and forms
  - The widgets are already fine-tuned to handle the intricacies of each web browser on the market, without us needing to change a thing

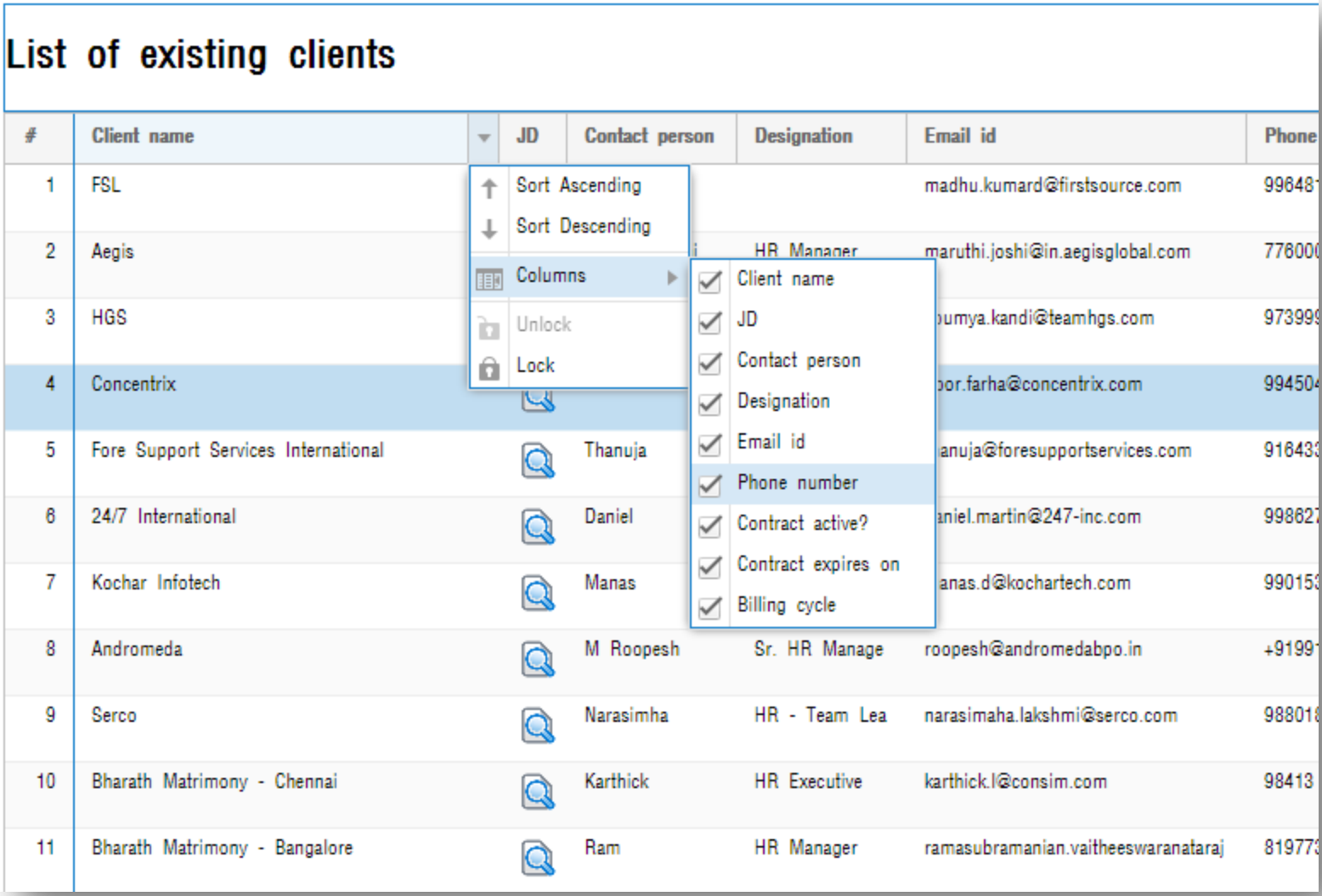
Easy application development

- Interacts with the user and browser via the EventManager, responding to the user's keystrokes, mouse clicks, and monitoring browser events
  - Window resize, font size changes

Easy application development

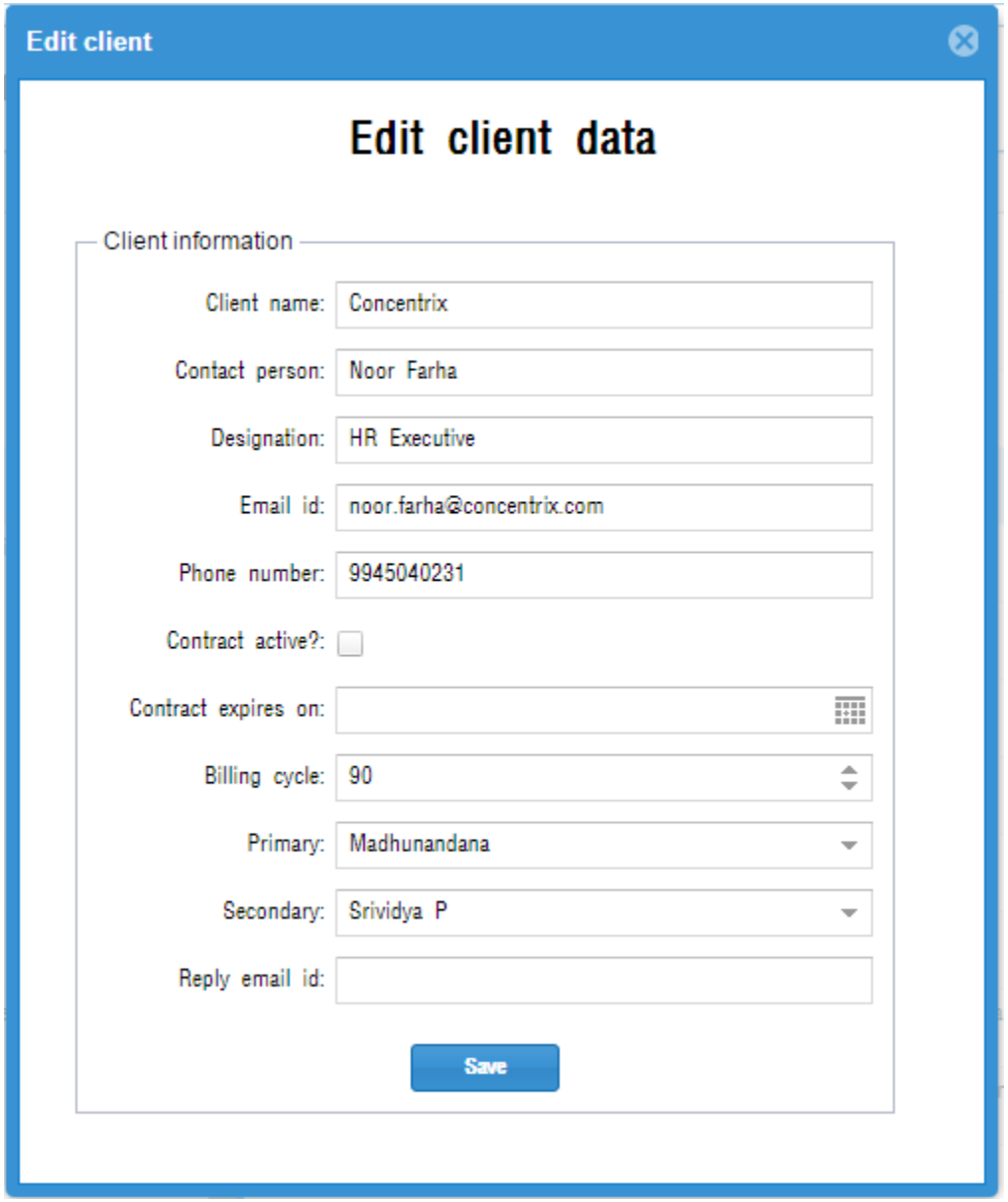
- Communicating with the server in the background without the need to refresh the page
- This allows us to request or post data to or from our web server using AJAX and process the feedback in real time

