BUILDING AN IOT SOLUTION

Ben Stob
Software Development Manager, ThingWorx®

June 8, 2016

AGENDA

- Scenario: Acme Garden Tractor Co.
- The Model – Best Practices
- Connectivity - Edge Device Communication with ThingWorx®
- Apps – Rapid Application Development
- Creating an Extension using the Eclipse Plugin
- Best Practices for Team Development

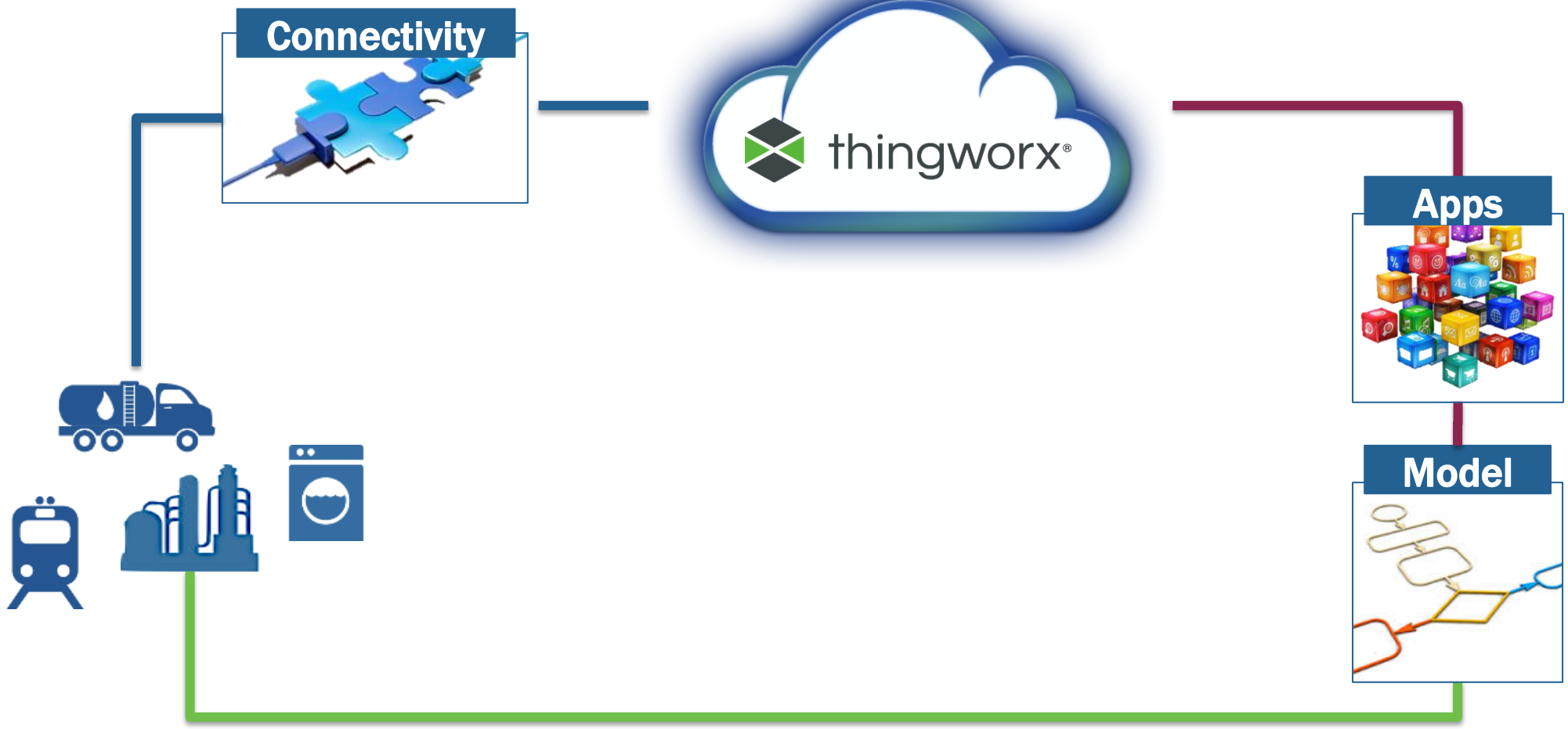
AGENDA

- [x] Scenario: Acme Garden Tractor Co.
- [ ] The Model – Best Practices
- [ ] Connectivity - Edge Device Communication with ThingWorx®
- [ ] Apps – Rapid Application Development
- [ ] Creating an Extension using the Eclipse Plugin
- [ ] Best Practices for Team Development

BUILDING AN IOT SOLUTION - SCENARIO

- Acme Garden Tractor Co. – Provider of residential and commercial lawn tractors
  - Problem Statement
    - Would like a Connected Service offering allowing customers to receive **personalized feedback** about their tractor and upload tractor data to their dealer to **improve customer interaction and satisfaction**.
  - Personas
    - Customers – access to performance data, maintenance information, recall and service notifications, and feedback
    - Dealers – make it easier for customers to find and engage them for service, aggregate reliability information across their customers
  - Use Scenarios
    - Customer takes delivery of their tractor, installs mobile app, and registers the tractor, which instantiates a representation of that tractor in the IoT solution
    - Customer uses tractor for session, tractor establishes secure connection to mobile app and transmits usage data
    - Dealer views usage data across registered customers to identify marketing opportunities and to report on reliability metrics to the manufacturer

