Introduction to ThingWorx

Introduction to Internet of Things (2min)

What are the objectives of this video?

Welcome to ThingWorx (THWX), in this short video we will introduce you to the main components of the ThingWorx platform. After this brief introduction, you can watch other video tutorials showing you the process of creating your own Internet of Things (IoT) Application.

A platform is a software and hardware environment under which various smaller application programs can be developed and run. Facebook is a good example of both a platform and an application. Using Facebook as a platform, developers can create smaller applications. End users benefit from some or all of those applications.

ThingWorx is a platform that lets developers create Internet of Things applications. The application is written to benefit the end user by providing a solution to a problem. For instance, when John Deere puts sensors on tractors, developers can use ThingWorx to create an application to tell the farmer when and where to plow, when to fertilize and when to water.
A typical Internet of Things solution or application created using ThingWorx will include at least one sensor or an aggregating device that captures data from multiple sensors. Both the sensor or the aggregating device can be considered the “Thing”. Another good example is a truck with a temperature sensor that captures an event such as a change in temperature. It sends this data event through a network to a software application that captures that event. An application created with ThingWorx can render this event on a gauge for visualization, store it for continuous monitoring, compare it to certain preset values and/or generate an alert if the value exceeds a certain threshold.

Figure 3
ThingWorx has several important components to help a developer build an IoT application. They include; the Composer, the Mashup Builder, storage, a search engine, collaboration, and connectivity to things.

The Composer is the modeling environment that makes it easy to model the things, business logic, visualization, data storage, collaboration, and security required for a connected application.

The Mashup builder allows developers to build their own dashboard using common components for visualization such as gauges, buttons, lists, charts, tables, maps, blogs and wikis, from a pre-existing library in ThingWorx. These components are called widgets. The mashup environment is similar to the front panel in National Instruments LabVIEW’s virtual instrument.

Storage allows for the collection of massive amounts of data to be stored, analyzed, and shared.

The search engine in ThingWorx is called SQUEAL, which stands for Search, Query, and Analysis. Using SQUEAL, users can correlate and filter data, search system and business records to deliver answers, and speed problem resolution.

Finally, ThingWorx supports connectivity to devices via several methods; including 3rd party device clouds, direct network connections, open APIs, and connectivity using the ThingWorx Edge MicroServer. You can even connect to social media applications such as Facebook, Twitter and Google+ to communicate with people.

Understanding the ThingWorx landscape and a few terms will help you more quickly understand the platform and its capabilities.
On the horizontal top menu bar, the user can search for previously created entities, create a new entity, import or export files to share, monitor, or get help or extra information from the ThingWorx community website. An entity refers to anything created in ThingWorx.

On the vertical left side menu are entity groups. Using entities, the user can create the model and visualize data; manage data storage, collaboration, security, and the system.

Using the Modeling category in the menu, a developer typically starts by creating a new “Entity”; this entity can be a “Thing”, which is the physical device in the field that produces data and sends it to ThingWorx. The thing can also be a sensor aggregating device such as smart trucks or vending machines with sensors, cameras, and/or GPS tracking; a database; or a social network such as Facebook or Twitter.

This thing produces an “Event” when values of its properties change. If the thing has a temperature sensor for example, it will have a Current Temperature property that will trigger an event when the value changes. This event could then be picked up by a subscription to make decisions concerning the device based on the current temperature. For example, it could post a blog entry if the temperature is higher than a certain value. An event could also be a change in location sent by an incorporated GPS sensor whose value is modeled in the thing through a location property.

The “Data Shapes” are built of one or more fields and describe how the data is structured in (custom) Events, InfoTables, Streams, and DataTables. Data Shapes are also considered an Entity.

In order to scale models, connect many Things, and avoid repetitious definition of the property for each new Thing, a developer can streamline this process using additional Entities such as Thing Templates and Thing Shapes.

Let’s say we are tracking a fleet of five distribution trucks and a central application is receiving a continuous stream of data about who the driver is, the truck’s speed and its location. Let’s also suppose that 3 of those trucks are refrigerated trucks and 2 are normal trucks.
Instead of repeating modeling efforts for each truck or each time a truck is added or retired; a **Thing Template** can be used to define the properties of the things of similar nature, so new things will be pointed to the template to inherit all its properties. **Thing Shapes** can also be used to define the properties of things or thing templates. They can also be used to execute functions or perform “Services”.

It’s important to note that a Thing can only inherit properties, services, events etc. from a single Thing Template, but both the Thing and the Thing Template can inherit the properties of more than one Thing Shape.

At a later time the fleet owner may wish to add a temperature and humidity sensors to the refrigerator trucks. This will help ensure that the content of the truck arrives in good condition. To do this, a new refrigerator shape could be created. The refrigerator truck things could inherit the properties of the Refrigerator Thing Shape.
**Thing Shapes** can also be used to execute functions or perform “Services”. As the fleet grows the fleet owner may decide to better manage the maintenance of the trucks. For example, a Thing Shape could inform the software that a truck tires has exceeded its 40K miles and has run for an extra 10K miles or is due for oil change. If a condition changes, such as using new & improved tires with longer life, that is the tires on need to change every 50K instead of 40K, this conditional change can be modified in one place; the Thing Shapes, and the Templates and the Things will inherit and implement the new changes automatically.

So now, when a truck approaches 50K, The Thing, Thing Template, of Thing Shape can produce an **Alert** to inform the system of the need for maintenance. This Alert is a special event programed according to a criterion defined by the user. A subscription to this alert can be used to send an email, or text message to the driver and/or the supervisor through scripts written in javascript.

The use of Thing Templates and Thing Shapes is recommended even for small projects to establish a good best practice for modeling your environment.

After modeling is completed, visualization is required to facilitate monitoring and interpreting the incoming data. Users create their own dashboard or **mashup** using drag and drop **widgets** from an existing library. These are data representations such as charts, gauges, or a navigation link. These widgets are bound to the incoming data. Other widgets can produce an outgoing event such as a button that executes a service when it’s clicked.
Use Figure 6: A Mashup for our truck example can include a map showing where each truck is at any given moment or track the route of one of the trucks. Charts can show the temperature and rpm over time, calculate the average and produce an Alert if temperature exceeds a pre-defined value. A table can show mileage traveled per day and calculate total since last maintenance to plan for the next event. For more examples, please visit the community website to see mashup building video tutorial that are available.

Connecting and Building an Application

In summary, there are three steps to creating an Internet of Things application:

First, download the Edge Micro Server (EMS) to a microcontroller such as Raspberry PI, so ThingWorx can communicate with things in the field such as sensors.

Second, bind the data to establish a communication between the remote thing and ThingWorx. The data will bind to various Thing Shapes.

Third, create the application and use the Mashup builder to build your dashboard for monitoring and control.

ThingWorx is an easy and powerful platform requiring little or no coding/ programming experience. The platform can be used for school projects or scaled up for an industrial application that manages over a million smart, connected things.

There are a lot of resources for further exploring ThingWorx that can be accessed through the community web site or by taking training classes.

Have fun and enjoy connecting your world!