SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – I-Rich Internet Applications – SCS1401

I. Rich Internet Application Overview

Introduction to Web2.0 - Key characteristics of Rich Internet Application - Current Rich Internet Application platforms - Rich Internet Application benefits - Rich Internet Application patterns and best practices - Rich Internet Application architecture - Restful Web Services with Nodes.

Introduction to Web 2.0

Web 1.0

- Web 1.0 refers to the first stage in the World Wide Web
- Entirely made up of web pages connected by hyperlinks.
- A set of static websites that were not yet providing interactive content.
- Used as "Information portal".

  Examples: Amazon,Yahoo, Personal web pages

Web 2.0

- Web 2.0 is the term used to describe a variety of web sites and applications that allow anyone to create and share online information or material they have created.
- It allows people to create, share, collaborate & communicate.
- Allows everyone to produce their content.
- Gives the users the possibility to control their data.
- **Web 2.0** allows groups of people to work on a document or spreadsheet simultaneously.
- In the background a computer keeps track of who made what changes where and when.
- **Web**-based applications can be accessed from anywhere.
- **Web 2.0 Examples**
  - Web applications (Google Docs, Flickr)
  - Video sharing sites (YouTube)
  - Wikis (Media Wiki)
  - Blogs (WordPress)

- Social networking (Facebook)
- Microblogging (Twitter)
- Hosted services (Google Maps)

Web 3.0

- It refers to the evolution of web utilization and interaction which includes altering the Web into a database.
- It enables the upgradation of back-end of the web, after a long time of focus on the front-end
- Data isn't owned but instead shared, where services show different views for the same web / the same data.
- The Semantic Web (3.0) promises to establish "the world's information" in more reasonable way than Google can ever attain with their existing engine schema.
- The Semantic Web necessitates the use of a declarative ontological language like OWL to produce domain-specific ontologies that machines can use to reason about information and make new conclusions.
- **Examples:** Online Coupons, Voice Search, FB app

Difference between Web 1.0, Web 2.0 and Web 3.0

The difference between web 1.0,2.0,3.0 are discussed in the figure 1.1.