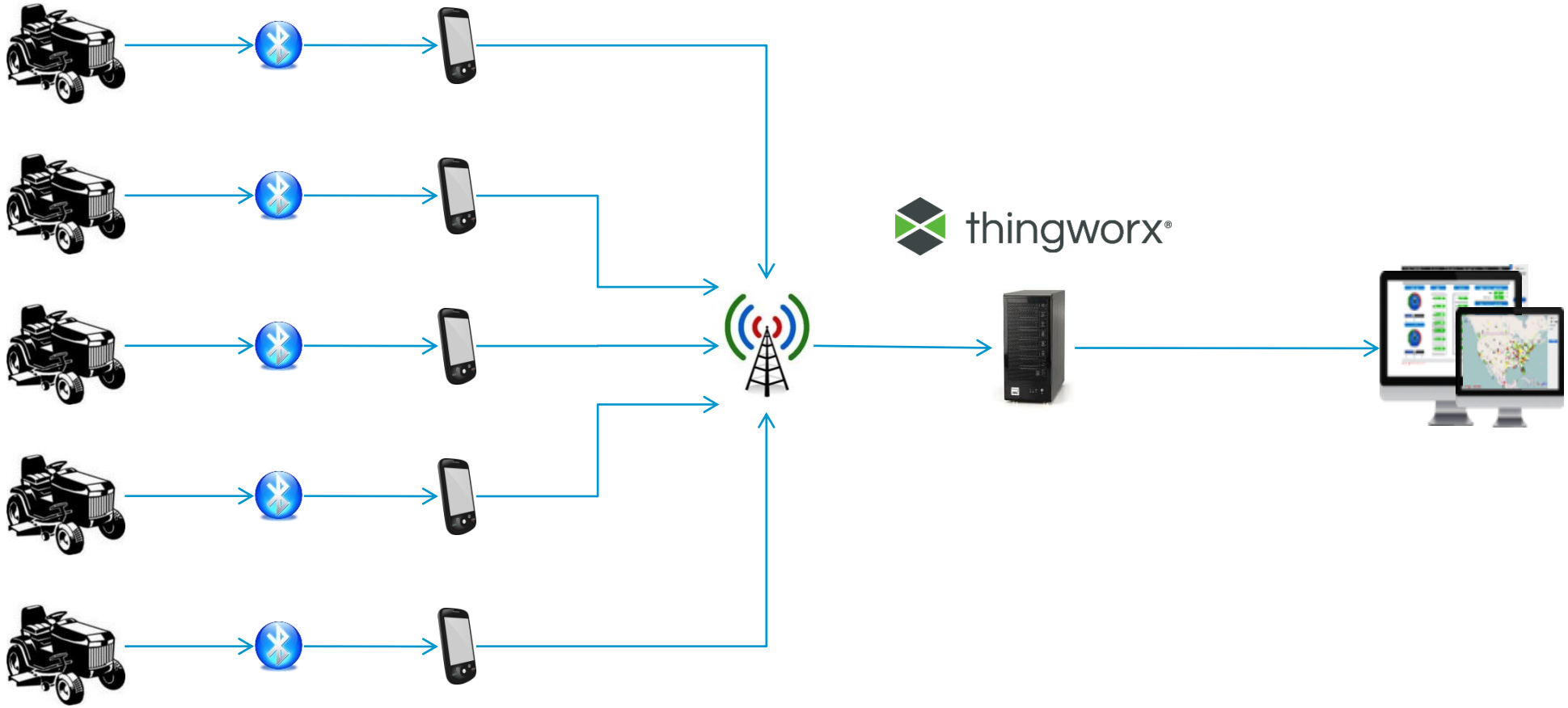
BUILDING AN IOT SOLUTION - SCENARIO



BUILDING AN IOT SOLUTION - SCENARIO

- Acme Garden Tractor Co.
  - Use Bluetooth® to connect to a mobile app and use ThingWorx® SDKs to communicate to ThingWorx® server via HTTPS over carrier network
  - Tractor data:
    - engine hours, engine temperature, peak engine temperature, engine RPM sampling, etc.
  - App data:
    - DIY services performed, e.g. oil change or blades replaced, location

BUILDING AN IOT SOLUTION - SCENARIO



AGENDA

- Scenario: Acme Garden Tractor Co.
- The Model – Best Practices
- Connectivity - Edge Device Communication with ThingWorx®
- Apps – Rapid Application Development
- Creating an Extension using the Eclipse Plugin
- Best Practices for Team Development

BUILDING AN IOT SOLUTION – THE MODEL

- Some basic questions to ask when modeling:
  - How do we represent the real-life device/smart product in ThingWorx®?
  - How can the model remain flexible and maintainable?
  - What is the data that we want to capture? How often?
  - What do we want to do with the data?
    - E.g., trigger alerts, subscribe to events to initiate a downstream process, perform data analysis, etc.
  - What capabilities does ThingWorx® provide?
  - What capabilities do we need to add to ThingWorx® (e.g., via an extension)?

BUILDING AN IOT SOLUTION – THE MODEL

- ThingWorx® Model Best Practices
  - Thursday, June 9 – 9:30am – 10:15am
  - Bob Elam, Product Owner, ThingWorx®
- ThingWorx® Solution Architecture
  - Thursday, June 9 – 10:30am – 11:15am
  - John Schaefer, Senior Vice President, ThingWorx®

BUILDING AN IOT SOLUTION – THE MODEL

- Modeling Basics
  - Objects in ThingWorx® are referred to as “entities”
  - Many of the entity types can have
    - Properties
    - Services
    - Events
    - Subscriptions
    - Configuration