Widget showcase

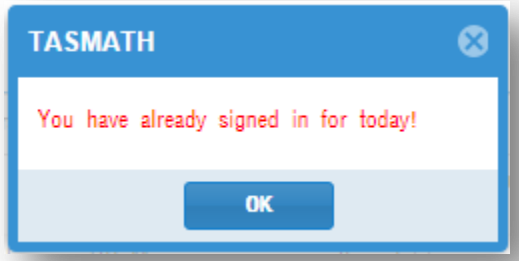


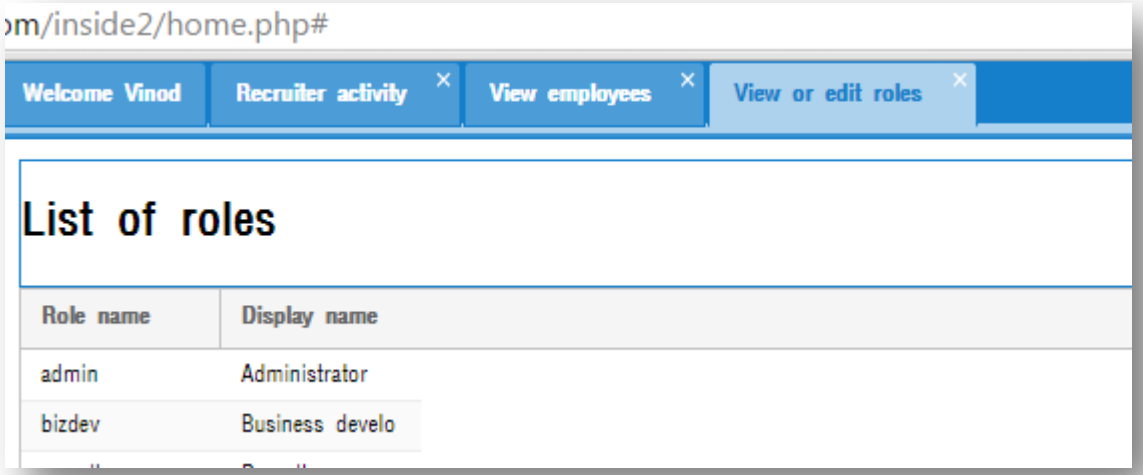
#	Client name	JD	Contact person	Designation	Email id	Phone
1	FSL				madhu.kumard@firstsource.com	99648
2	Aegis			HR Manager	maruthi.joshi@in.aegisglobal.com	77600
3	HGS				umya.kandi@teamhgs.com	97399
4	Concentrix				or.farha@concentrix.com	99450
5	Fore Support Services International		Thanuja		anuja@foresupportservices.com	91643
6	24/7 International		Daniel		niel.martin@247-inc.com	99862
7	Kochar Infotech		Manas		anas.d@kochartech.com	99015
8	Andromeda		M Roopesh	Sr. HR Manage	roopesh@andromedabpo.in	+9199
9	Serco		Narasimha	HR - Team Lea	narasimaha.lakshmi@serco.com	98801
10	Bharath Matrimony - Chennai		Karthick	HR Executive	karthick.l@consim.com	98413
11	Bharath Matrimony - Bangalore		Ram	HR Manager	ramasubramanian.vaitheeswaranataraj	81977

Widget showcase

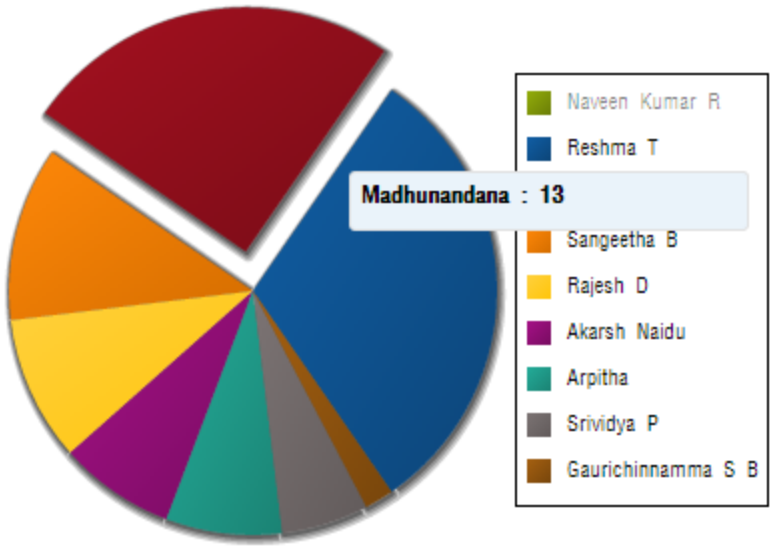


Widget showcase

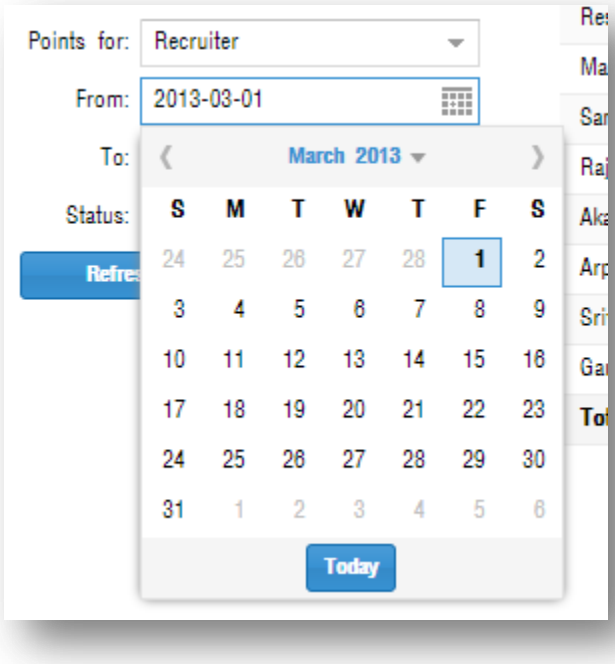




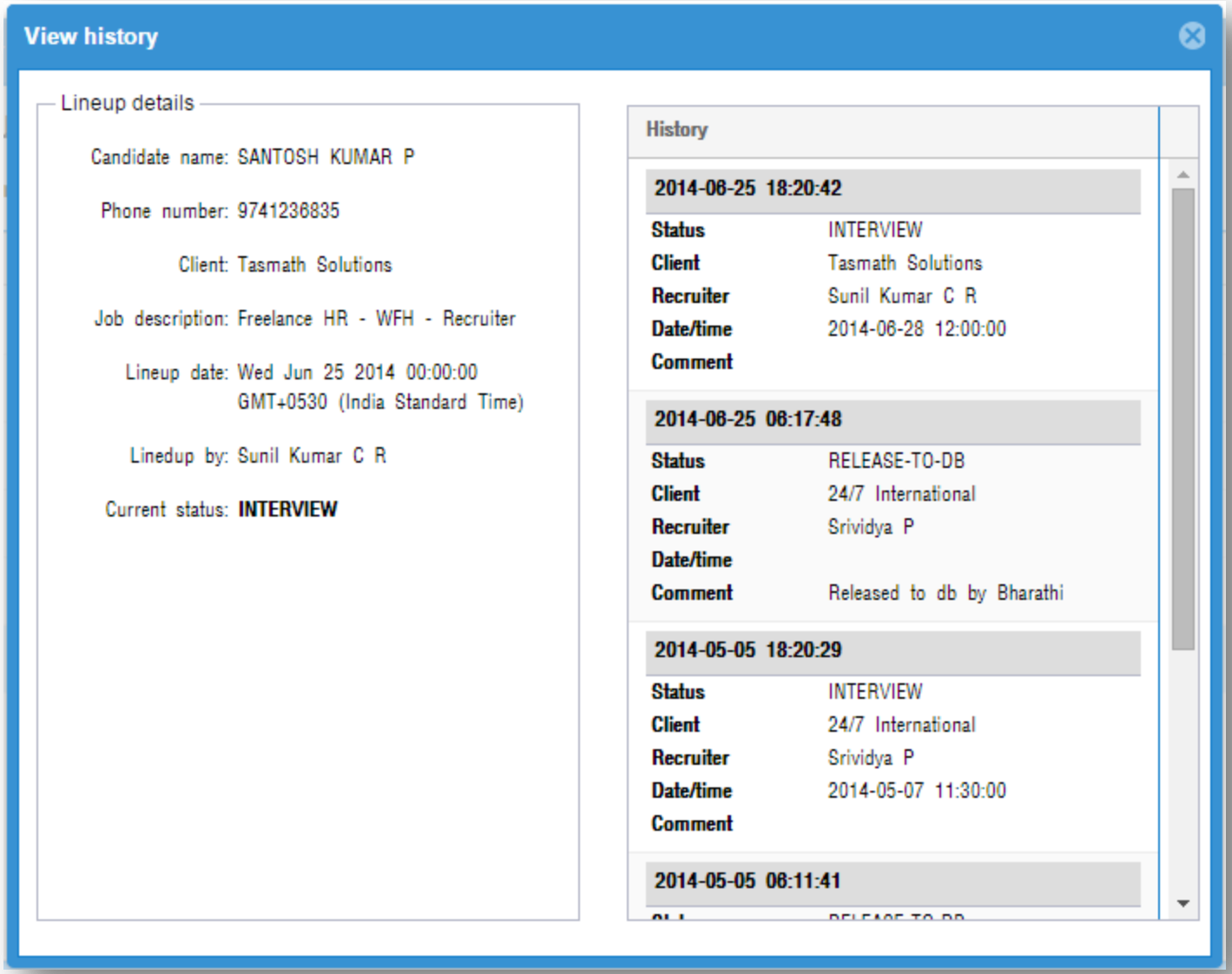
Name	Points
Naveen Kumar R	84
Reshma T	16
Madhunandana	13
Sangeetha B	6
Rajesh D	5
Akarsh Naidu	4
Arpitha	4
Srividya P	3
Gaurichinnamma S B	1
Total	116