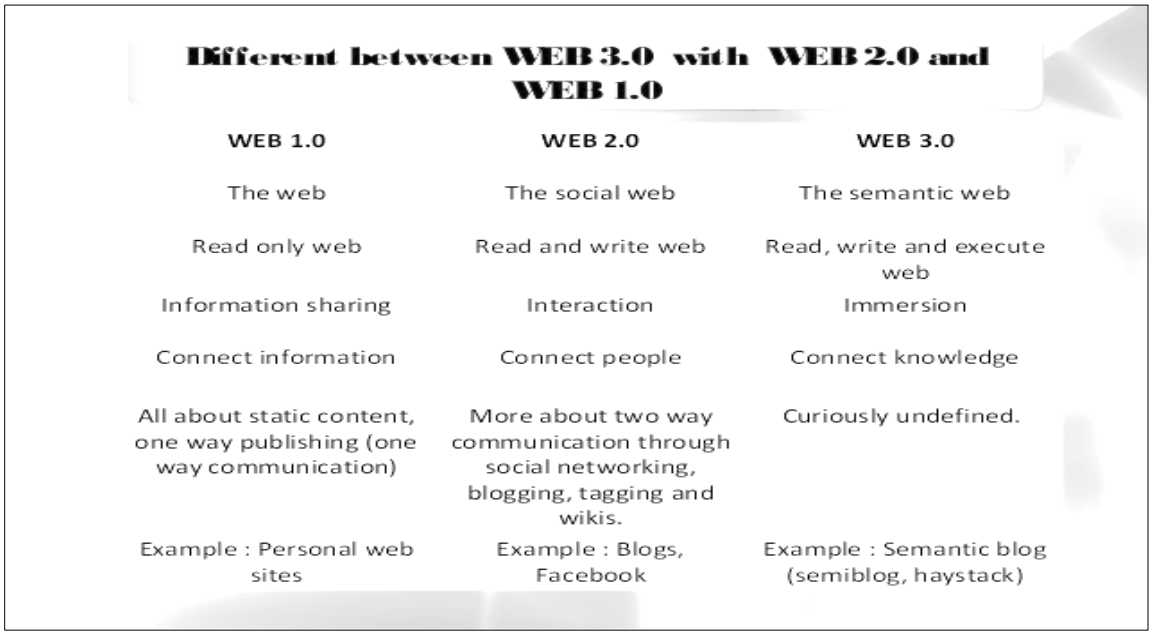

Figure 1.1 Difference between Web 1.0,Web 2.0,Web 3.0

What is RIA?

- Rich Internet Applications (RIAs) are web applications that have the features and functionality of traditional desktop applications.
- RIAs typically provide a "no-refresh" look to the user interface.
- RIA provides HDuX – High Definition User eXperience.

Limitations of HTML based Web Applications

- **Process Complexity** – Multi step tasks are time consuming and frustrating.
- **Data Complexity** – HTML based web applications don't facilitate the manipulation and visualization of complex data.
- **Feedback Complexity** – Server based processing limits the scope of an interactive user experience.

What is special in RIA?

- RIA works in Web.
- RIA appears – never refreshes.

- RIA reduces network traffic.
- RIA is rich.
- RIA makes it easy.

History of RIA

- The term "rich Internet application" was introduced in a white paper in March 2002 by Macromedia (now merged into Adobe).
- The concept had existed earlier under names such as:
  - Remote Scripting, by Microsoft, circa 1999
  - X Internet, by Forrester Research in October 2000
  - Rich (Web) clients
  - Rich Web application

Differences between a Desktop, Traditional Web Application and RIA

The below figure 1.2 describes the difference between Desktop, Traditional Web Applications and Rich Internet Applications.

Features	Desktop	Web	RIA
Rich User Experience	Yes	No	Yes
Interactive, Responsive	Yes	No	Yes
Low Maintenance	No	Yes	Yes

Figure 1.2 Difference between Desktop, Traditional Web, RIA

RIA Tools

- **Adobe Flex** → a highly productive, open source application framework for building and maintaining expressive web applications that deploy consistently on all major browsers, desktops and devices.
- **OpenLaszlo** → a open source platform for the development and delivery of Rich Internet Applications, consists of the LZX programming language and the OpenLaszlo Server.
- **Microsoft Silverlight** → a powerful development tool for creating engaging, interactive user experiences for Web and mobile applications.
- **JavaFX** → a software platform for creating and delivering desktop applications, as well as Rich Internet Applications (RIAs) that can run across a wide variety of devices.
- The logos of RIA tools are shown in figure 1.3.

Figure 1.3 RIA Tools

Platforms of RIA

- Adobe Flash
- Java
- Microsoft Silverlight
- AJAX

Adobe Flash

- Manipulates vector and raster graphics to provide animation of text, drawings, and still images.
- Supports bidirectional streaming of audio and video.
- It can capture user input via mouse, keyboard, microphone, and camera.
- Flash contains an object-oriented language called ActionScript and supports automation via the JavaScript Flash language (JSFL).
- Flash content may be displayed on various computer systems and devices, using Adobe Flash Player, some mobile phones and a few other electronic devices (using Flash Lite).

Java

- Java applets are used to create interactive visualization.
- To present video, three dimensional objects and other media.

- Java applets are more appropriate for complex visualizations that require significant programming effort in high level language or communications between applet and originating server.
- JavaFX is considered as another competitor for Rich Internet Applications.

Microsoft Silverlight

- It has emerged as a potential competitor to Flash.
- To provide video streaming for many high profile events, including the 2008 Summer Olympics in Beijing, the 2010 Winter Olympics in Vancouver, and the 2008 conventions for both major political parties in the United States.
- Silverlight is also used by Netflix for its instant video streaming service.