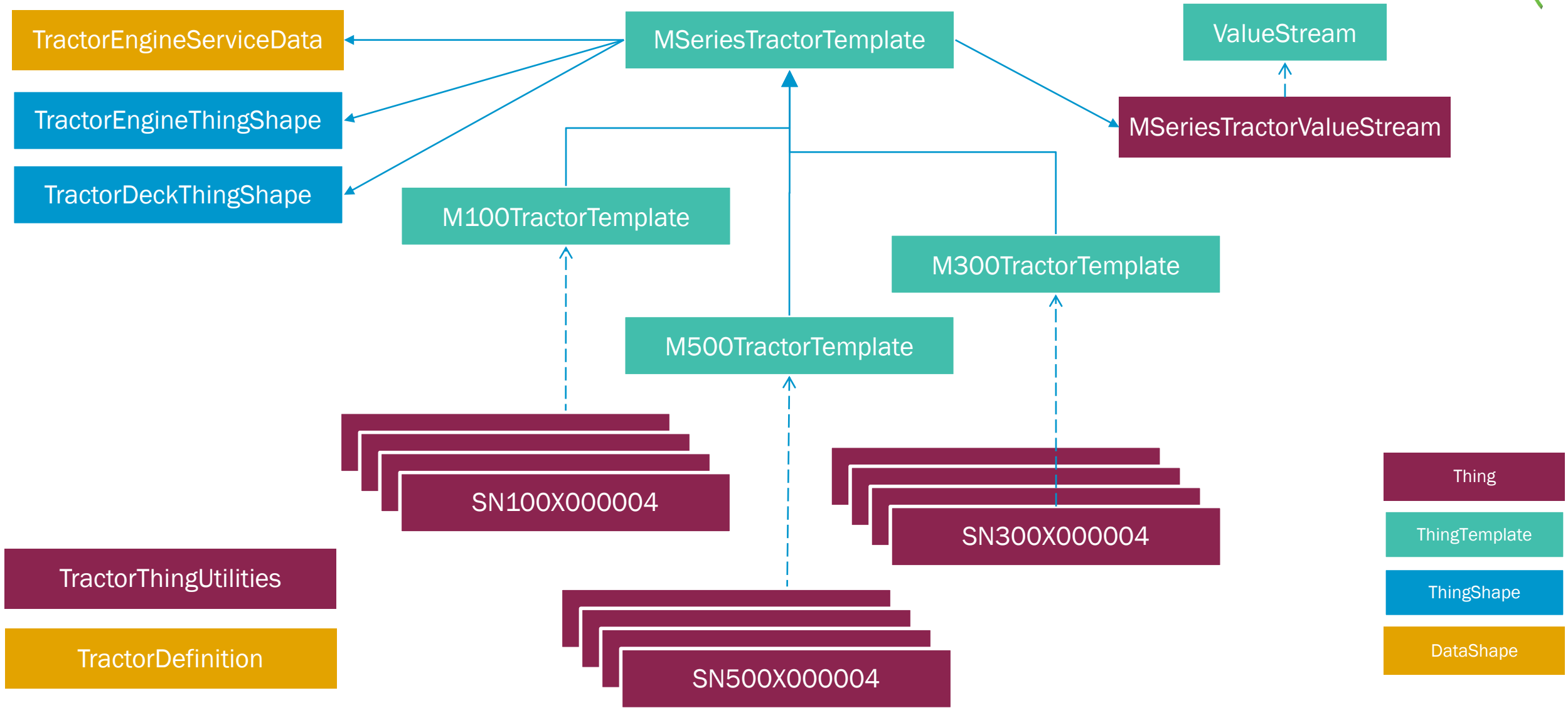
BUILDING AN IOT SOLUTION – THE MODEL

- Thing:
  - An instance of an actual asset/device/real-world physical thing
    - E.g., the specific Acme Tractor sitting in my garage
  - Can extend ThingTemplates
  - Can derive additional characteristics from ThingShapes

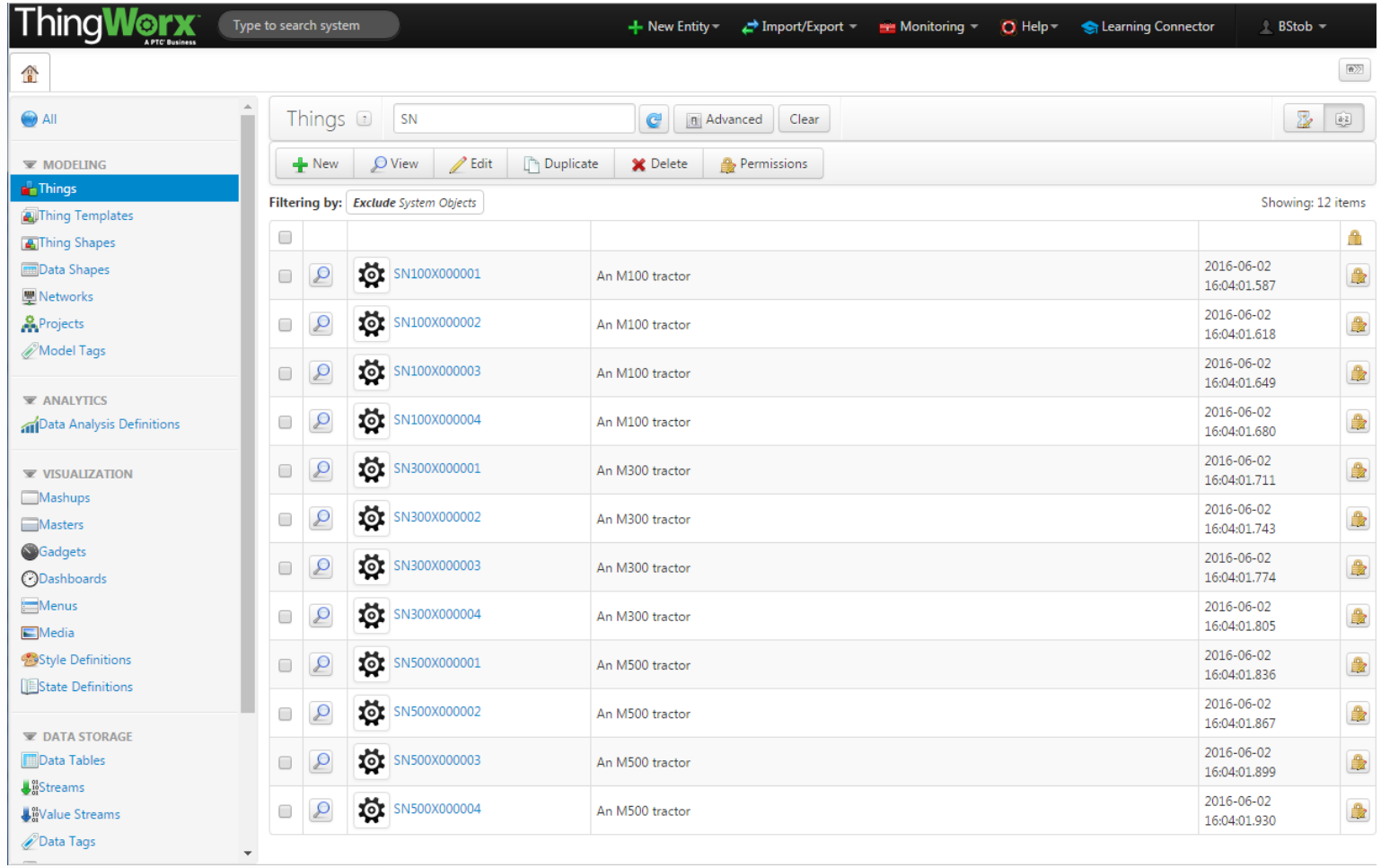
- ThingTemplates:
  - The general definition of the asset/device/real-world physical thing being modeled
    - E.g., all Acme Tractors of a certain model will have the same definition
  - Can extend other ThingTemplates
  - Can derive additional characteristics from ThingShapes

