sphinx_demo Documentation

dario

Jul 25, 2018

CONTENTS:

CHAPTER ONE

README

3.0 yeaa!!!!

CHAPTER TWO

INTRODUCTION

ChartFactor is a data visualization toolkit that helps create next generation data applications. ChartFactor provides the following benefits: • Rich library of pre-packaged visualizations • Data providers for several data technologies • Extensibility • Clear and concise programming model Let's explore these areas in more detail.

CHAPTER

THREE

RICH LIBRARY OF PRE-PACKAGED VISUALIZATIONS

Data visualizations, even simple bars, can be time consuming to implement when taking into account details such as labels, legends, tooltips, colors, axis display, between others. We also need to consider animation when users interact with their data. ChartFactor not only removes the work of programmatically binding the data to the chart, but also provides sensible defaults for many out-of-the-box visualizations to light up powerful visualizations in minutes.

CHAPTER

FOUR

DATA PROVIDERS FOR MANY DATA TECHNOLOGIES

It is common for modern data environments to include several data source technologies. For example, data for log analysis may reside in ElasticSearch while data obtained from business operations may reside in big data stores such as Impala, Dremio, or behind a BI product like Zoomdata. ChartFactor encapsulates interactions with these technologies behind data provider implementations to enable obtaining aggregate or raw data in a uniform fashion. Developers can also plug in their own data providers if needed. The ChartFactor core components are separate from protocols, applications, and other packages, enabling data provider upgrades and new feature additions to be accomplished faster and with minimal retesting of the entire data application.

CHAPTER

FIVE

EXTENSIBILITY

New data source technologies appear frequently. Visualization libraries are constantly evolving to provide better user experience and performance. ChartFactor allows developers to take advantage of their favorite visualization libraries without limitation. It also allows developers to create their own data providers that use technology-specific query libraries.

CHAPTER

SIX

CLEAR AND CONCISE PROGRAMMING MODEL

Several asynchronous programming models exist to interact with backend systems. Examples are callbacks, promises, and generators. The typical implementation invokes a backend query API and provides a callback function to process results and handle errors. Under this model, it is common to see one callback calling the next callback in a chain that quickly becomes hard to read and maintain. To simplify this model, ChartFactor provides a declarative way to specify data aggregations and how visualizations should be rendered. For example, on the data side, developers can specify how data is grouped, sorted, limited, what metrics are aggregated, what operation is used in the aggregation. On the visualization side, developers can specify the type of chart to render and all its visualization details. This declarations become a "program" that ChartFactor then executes in an efficient way. ChartFactor will query the data, wait for the data to be returned, and render visualizations according to the declarations provided.

CHAPTER

SEVEN

WHAT ARE THE INNOVATIONS THAT CHARTFACTOR BRINGS TO DATA APPLICATION DEVELOPERS

ChartFactor (patent pending) simplifies the development of data applications through the following innovative components: 1) A computational DAG (directed acyclic graph) that allows developers to declare operations for later execution 2) An optimiz

CHAPTER EIGHT

INDICES AND TABLES

- genindex
- modindex
- search