Characteristics of RIA

- Rich user experience
- Interactive- An RIA can use a wider range of controls that allow greater efficiency and enhance the user experience.
  - Users can interact directly with page elements through editing or drag-and-drop tools.
  - They can also do things like pan across a map or other image.
- Responsive
- Low maintenance
- Is plastic (changing, transforming)
- Breaks walled gardens
- **Partial-page updating:** RIAs incorporate additional technologies, such as real-time streaming, high-performance client-side virtual machines, and local caching mechanisms that reduce latency (wait times) and increase responsiveness.
- **Better feedback:** Because of their ability to change parts of pages without reloading, RIAs can provide the user with fast and accurate feedback, real-time confirmation of actions and choices, and informative and detailed error messages.
- **Consistency of look and feel:** With RIA tools, the user interface and experience with different browsers and operating systems can be more carefully controlled and made consistent.
- **Offline use**: When connectivity is unavailable, it might still be possible to use an RIA if the app is designed to retain its state locally on the client machine.
- **Caching**

- RIA client has the ability of keeping the server information during a period of time improving application performance and UI responsiveness.
- **Security**
- RIAs should be as secure as any other web application, and the framework should be well equipped to enforce limitations appropriately when the user lacks the required privileges, especially when running within a constrained environment such as a sandbox.
- **Advanced Communication**
- Sophisticated communications with supporting servers through optimized network protocols can considerably enhance the user experience.
- **Rapid Development**

  An RIA Framework should facilitate rapid development of a rich user experience through its easy-to-use interfaces in ways that help developers.
- **Improved Features**

  RIA allow programmers to embed various functionalities in graphics-based web pages that look fascinating and engaging like desktop applications.

  RIA provide complex application screens on which various mixed media, including different fonts, vector graphic and bitmap files online conferencing etc. are paused by using different modern development tools.

Advantages of RIA

- **Remotely accessed application** – The basic principle of RIA is the ability to have any new user connect and run the application from any location as long as they are connected to the network.
- **Full interactive experience** – Unlike Web applications that provide page-by-page interaction and feedback, RIA provides a full interactive end-user experience.
- **Integrated Form Editor**
- **A Comprehensive Solution** – A comprehensive end-to-end solution facilitating full interactive distributed clients and a centralized managed server.
- **A Single Unified IDE & Paradigm** – The RIA is supported by a single and unified IDE and development paradigm for defining server-side and client-side logic along with the user interface design.
- **Automatic Logic Partitioning** – The RIA deployment modules facilitate optimized automatic logic partitioning between client and server.

- **Performance-Aware Development** – The Rich Client Studio is a performance-aware platform for developing response-optimized applications.
- **Native Look & Feel –** It provides an automatic reflection of the native look and feel of the selected client machine.
- **Browser free solution** – RIA is independent of browser vendors and browser versions.
- **Local Resources Activation** – RIA supports various activities that can be executed on the client side. For example:
- OS command
- File manipulation
- OS environment manipulation
- **More Responsive**: The agile response from applications keeps the user engaged while improving user productivity.
- **Interactive User Interface:**
    - RIAs have more interactive UI as they can be used to provide information in more appealing way in lesser time as compared to conventional internet applications.
    - This fast interactivity with the application improves user satisfaction quite significantly.
    - **Less Internet Traffic and Faster Processing:**
        - Rich internet application does not refresh entire page and that leads to less traffic and faster processing.
- **Simplifying Online Transactions:**
    - RIAs eliminate the multi-page for multi-step transactions by presenting all pertinent information to users without leaving the initial environment.
    - This improves customer satisfaction and customer loyalty as they view the service provider as someone who understands their needs.
- **Easier Mobile Access to Information**

  A number of manufacturing systems can potentially benefit from mobile access.

  By using RIAs, plant managers and supervisors can access mission-critical information without being anchored to a static workstation.

  Mobile devices can help management more quickly identify and resolve problems on the shop floor, locate inventory and update information, in real-time and on-location.
- **Better Data Visualization**

Manufacturers who have implemented RIAs find users suggesting new and useful ways of visualizing data all the time.