- ThingShapes:
  - Define services, properties, events, that are applicable across different types of Things or ThingTemplates

BUILDING AN IOT SOLUTION – THE MODEL



DEMO – THE MODEL



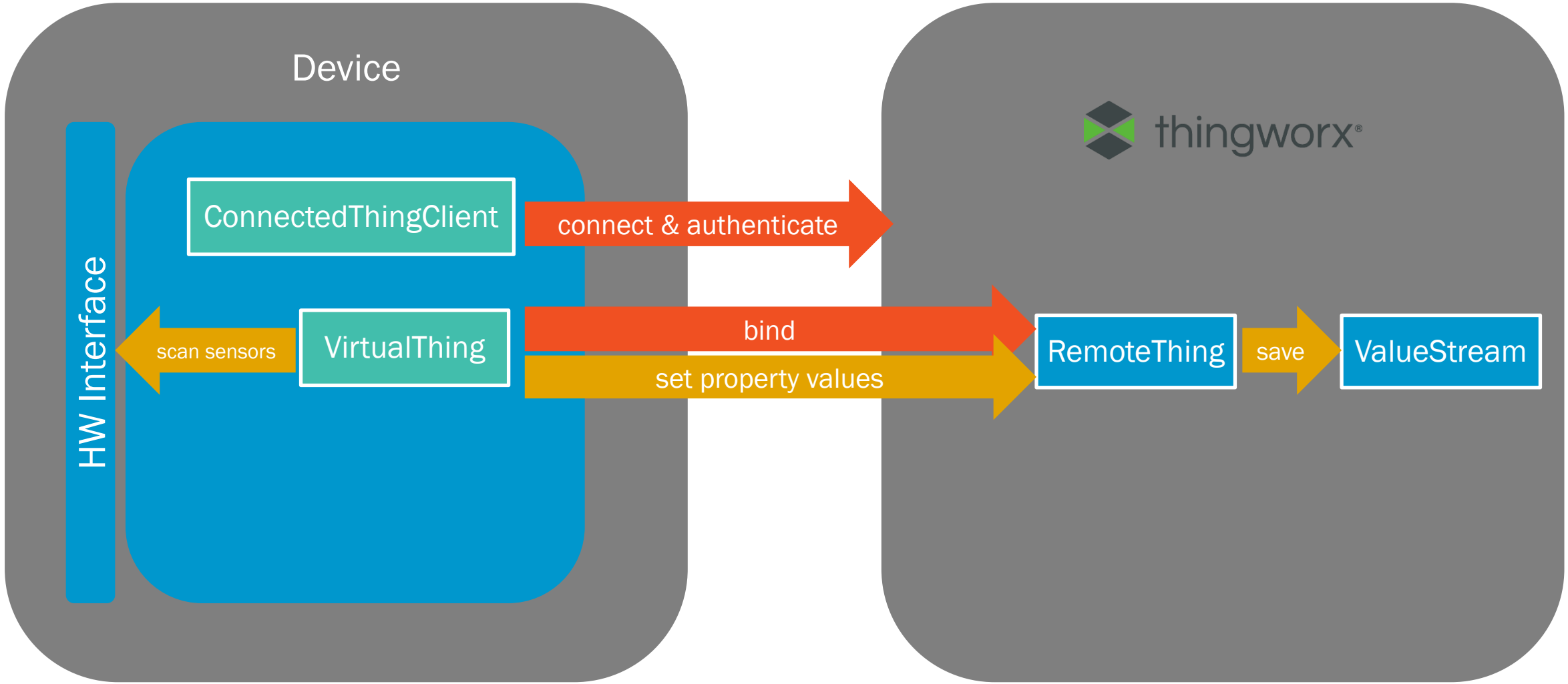
AGENDA

- Scenario: Acme Garden Tractor Co.
- The Model – Best Practices
- Connectivity - Edge Device Communication with ThingWorx®
- Apps – Rapid Application Development
- Creating an Extension using the Eclipse Plugin
- Best Practices for Team Development