Widget showcase



Widget showcase



Hides browser quirks

- Extjs 3.2 is compatible with:
  - Internet Explorer 6+
  - Firefox 1.5 + (PC, Mac)
  - Safari 2+
  - Opera 9 + (PC, Mac)
  - Chrome 1+

Getting Ext JS

- Download the zip file from:

  http://www.sencha.com/products/extjs/download/ext-js-3.2.1

Getting started

- Add references to CSS and JavaScript files

```
<link rel="stylesheet" type="text/css"
      href="../resources/css/ext-all.css" />

<script src="../adapter/ext/ext-base-debug.js"></script>
<script src="../ext-all-debug.js.js"></script>
```

- Check if Ext is loaded

```
<script type="text/javascript">
    Ext.onReady(function(){
        Ext.Msg.alert('VinApp', 'Hello, World!');
    });
</script>
```

ext-all.css

- A stylesheet file that controls the look and feel of Ext JS widgets
- This file must always be included as-is, with no modifications
- Any changes to the CSS in this file would break future upgrades
- If we decide that the look and feel of Ext JS needs to be adjusted, another stylesheet containing the overrides should be included after the ext-all.css file

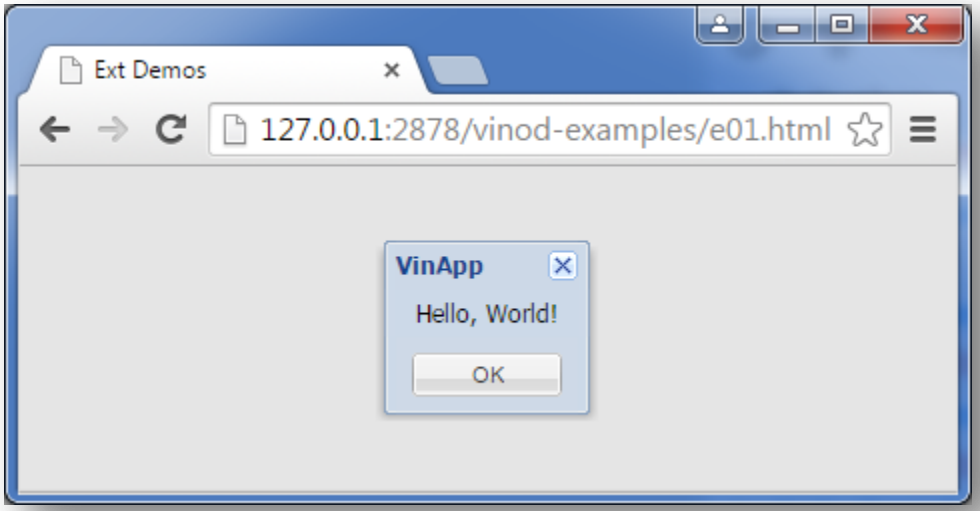
ext-base.js

- This file provides the core functionality of Ext JS
- It's the foundation upon which Ext JS builds its capabilities, and provides the interface to the browser environment
- This is the file that we would change if we wanted to use another library, such as jQuery, along with Ext JS

ext-all-debug.js/ext-all.js

- All of the widgets live in this file
- The debug version should always be used during development, and then swapped out for the non-debug version for production

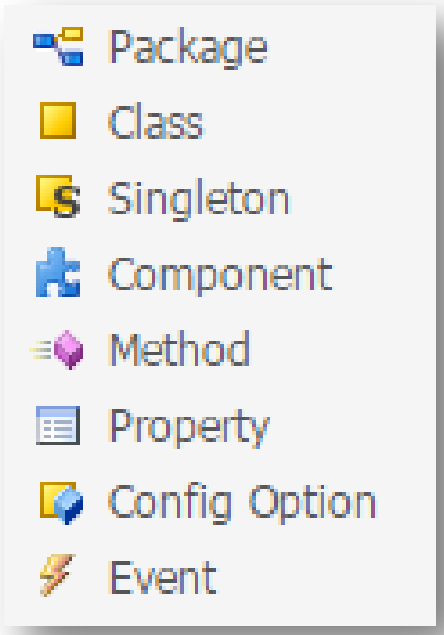
The outcome

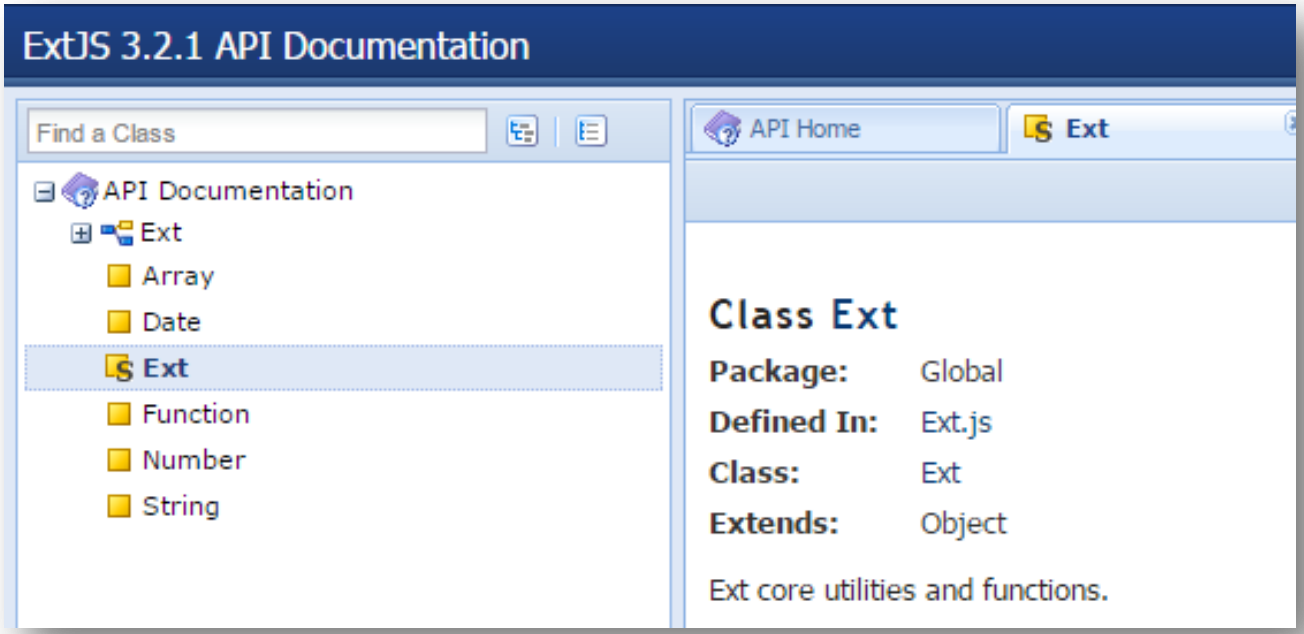

Ext Demos
127.0.0.1:2878/vinod-examples/e01.html
VinApp
Hello, World!
OK

Note about Ext.onReady

- Ext JS can only render widgets when the HTML document has been fully initialized by the browser
- All Ext JS pages must only begin accessing the document within an Ext.onReady call