- Data visualization improves decision making and can reduce product costs and direct material spent by identifying bottlenecks, anticipating shortages and improving lean manufacturing procedures.

- **Batch and Item Level Tracking**

- Tracking of inventory, supplies, spare parts, and even labor resources using RFID enhances business efficiency.

- Use of RFID and custom applications to support batch and item level tracking in manufacturing processes can improve productivity for activities.

- **Integration Across Multiple Systems**

    - RIAs can pull data from multiple data warehouses and systems, and present information on a single screen or reduced number of screens with visualization capabilities, as well as to drill-down to necessary detail.

Business Benefits of RIA

- Enables more transactions
- Retains customers
- Low operational cost
- Improves performance

<u>Application Benefits:</u>

- Add value to your applications
- Meet your customer expectations
- Simplify & speed up processes
- Get regular and dedicated visitors
- Save costs on Bandwidth
- Increase your productivity

First Generation Web Applications

- First generation web applications were page oriented.
- All GUI logic and application logic inside the same web page shown in Figure 1.4.

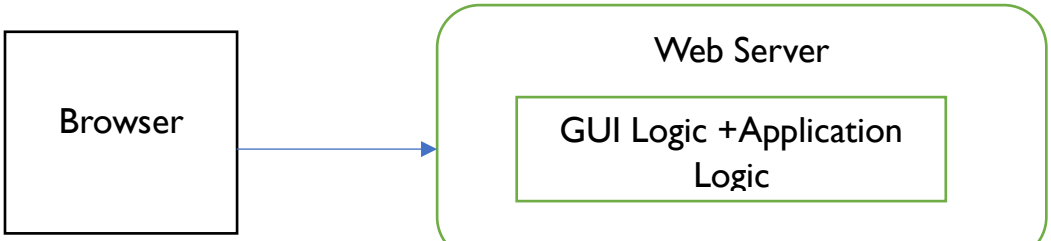

Figure 1.4 First Generation of Web Applications

- Every action the application allowed was typically embedded in its own web page script.
- Each script was like a separate transaction which executed the application logic, and generated the GUI to be sent back to the browser after the application logic was executed.
- First generation web page technologies include Servlets (Java), JSP (JavaServer Pages), ASP, PHP and CGI scripts in Perl etc.

Second Generation Web Applications

- In second generation web applications developers found ways to separate the GUI logic from the application logic on the server shown in Figure 1.5.
- Web page scripts were used for the GUI logic
- Real classes and objects were used for the application logic
- Frameworks were developed to help make second generation web applications easier to develop.
- Examples of such frameworks are ASP.NET (.NET), Struts + Struts 2 (Java), Spring MVC (Java), JSF (JavaServer Faces), Wicket (Java) Tapestry (Java) and many others.
- GUI logic was written in the same language as the application logic, changing the programming language meant rewriting the whole application again.