BUILDING AN IOT SOLUTION - CONNECTIVITY

- Develop custom application to interface with the device and ThingWorx®
- Options for interfacing to the ThingWorx® Platform:
  - ThingWorx™ WS EMS (WebSocket-based Edge Micro Server)
    - HTTP RESTful APIs
    - Lua Script
  - ThingWorx™ Edge SDK
    - Java, C, .NET, Android, iOS, Protocol Adaptor
- Options for interfacing with device hardware
  - Use hardware-specific libraries

BUILDING AN IOT SOLUTION - CONNECTIVITY



BUILDING AN IOT SOLUTION - CONNECTIVITY

- Model must have a RemoteThing for each device being connected
  - Usually have same name as the VirtualThing on the device, but not required
  - It needs to have RemoteProperties defined that correspond to the properties defined on the VirtualThing
    - These don't need to be the same name but must be bound to each other
- Manually creating remote things for thousands of devices is obviously not practical
  - The best practice is to create service(s) that programmatically create them
  - The service(s) can be invoked by external systems based on the business process for bringing the devices on-line / into production

BUILDING AN IOT SOLUTION - CONNECTIVITY

- On a custom Thing utility entity or on a Java-based Resource entity, create a Service that:
  1. Invokes EntityServices.CreateThing(), passing in the ThingTemplate that
  2. Enables the Thing
  3. Restarts the Thing

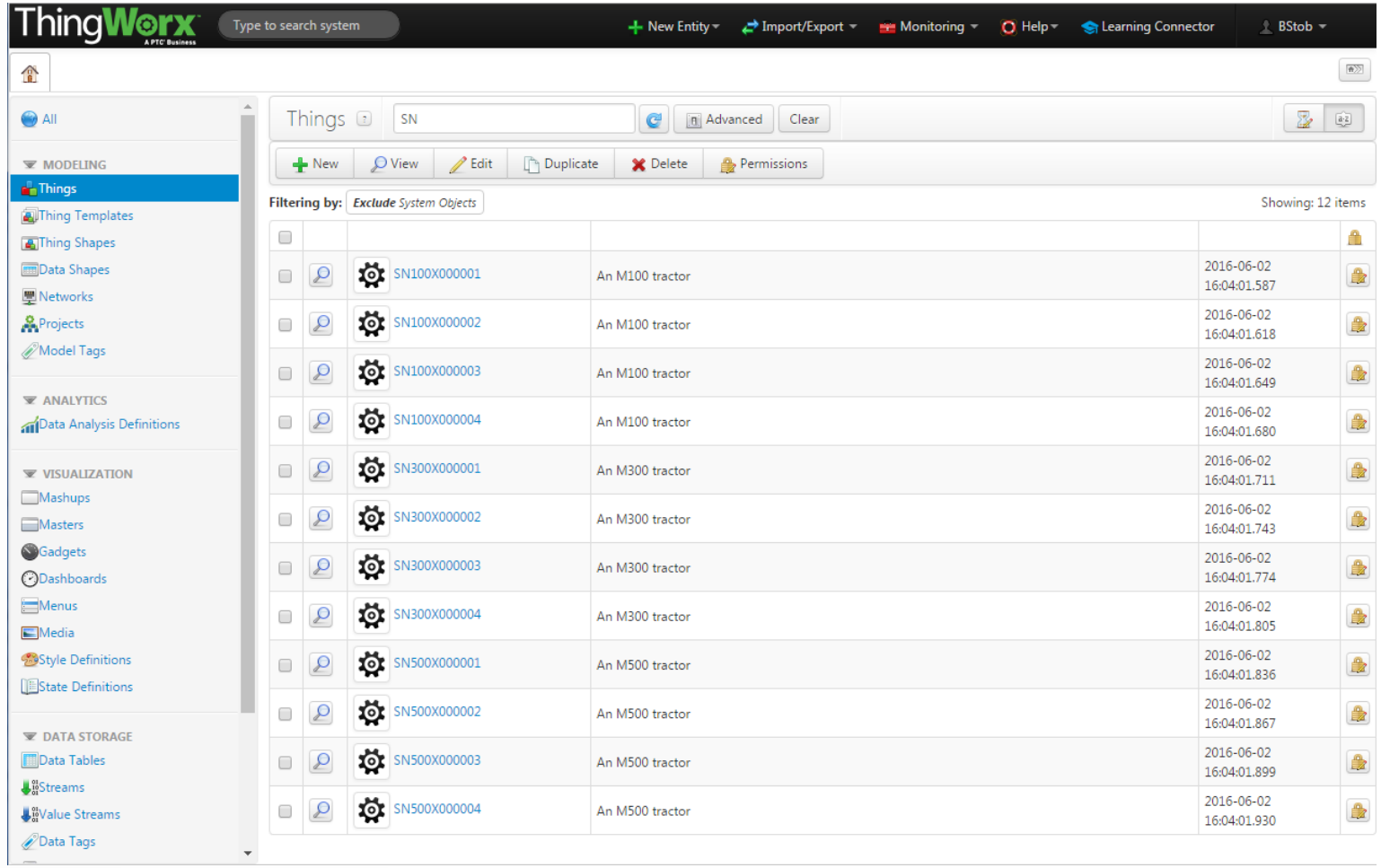
JavaScript snippet:

```
var params = {
   thingTemplateName: "RemoteThing" /* THINGTEMPLATENAME */,
   description: "MyThing"  /* STRING */,
   name: serialNumber /* STRING */
};

Resources["EntityServices"].CreateThing(params);
Things[params.name].EnableThing();
Things[params.name].RestartThing();
```