Ext class system





Enhancing the existing classes

- Extjs adds extra capabilities to existing JavaScript classes
  - Array, Date, Function, Number, String,
- For example, the String class is added with these methods:
  - Static methods: escape, format, leftpad
  - Non static methods: toggle, trim

The ‘Ext’ class

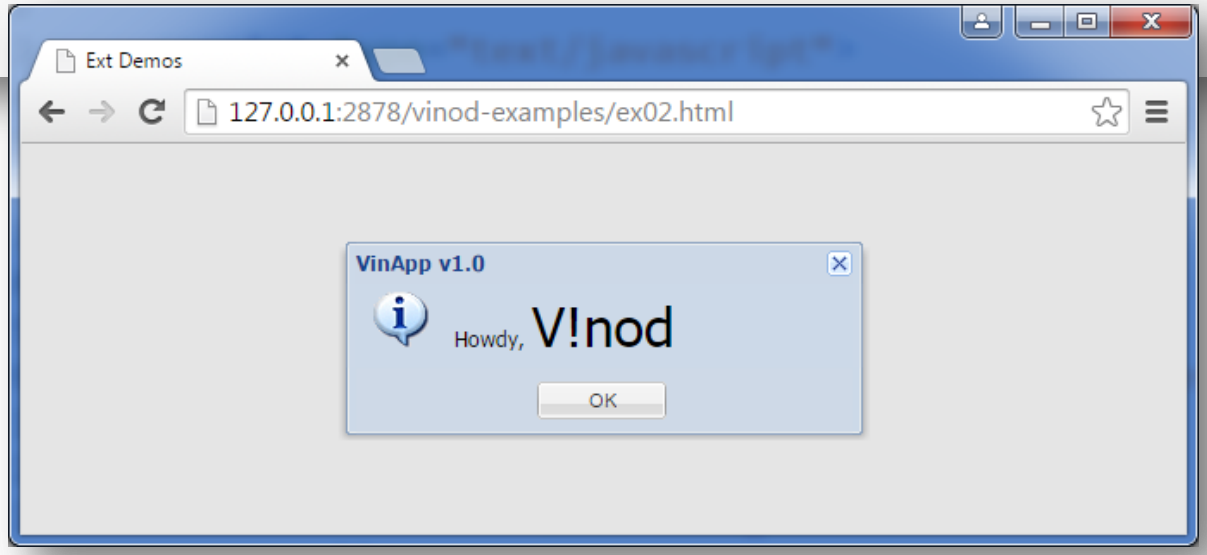
- Contains core utilities and functions
  - `isIe, isChrome, isOpera, isGeco, isLinux`
  - `apply, applyIf, encode, decode, get, fly, each`
- This class is a singleton
  - Can not create an object

```
> var e = new Ext()
  ▶ Uncaught ▶ TypeError: object is not a function
> Ext.encode({name: "Vinod"})
<· "{"name":"Vinod"}"
```

Creating a message box

- Let's create a customized message box
- Ext.Msg is a short name for Ext.MessageBox

```
Ext.Msg.show({
   title: 'VinApp v1.0',
   msg: 'Howdy, <span style="font-size: xx-large">V!nod</span>',
   width: 300,
   buttons: Ext.MessageBox.OK,
   icon: Ext.MessageBox.INFO
});
```



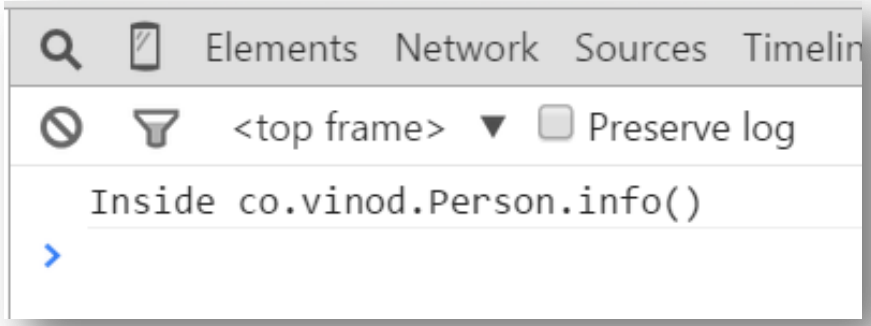
Extjs OOP support

- You can create a namespace for your classes using the Ext.namespace(..) function
  - Ext.namespace(“co.vinod”);
- Using Ext.extend(..), we can create a class
  - co.vinod.Person = Ext.extend(Object, {…});

Creating a class

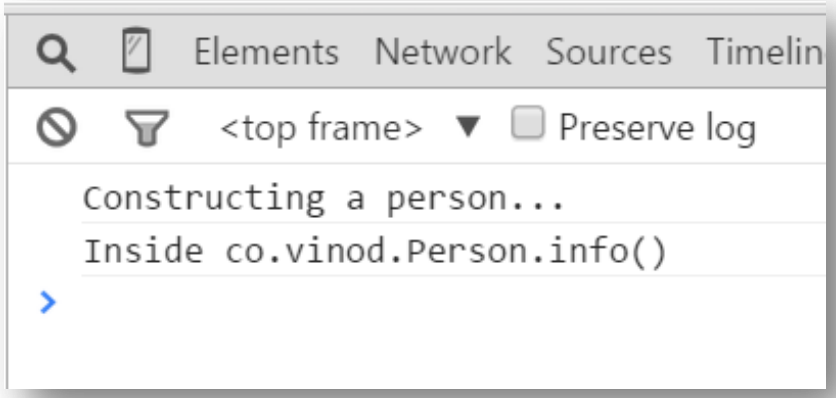
```
Ext.namespace("co.vinod");
co.vinod.Person = Ext.extend(Object, {
    info: function(){
        console.log("Inside co.vinod.Person.info()")
    }
});

var p1 = new co.vinod.Person();
p1.info();
```



Using constructor

```
Ext.namespace("co.vinod");
co.vinod.Person = Ext.extend(Object, {
    constructor: function(config){
        console.log("Constructing a person...");
        co.vinod.Person.superclass.constructor.call(this, config);
    },
    info: function(){
        console.log("Inside co.vinod.Person.info()")
    }
});
```