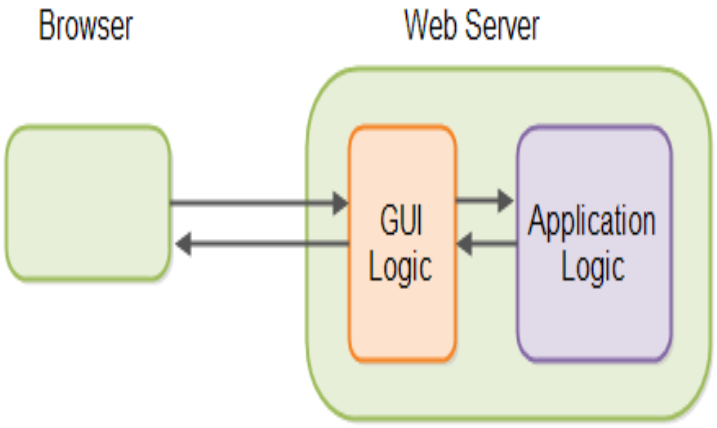

Figure 1.5 Second Generation Web Applications

RIA Web Applications

- RIA (Rich Internet Applications) web applications are the third generation of web applications.
- RIA technologies are typically executed in the browser using either JavaScript, Flash, JavaFX or Silverlight
- The GUI logic is now moved from the web server to the browser.
- GUI logic is executed in the browser, the CPU time needed to generate the GUI is lifted off the server, freeing up more CPU cycles for executing application logic.
- GUI state can be kept in the browser, thus further cleaning up the server side of RIA web applications.
- GUI logic is completely separated from the application logic is shown in Figure 1.6, it becomes easier to develop reusable GUI components
- No matter what server-side programming language is used for the application logic.

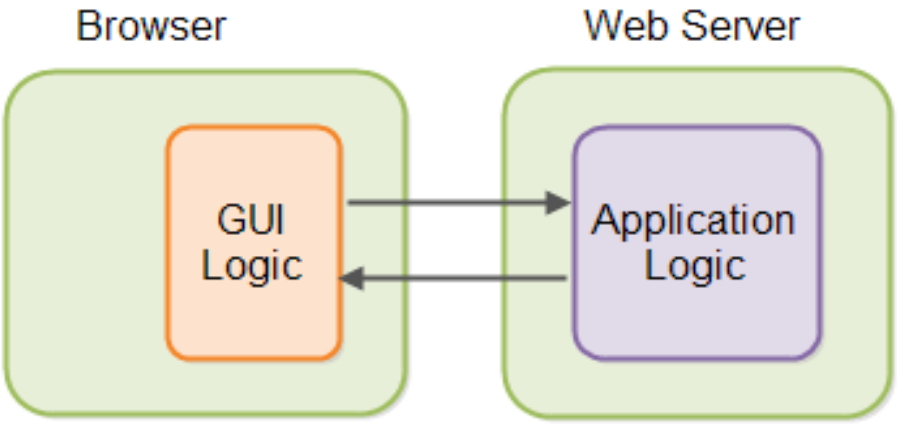

Figure 1.6 RIA Web Applications

Architecture of RIA

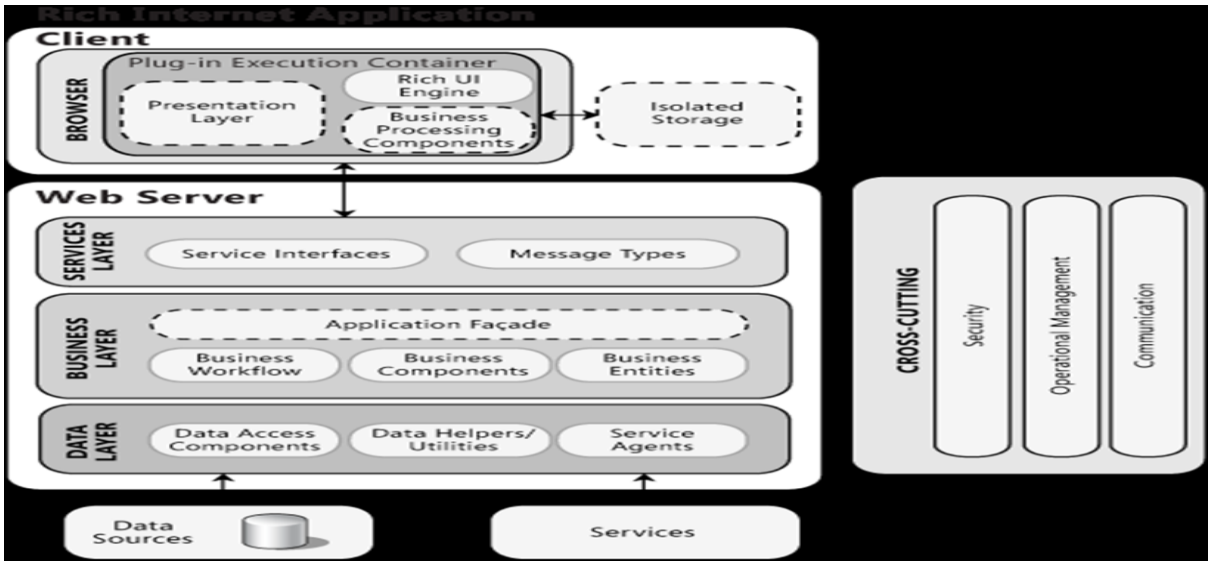

Figure 1.7 Architecture of RIA

A typical Rich Internet Application is decomposed into three layers is shown in Figure 1.7.