Java snippet:

```
EntityServices es = new EntityServices();
es.CreateThing(thingName, desc, null, "RemoteThing");
Thing myThing = (Thing) EntityUtilities.findEntity(
   thingName, ThingworxRelationshipTypes.Things);

myThing.EnableThing();
myThing.RestartThing();
```

DEMO – TRACTOR THING CREATOR AND SIMULATOR



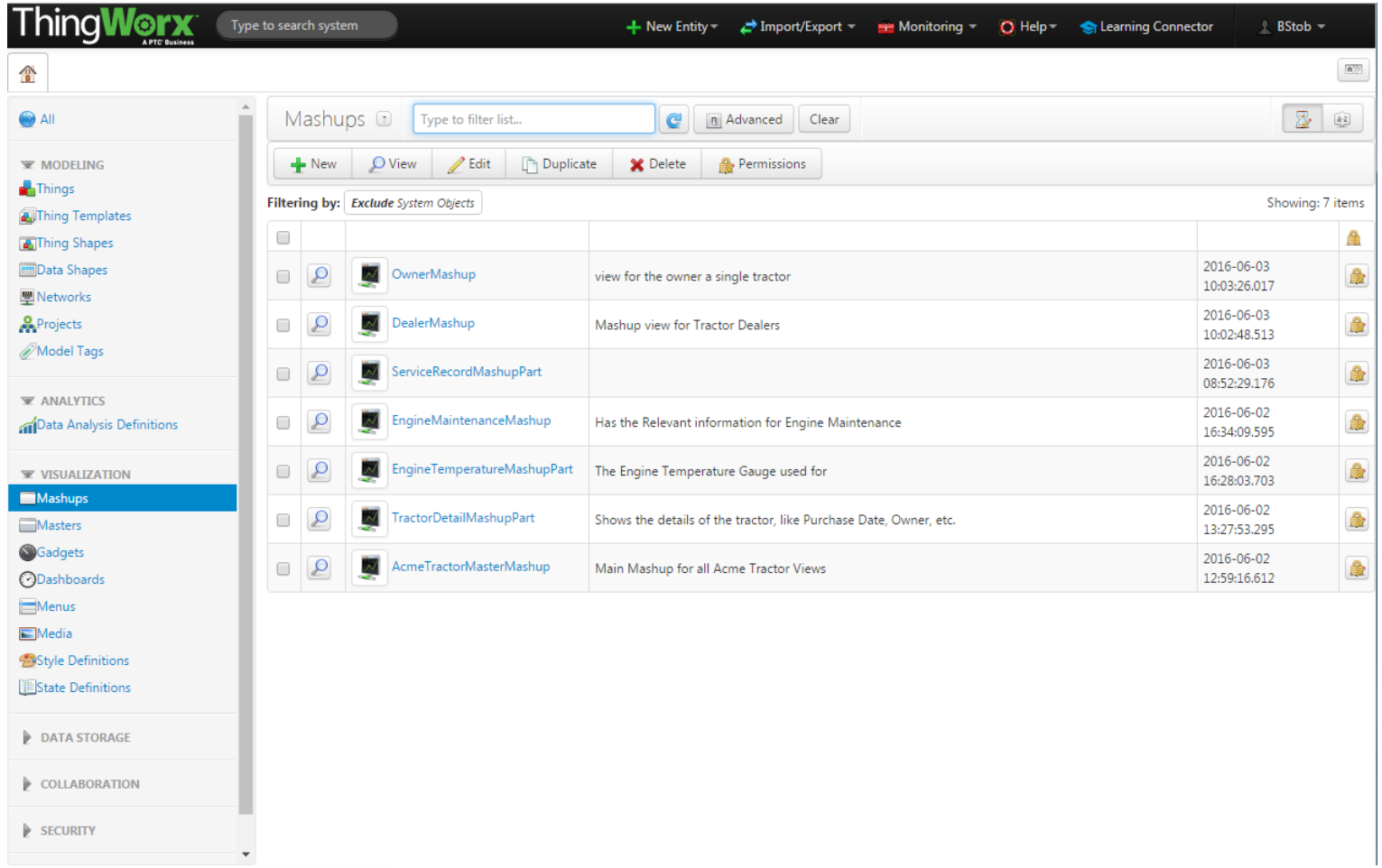
AGENDA

- ❑ Scenario: Acme Garden Tractor Co.
- ❑ The Model – Best Practices
- ❑ Connectivity - Edge Device Communication with ThingWorx®
- ❑ Apps – Rapid Application Development
- ❑ Creating an Extension using the Eclipse Plugin
- ❑ Best Practices for Team Development

BUILDING AN IOT SOLUTION - APPS

- Web apps can be quickly and easily created using the ThingWorx® Mashup and Visualization tools in Composer.
- The Mashup editor provides a rich set of widgets that can be bound to the various services from the model to display data, all without having to do any coding