Using constructor

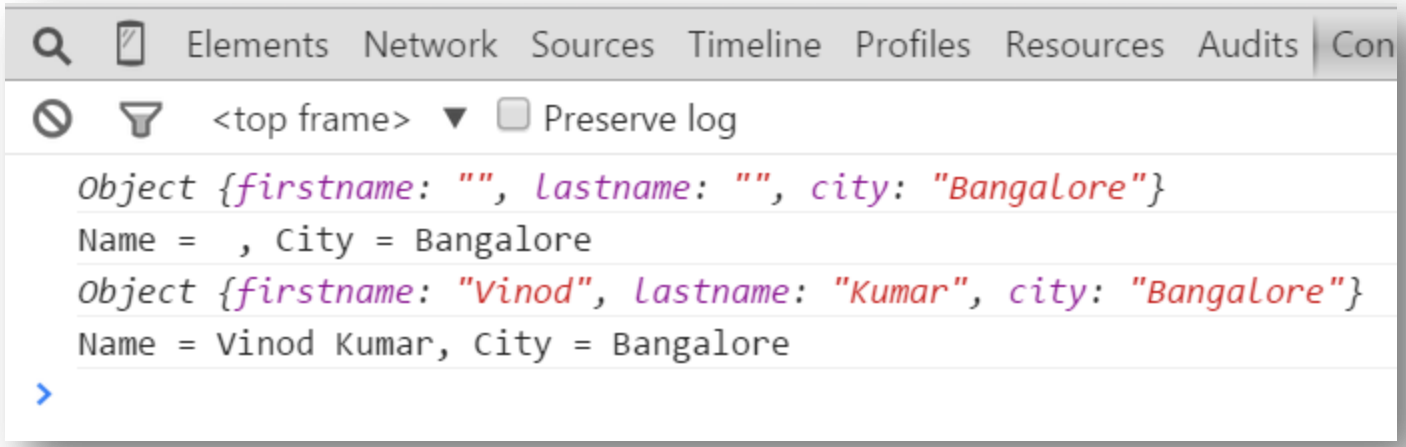
```
constructor: function(config){
    config = config || {};
    var defaults = {
        firstname: "",
        lastname: "",
        city: "Bangalore"
    };
    Ext.applyIf(config, defaults);
    Ext.applyIf(this, config);
    console.log(config);
    co.vinod.Person.superclass.constructor.call(this, config);
},
```

```
var p1 = new co.vinod.Person();
var p2 = new co.vinod.Person({
    firstname: "Vinod",
    lastname: "Kumar"});
```

Using constructor

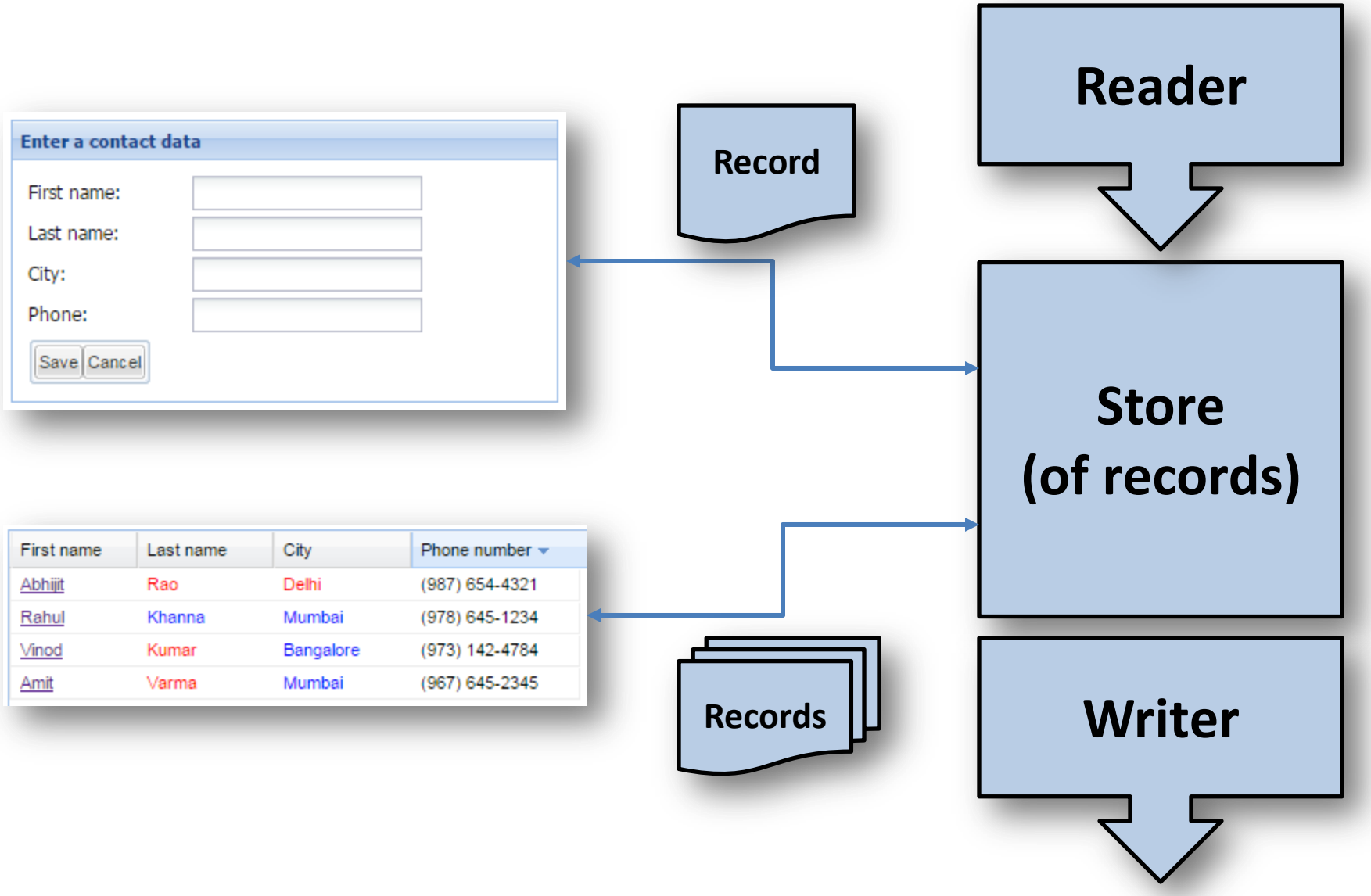
```
info: function(){
    console.log("Name = " + this.firstname
        + " " + this.lastname
        + ", City = " + this.city);
}
```

```
p1.info();
p2.info();
```



WORKING WITH DATA

Working with data stores



Records

- A record represents the unit of related data
- A record contains one ore more fields
- For example,

```
{
    id          : 20,
    firstName   : "Vinod",
    lastName    : "Kumar",
    city        : "Bangalore",
    phone       : "9731424784"
}
```

Defining a record

- A record definition can be created using the method

  Ext.data.Record.create(options);

```
var Person = Ext.data.Record.create([
    { name: "id", type: "int", mapping: "_id"},
    { name: "firstName", type: "string", mapping: "first_name" },
    { name: "lastName", type: "string", mapping: "last_name" },
    { name: "city", type: "string" },
    { name: "phone", type: "string" }
]);
```

- An array of field definitions is supplied as argument

```
{ name: "firstName", type: "string", mapping: "first_name" },
```

Useful options for a field

- name - The name by which the field is referenced within the Record
- mapping - Usually a string representing the field in the data coming from (or sent to) the server.
  - For example, if the name of the property in the JSON coming from the server is “first_name”, it can be renamed in the record as “firstName”

Useful options for a field

- type - The data type for automatic conversion from received data to the stored value
  - Some of the possible values are
    - string
    - int
    - float
    - boolean
    - date

Creating a record

- Once a record definition is available, a record instance can be created by using the “new” operator

```
var p1 = new Person({
    id          : 20,
    firstName   : "Vinod",
    lastName    : "Kumar",
    city        : "Bangalore",
    phone       : "9731424784"
});
```

Using a record

- A data in a record can be added to a store or loaded to a form using the methods provided by the respective components

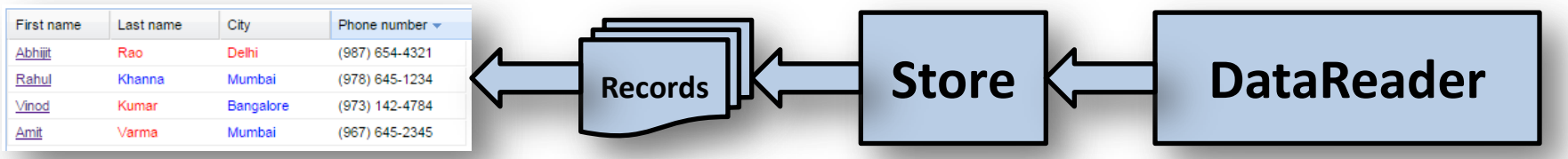
```
var p1 = grid.getSelectionModel().getSelected();
formPanel.getForm().loadRecord(p1)
var p1 = new Person({…});
personStore.add(p1);
```

Store

- Store represents a client side cache of Record objects which provide input data for Components such as the GridPanel, the ComboBox, or the DataView
- An instance of a store can be created using Ext.data.Store or one of its subclasses:
  - ArrayStore
  - JsonStore
  - XmlStore

Store

- A Store object has no inherent knowledge of the format of the data object
  - It could be an Array, XML, or JSON
- A Store uses an appropriate implementation of a DataReader to create Record instances from the data object



DataReader

- The class Ext.data.DataReader is an abstract base class for reading structured data from a data source
- Converts the data into a Record and metadata for use by an Ext.data.Store.
- Use one of the subclasses:
  - ArrayReader, JsonReader and XmlReader.