- **Presentation layer** - contains UI and presentation logic components
- **Business layer** - contains business logic, business workflow and business entities components
- **Data layer** – Contains data access and service agent components.
- It is common in RIAs to move some of business processing and even the data access code to the client.
- The client may in fact contain some or all of the functionality of the business and data layers, depending on the application scenario.

UI components

- Users can interact with the application
- Format data and render data to users
- Acquire and validate data

Application facade

- To combine multiple business operations into single message-based operation
- The facade can be accessed from the presentation layer using a range of communication technologies.

Data access logic components

- Abstract the logic needed to access the underlying data stores.
- This centralizes data access functionality, makes it easier to configure and maintain.

Data helpers/utilities

- For centralizing generic data access functionality (managing db connection and caching data).
- Data source specific helper components can be designed to abstract the complexity of accessing the db.
- Service Agents – Basic mapping between the format of the data exposed by the service and the format your application needs.
- Business components – implement the business logic of the application. Implement business rules and perform business tasks.
- Business workflows – Define and coordinate long running, multistep business processes. They can be implemented using business process management tools.
- Design considerations
- Choose an appropriate technology based on application requirements.(eg . Windows Forms, OBA, WPF)
- Separate presentation logic from interface implementation

    - Eases maintenance

    - Promotes reusability

    - Improves testability

- Identify the presentation tasks and presentation flows

    - Helps to design each screen and each step-in multi-screen or wizard Processes.

- Design to provide a suitable and usable interface

    - Features like layout, navigation, choice of controls to maximize accessibility and usability.

- Extract business rules and other tasks not related to the interface.
- Reuse common presentation logic – (Libraries that contain templates, generalized client-side validation functions and helper classes)
- Loose couple your client from any remote services it uses.

    - Use a message-based interface to communicate with services located on separate

physical tiers.

- Avoid tight coupling to objects in other layers – Use the abstract base classes or messaging when communicating with other layers of the application.
- Reduce round trips when accessing remote layers – Use coarse grained methods and execute them asynchronously to avoid blocking or freezing the UI.

General design considerations

1.Choose a RIA based on audience, rich interface, and ease of deployment

- Consider designing a RIA when your vital audience is using a browser that supports RIAs.
- If part of your vital audience is on a non-RIA supported browser, consider whether limiting browser choice to a supported version is a possibility.
- If you cannot influence the browser choice, consider if the loss of audience is significant enough to require choice of an alternative type of application, such as a Web application using AJAX.
- With a RIA, the ease of deployment and maintenance is similar to that of a Web application, assuming that your clients have a reliable network connection.
- RIA implementations are well suited to Web-based scenarios where you need visualization beyond that provided by basic HTML.
- They are likely to have more consistent behavior and require less testing across the range of supported browsers when compared to Web applications that utilize advanced functions and code customizations.
- RIA implementations are also perfect for streaming-media applications.
- They are less suited to extremely complex multipage UIs.

2.Design to use a Web infrastructure utilizing services.

- RIA implementations require an infrastructure similar to Web applications.
- Typically, a RIA will perform processing on the client, but also communicate with other networked services.
- For example, to persist data in a database.

3.Design to take advantage of client processing power.

RIAs run on the client computer and can take advantage of all the processing power available there.

Consider moving as much functionality as possible onto the client to improve user experience.

- Sensitive business rules should still be executed on the server, because the client logic can be circumvented.