DEMO – ACME TRACTOR MASHUPS



AGENDA

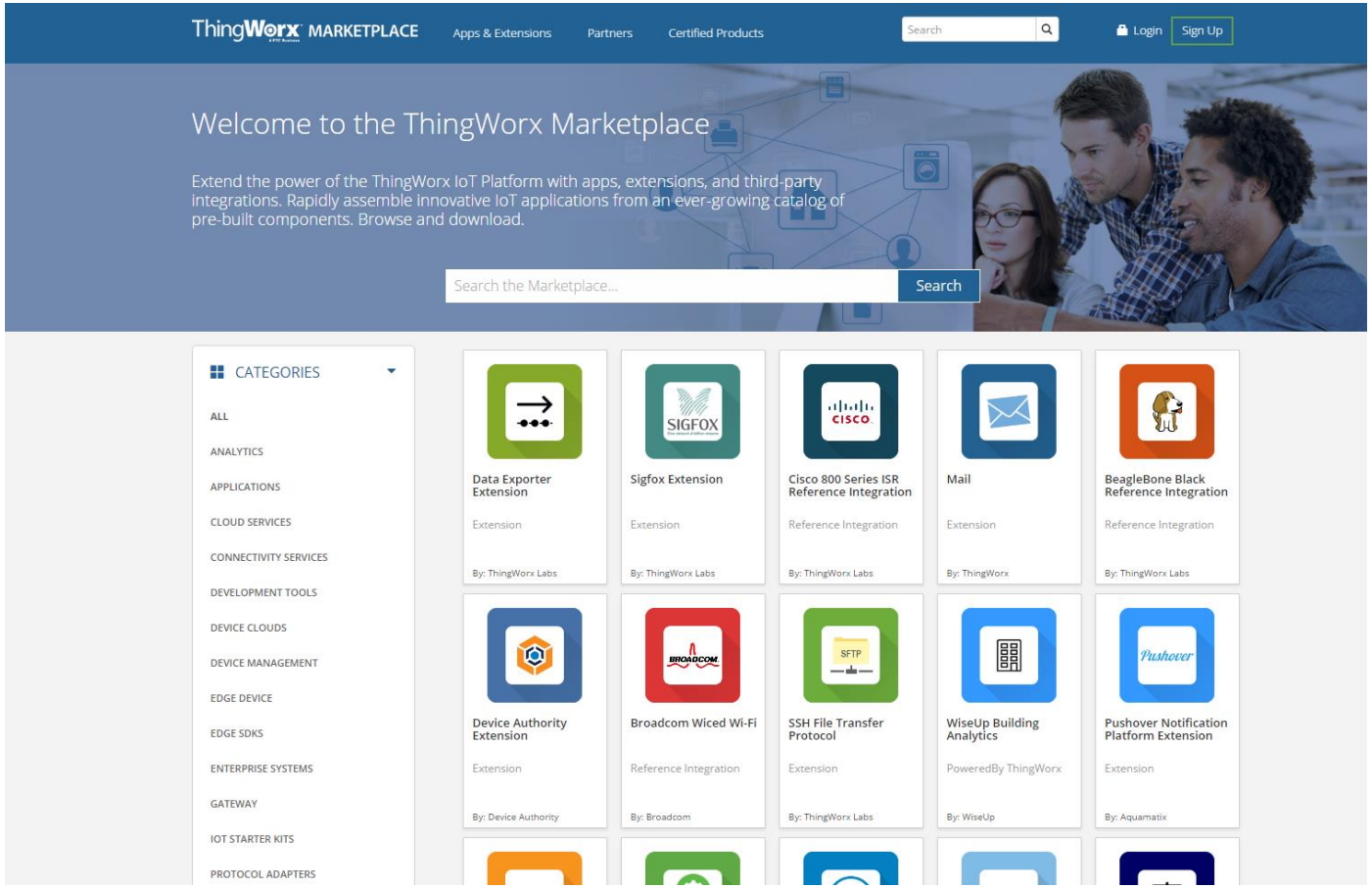
- ❑ Scenario: Acme Garden Tractor Co.
- ❑ The Model – Best Practices
- ❑ Connectivity - Edge Device Communication with ThingWorx®
- ❑ Apps – Rapid Application Development
- ❑ Creating an Extension using the Eclipse Plugin
- ❑ Best Practices for Team Development

BUILDING AN IOT SOLUTION - CREATING EXTENSIONS

- What is an Extension?
  - Collection of entities, resources, and widgets used to expose new functionality in the ThingWorx Platform.
  - Packaged into a .zip file which can be imported to any ThingWorx® Platform
  - E.g., the Mail extension adds the ability to send emails from ThingWorx®
- Why create Extensions?
  - The best practice for building your solution is to do as much as possible in ThingWorx® Composer and Mashup Builder using the tools it offers.
  - In some cases, it's necessary to build extensions to:
    - Use 3rd-party Java libraries not part of the platform
    - Create global service(s) on a custom Resource Java-based entity
    - Build a custom JavaScript widget to use in mashups
    - Build custom authenticators and/or directory services

BUILDING AN IOT SOLUTION - CREATING EXTENSIONS

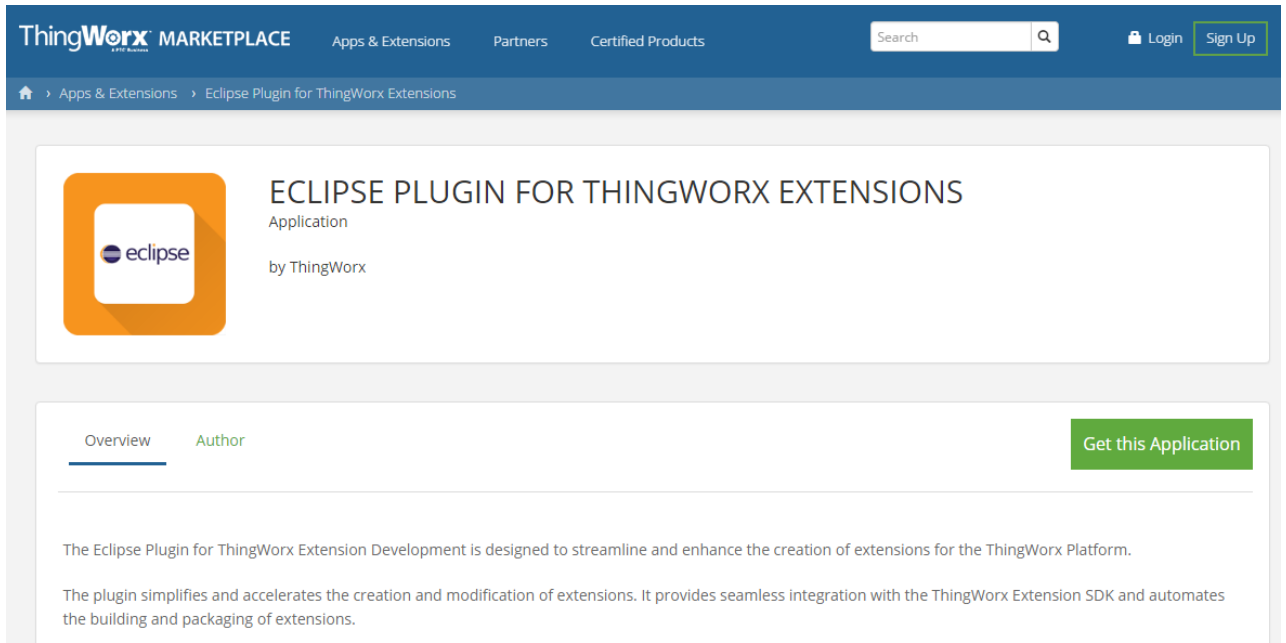
- There is a large collection of Extensions in the ThingWorx® Marketplace:
  - http://marketplace.thingworx.com/