DataReader

- A JsonStore expects a “success: true” attribute on every successful response
- Stores require the data to be placed under a root attribute (default is “data”)
- Read operation
  - Return store data in array under the “data” element

Structure of data expected by Ext

```
{
    "data": [
        {
            "_id": 1,
            "first_name": "Vinod",
            "last_name": "Kumar",
            "city": "Bangalore",
            "phone": "9731424784"
        },
        {
            // data
        },
        {
            // data
        }
    ],
    "success": true
}
```

Adding a Reader to a Store

```
var store = new Ext.data.Store({
    url: "http://localhost/persons",
    reader: new Ext.data.JsonReader({
        fields: Person,
        root: "data",
        successProperty: "success",
        idProperty: "_id",
    })
});
```

DataWriter

- Ext.data.DataWriter facilitates save, update and delete actions between an Ext.data.Store and a server-side framework
- A Writer enabled Store will automatically manage the Ajax requests to perform CRUD actions on a Store
- Ext.data.DataWriter is an abstract base class which is intended to be extended and should not be created directly
- For instantiating a DataWriter, use
  - JsonWriter
  - XmlWriter

JsonWriter

```
var store = new Ext.data.Store({
    restful: true,
    autoLoad: true,
    url: "http://localhost/persons",
    reader: new Ext.data.JsonReader({
        successProperty: 'success',
        idProperty: '_id',
        root: 'data',
        fields: Person
    }),
    writer: new Ext.data.JsonWriter({
        encode: false,
        writeAllFields: true
    }),
    autoSave: true
});
```

- encode: false
  - sends data as request payload than as form data
- writeAllFields: true
  - Send all data available in the record when updating
- writeAllFields: false
  - Send only modified data during an update

Store URL CRUD operations

- restful: true
- url: “http://localhost/persons”

Store operation	HTML operation	URL
create	POST	http://localhost/persons
load	GET	http://localhost/persons
save	UPDATE	http://localhost/persons/{id}
destroy	DELETE	http://localhost/persons/{id}

Overriding the JsonWriter

- Since the JsonReader is configured to fetch the data from a root property (“data”), while sending the data, the JsonWriter also sends the record under a property “data”

JsonWriter

```
var person1 = new Person({
    "id": 89,
    "firstName": "John",
    "lastName": "Doe",
    "city": "Newyork",
    "phone": "(555) 234-2324"
    });

store.insert(0, person1);
```

```
{data: {_id: 89, first_name: "John", last_name: "Doe",
            city: "Newyork", phone: "(555) 234-2324"}}
```

Overriding the JsonWriter

- This means the server-side code also need to fetch the actual data from the “data” property
- In most frameworks, the POST and PUT request handlers would simply read the payload and interpret it as the record to be added/updated
- Hence, we need to override the default behaviour of a JsonWriter to send only the record as payload

Overriding the JsonWriter

```
Ext.override(Ext.data.JsonWriter, {
    render : function(params, baseParams, data) {
        if (this.encode === true) {
            Ext.apply(params, baseParams);
            params[this.meta.root] = Ext.encode(data);
        }
        else {
            var jdata = Ext.apply({}, baseParams);
            jdata = Ext.apply(jdata, data);
            params.jsonData = jdata;
        }
    }
});
```

Before and after overriding the JsonWriter

Request Payload view source

{data: {_id: 89, first_name: "John", last_name: "Doe", city: "Newyork", phone: "(555) 234-2324"}}

Request Payload view source

{_id: 89, first_name: "John", last_name: "Doe", city: "Newyork", phone: "(555) 234-2324"}

Loading data into a grid

- Ext.grid.GridPanel can be used to represent the array of records in a tabular (grid) style

	First name	Last name	City	Phone number
1	Vinod	Kumar	Bangalore	9731424784
2	Shyam	Sundar	Bangalore	9998887776
3	John	Doel	Dallas	199282292
4	Jane	Doe	Newyork	1988827722
5	Krishna	Kamath	Bangalore	929282822
6	Ramesh	Raj	Chennai	8272625522
7	Anil	Kumar	Shimoga	6474746464
8	Arun	Kumar	Bangalore	7262525354
9	Scott	Ross	Dallas	
10	John	Smith	Newyork	5564425222
11	Satya	Narayan	Bangalore	
12	Scott			
13	John	Doe	Newyork	(555) 234-2324

Creating a grid

- The grid can be constructed by supplying the “store” and “columns” property

```
var grid = new Ext.grid.GridPanel({
    store: store,
    columns: [
        new Ext.grid.RowNumberer(),
        {
            header: "First name",
            dataIndex: "firstName",
            width: 200,
            sortable: true
        },
        { // column definition for lastName },
        { // column definition for city },
        { // column definition for phone }
    ]
});
```

Making the grid editable

- Add the RowEditor.css and RowEditor.js references to your HTML

```
<link type="text/css" rel="stylesheet"
      href="ux/css/RowEditor.css" />

<script type="text/javascript"
    src="ux/RowEditor.js">
</script>
```

Making the grid editable

- Instantiage the RowEditor plugin

```
var editor = new Ext.ux.grid.RowEditor({
    saveText: 'Update'
});
```

- Add the plugin to the grid

```
var grid = new Ext.grid.GridPanel({
    plugins: [ editor ],
    store: store,
    columns: [
        new Ext.grid.RowNumberer(),
```