4.Design for execution within the browser sandbox

- RIA implementations have higher security by default and therefore may not have access to all resources on a machine, such as cameras and hardware video acceleration.
- Access to the local file system is limited.
- Local storage is available, but there is a maximum limit.

5. **Determine the complexity of your UI requirements**.
   - Consider the complexity of your UI. RIA implementations work best when using a single screen for all operations.
   - They can be extended to multiple screens, but this requires extra code and screen flow consideration.
   - Users should be able to easily navigate or pause, and return to the appropriate point in a screen flow, without restarting the whole process.
   - For multi-page UIs, use deep linking methods. Also, manipulate the Uniform Resource Locator (URL), the history list, and the browser's back and forward buttons to avoid confusion as users navigate between screens.
6. **Use scenarios to increase application performance or responsiveness**.
   - List and examine the common application scenarios to decide how to divide and load components of your application, as well as how to cache data or move business logic to the client.
   - To reduce the download and startup time for the application, segregate functionality into separate downloadable components.
7. **Design for scenarios where the plug-in is not installed**.
   - Because RIA implementations require a browser plug-in, you should design for non-interruptive plug-in installation.
   - Consider whether your clients have access to, have permission to, and will want to install the plug-in.
   - Consider what control you have over the installation process.
8. Plan for the scenario where users cannot install the plug-in by displaying an informative error message, or by providing an alternative Web UI.

RIA patterns

- The interaction is the trigger of the RIA pattern.

- Every pattern starts with a user event or a system event, e.g. mouseover, on focus, keyboard stroke or time event.
- A variety of operations can be triggered by the interaction, such as validate, search and refresh.
- Finally, the result of the operation implies an update of the user interface.

RIA pattern – Autocompletion

- User has to deal with a lot of forms and input fields.
- There are forms for analysis parameter, search masks or user registration.
- Unnecessary work would be burdened to the user if he had to fill in every input field.
- Therefore the application should fill in redundant input fields automatically.
- All data captured by application logics could only be suggestions in order to allow a better usability.
- The user is still the last instance for checking the correctness of the suggested data.
- Therefore he should be able to overwrite every auto-completed input field.
- Problem - How could the user of a RIA get an immediate suggestion for values in an input field, after he has entered some initial data or has filled in other related fields?
- Motivation - Assistance by filling in input fields and forms.

(1) People make mistakes when they type.

(2) Typing in data is a tedious work.

(3) Typing speed remains a bottleneck; faster user input is aimed by reducing the number of keystrokes.

(4) Solution - Suggest words or phrases that are likely to complete what the user is typing.

(5) As soon as the user moves to another input element or even as soon as the user inputs a character, the RIA in the background will try to query databases and find relations to already entered data.

(6) If such data could be found and the user has not yet completed it himself, a completed value for the input field is suggested to user.

RIA patterns – autosave

<u>Use When</u>

• composing email messages

• composing web documents (spreadsheet, text)

Possible Pitfalls

• if save takes a while can be disruptive to typing

• can cause interface change (as in gmail) leading to unexpected behavior

Best Practice

• make auto-save as transparent as possible

• email, make it default behavior

• documents, allow it to be turned on

• catch navigation away from page and offer to save

RIA pattern- Busy indicator

Use When

• Need to show that system is processing

• want to show indication in context

Possible Pitfalls

• can be distracting if not necessary

Best Practices

• Place the busy indication as close to the user input as possible

- Use small animated indicator beside input or inside input field

•Place the busy indication at the place where the results will appear

• Don't use too many indicators as it will make for a noisy interface

• Avoid using indicator if delay is really short

RIA pattern – Item selection

pattern. item selection.

Use When

• within a paging context

• need a simple way to provide discontiguous selection

Potential Pitfalls

• confusion between checkbox and clicking in row

• mixing with drag and drop

• handling actions on no selection

Best Practices

• use only within context of paging; not for scrolled content

• combine with a row of buttons or toolbar that operates on the selected items

• use light shading to re-enforce selected state

• avoid using with drag and drop

RIA pattern – live search

Use When

• user needs to search for content and are uncertain on the correct keywords.