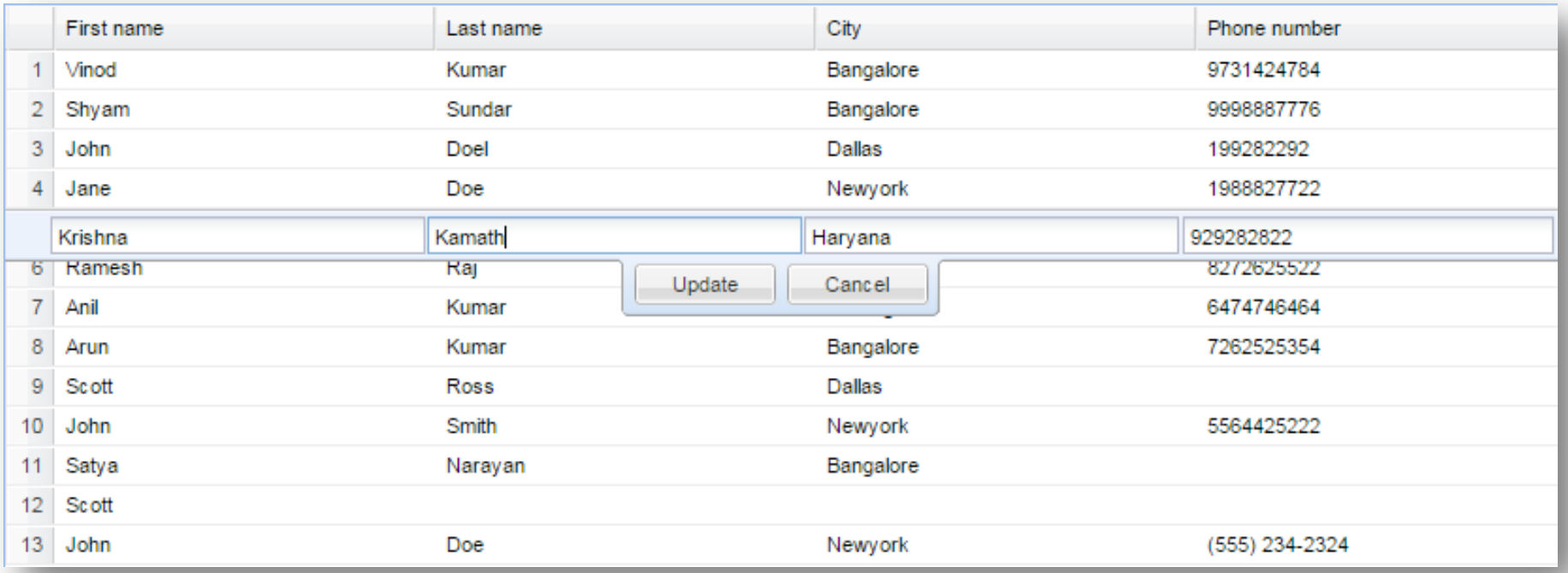
Making the grid editable

- Add the editor type for the reqired columns in the grid

```
var grid = new Ext.grid.GridPanel({
    plugins: [ editor ],
    store: store,
    columns: [
        new Ext.grid.RowNumberer(),
        {
            header: "First name",
            dataIndex: "firstName",
            width: 200,
            sortable: true,
            editor: {
                xtype: "textfield",
                allowBlank: false
            }
        },
        {
            header: "Last name",
```

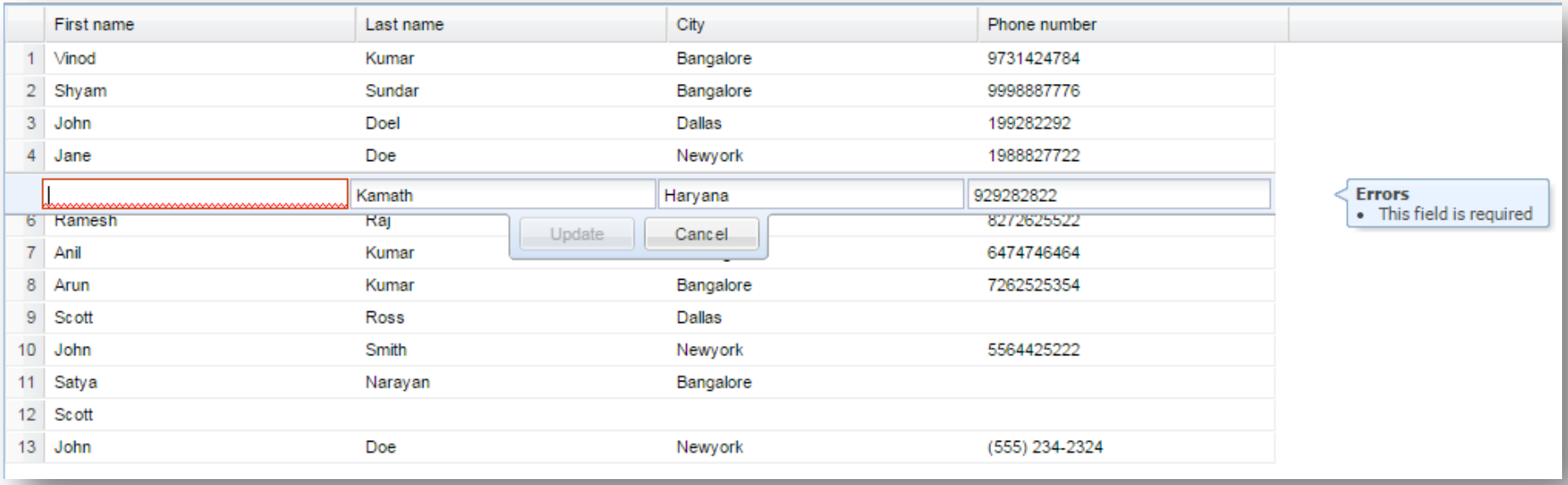
Editable grid

- Double clicking a row will allow an inline editing



Editable grid

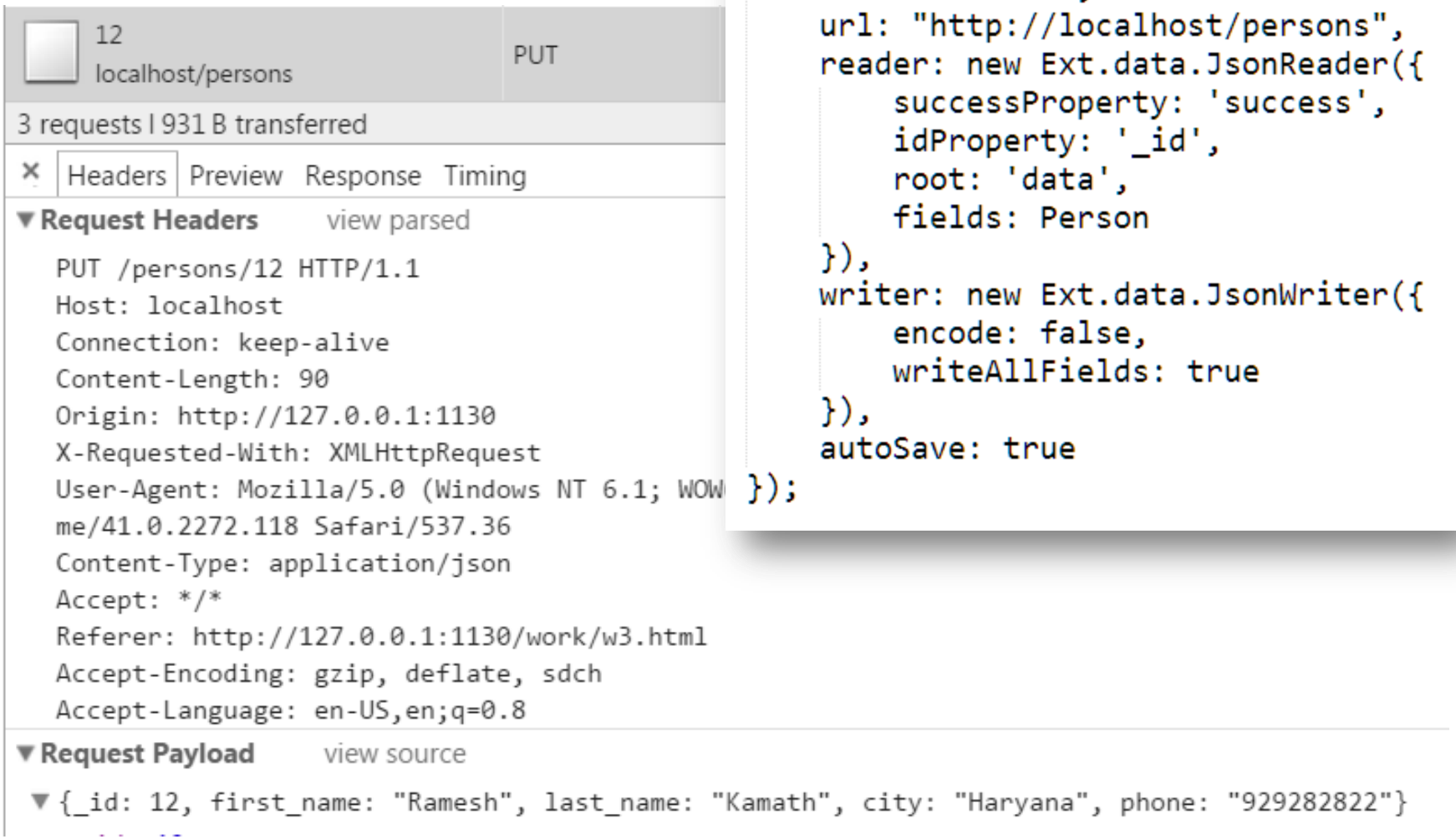
- Invalid data will not be allowed



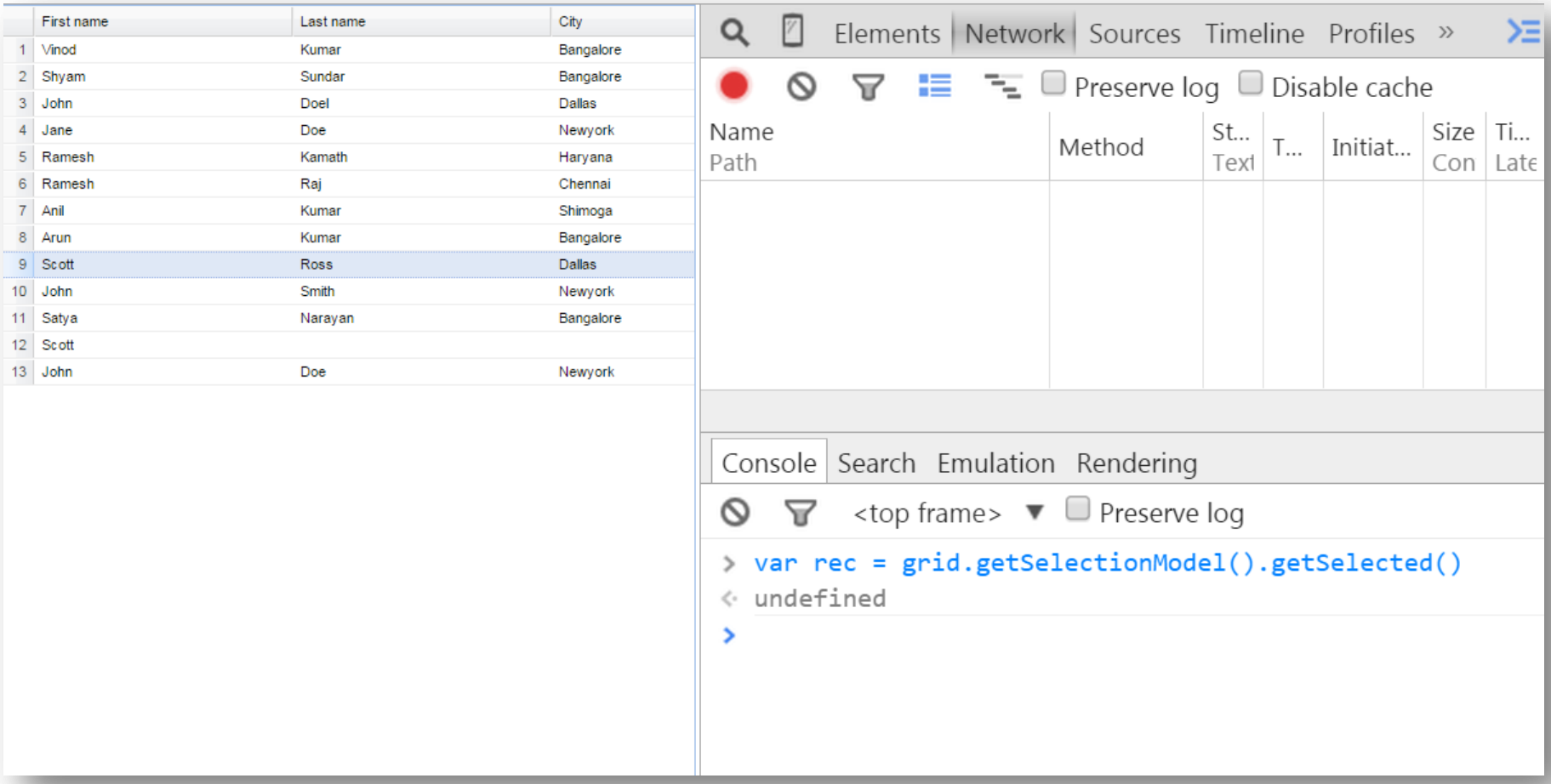
Editable grid

- Network analysis

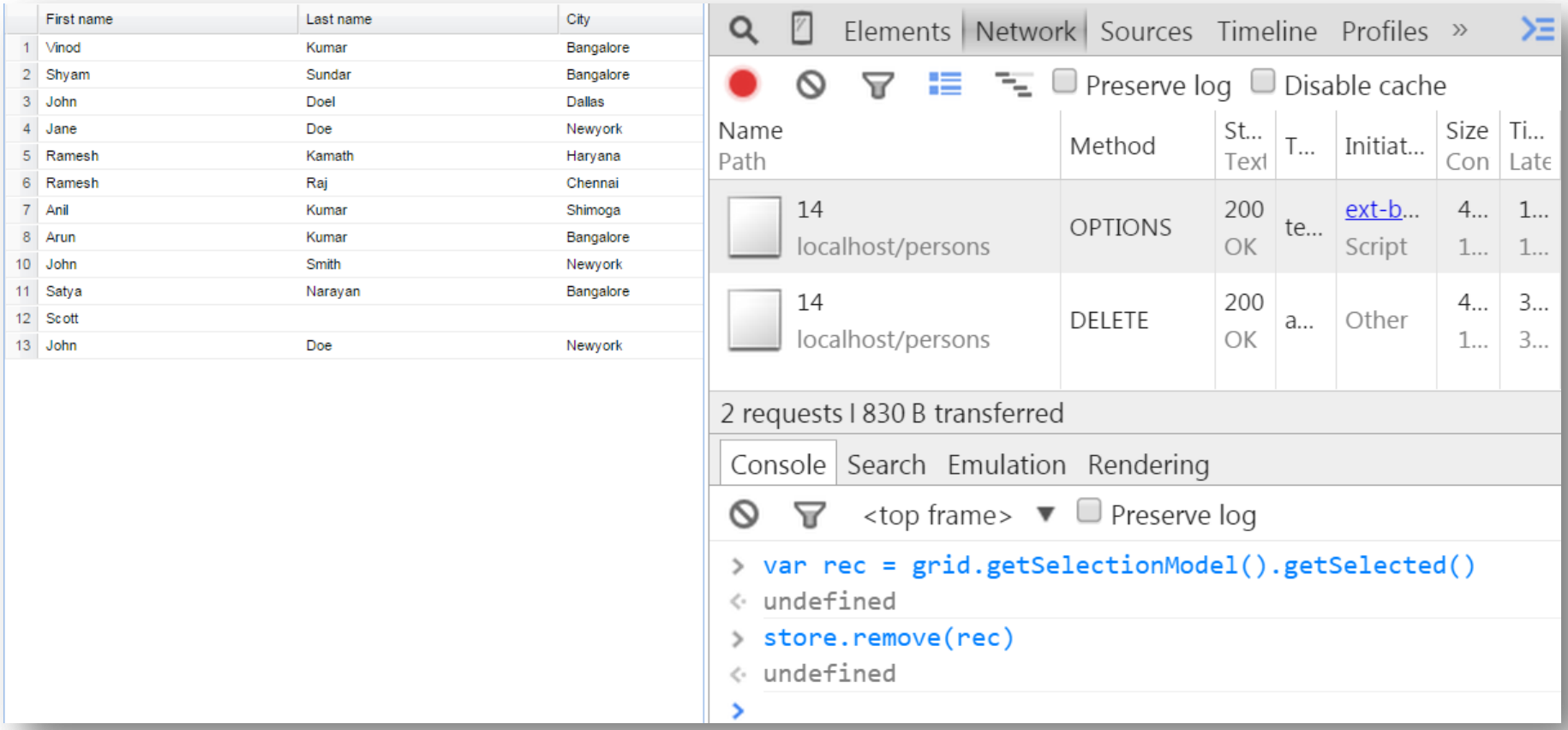
```
var store = new Ext.data.Store({
    restful: true,
    autoLoad: true,
    url: "http://localhost/persons",
    reader: new Ext.data.JsonReader({
        successProperty: 'success',
        idProperty: '_id',
        root: 'data',
        fields: Person
    }),
    writer: new Ext.data.JsonWriter({
        encode: false,
        writeAllFields: true
    }),
    autoSave: true
});
```



Deleting a row from a grid



Deleting a row from a grid



Adding a new row

- Inserting a new record into the store causes the same being reflected in the grid