Potential Pitfalls

• if results are returned too quick, will be distracting

• if results are not returned quick enough, it will feel sluggish

Best Practices

• start returning results when the user “slows down” typing

• show results below text entry field for feedback

RIA pattern – Refining search

Use When

• user needs to refine a search

• for merchandise search

Potential Pitfalls

• sluggish performance

Best Practices

• place refining criteria to left of results

• use checkboxes for toggling filters

• use sliders for value ranges

• generally avoid sliders for single values (can combine slider & input)

• provide a “show all” to undo refinement

• try to keep criteria above the fold

RESTful Web Services with Nodejs

- REST defines a set of architectural principles by which you can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages.
- Implementation of a REST Web service follows four basic design principles:
- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.
- Transfer XML, JavaScript Object Notation (JSON), or both.

Use HTTP methods explicitly

- This REST design principle establishes a one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods.
- To create a resource on the server, use POST.
- To retrieve a resource, use GET.
- To change the state of a resource or to update it, use PUT.
- To remove or delete a resource, use DELETE.
- A stateful service keeps track of where the application leaves off while navigating the set. In this stateful design, the service increments and stores a previous Page variable somewhere to be able to respond to requests for next.
- In a Java Platform, Enterprise Edition (Java EE) environment stateful services require a lot of up-front consideration to efficiently store and enable the synchronization of session data across a cluster of Java EE containers
- Stateless server-side components, on the other hand, are less complicated to design, write, and distribute across load-balanced servers.
- A stateless service not only performs better, it shifts most of the responsibility of maintaining state to the client application.

Expose directory structure-like URIs

- This type of URI is hierarchical, rooted at a single path, and branching from it are sub paths that expose the service's main areas.
- A URI is not merely a slash-delimited string, but rather a tree with subordinate and superordinate branches connected at nodes.

- For example, in a discussion threading service that gathers topics ranging from Java to paper, you might define a structured set of URIs like this:
- http://www.myservice.org/discussion/topics/{topic}
- The root, /discussion, has a /topics node beneath it. Underneath that there are a series of topic names, such as gossip, technology, and so on, each of which points to a discussion thread. Within this structure, it's easy to pull up discussion threads just by typing something after /topics/.

Transfer XML, JSON, or both

- The objects in your data model are usually related in some way, and the relationships between data model objects (resources) should be reflected in the way they are represented for transfer to a client application.
- In the discussion threading service, an example of connected resource representations might include a root discussion topic and its attributes, and embed links to the responses given to that topic.
- XML representation of a thread

```
<?xml version="1.0"?>
<discussion date="{date}" topic="{topic}">
  <comment>{comment}</comment>
  <replies>
    <reply from="joe@mail.com" href="/discussion/topics/{topic}/joe"/>
    <reply from="bob@mail.com" href="/discussion/topics/{topic}/bob"/>
  </replies>
</discussion>
```

- To give client applications the ability to request a specific content type that's best suited for them, construct your service so that it makes use of the built-in HTTP Accept header, where the value of the header is a MIME type.
- Common MIME types used by RESTful services

MIME-Type
Content-Type

- JSON
  application/json
- XML
  application/xml

- XHTML
  application/xhtml+xml
- Node.js is a software system designed for writing highly-scalable internet applications, notably web servers.
- A complete implementation of hello world as an HTTP Server in Node.js:

```
var http = require('http');

http.createServer(function (request, response) {

response.writeHead(200, {'Content-Type': 'text/plain'});

response.end('Hello World\n');

}).listen(8000);

console.log('Server running at http://localhost:8000/');
```

References

1.Shreeraj Shah, " Web 2.0 Security: Defending Ajax, RIA, and SOA ", Course Technology PTR , 2007.

2. Krishna Sankar , Susan A. Bouchard, " Enterprise Web 2.0 Fundamentals ", Cisco Press , 2010.