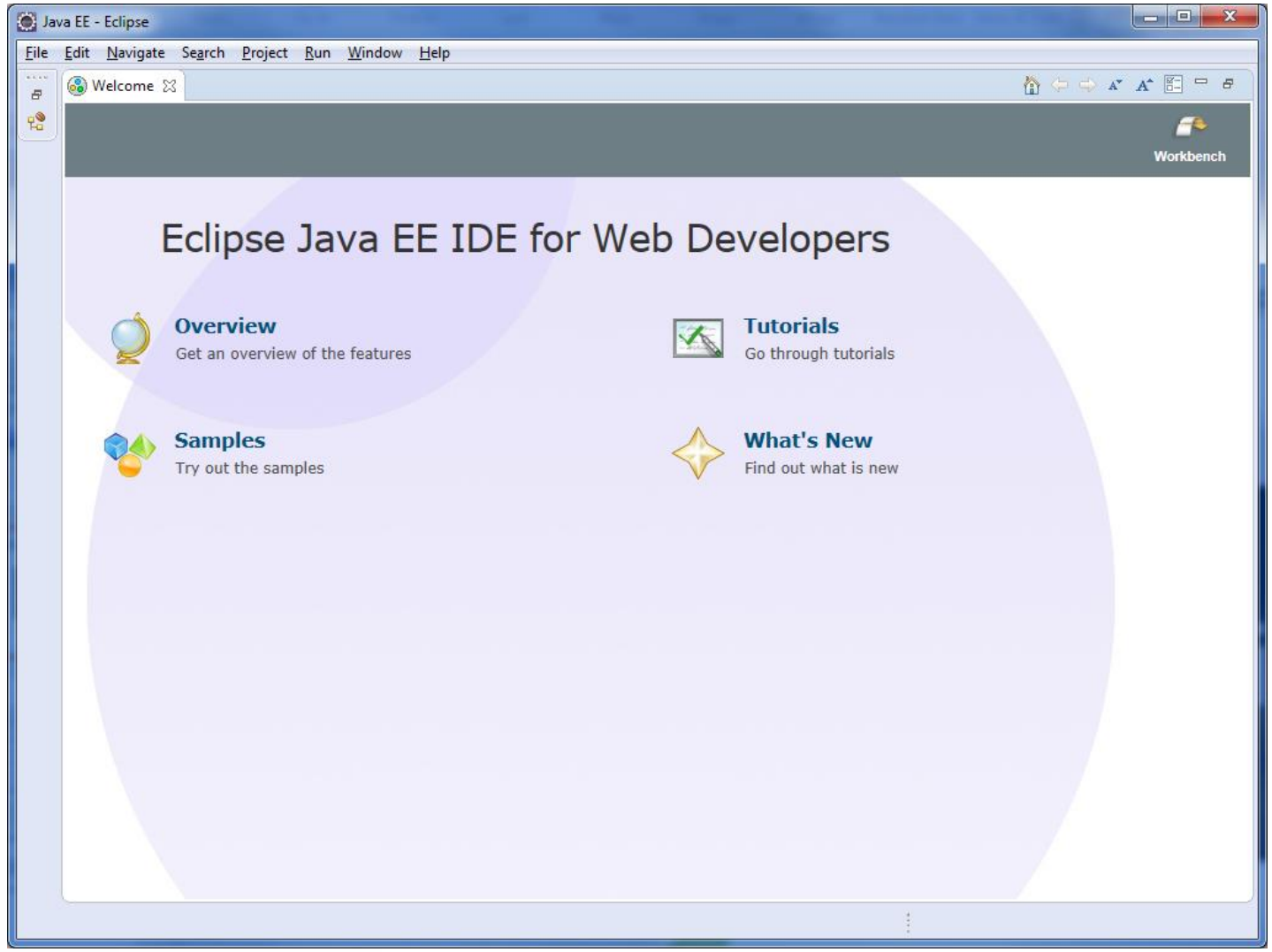
BUILDING AN IOT SOLUTION - CREATING EXTENSIONS

- The following resources are available to help creating extensions:
  - ThingWorx® Extension Development Guide
  - Eclipse Plugin for ThingWorx® Extension
  - Extension SDK

- Available on the ThingWorx Marketplace:
  - http://marketplace.thingworx.com/Items/eclipse-plugin-for-thingworx-extensions



DEMO – ECLIPSE PLUGIN FOR EXTENSION DEVELOPMENT



AGENDA

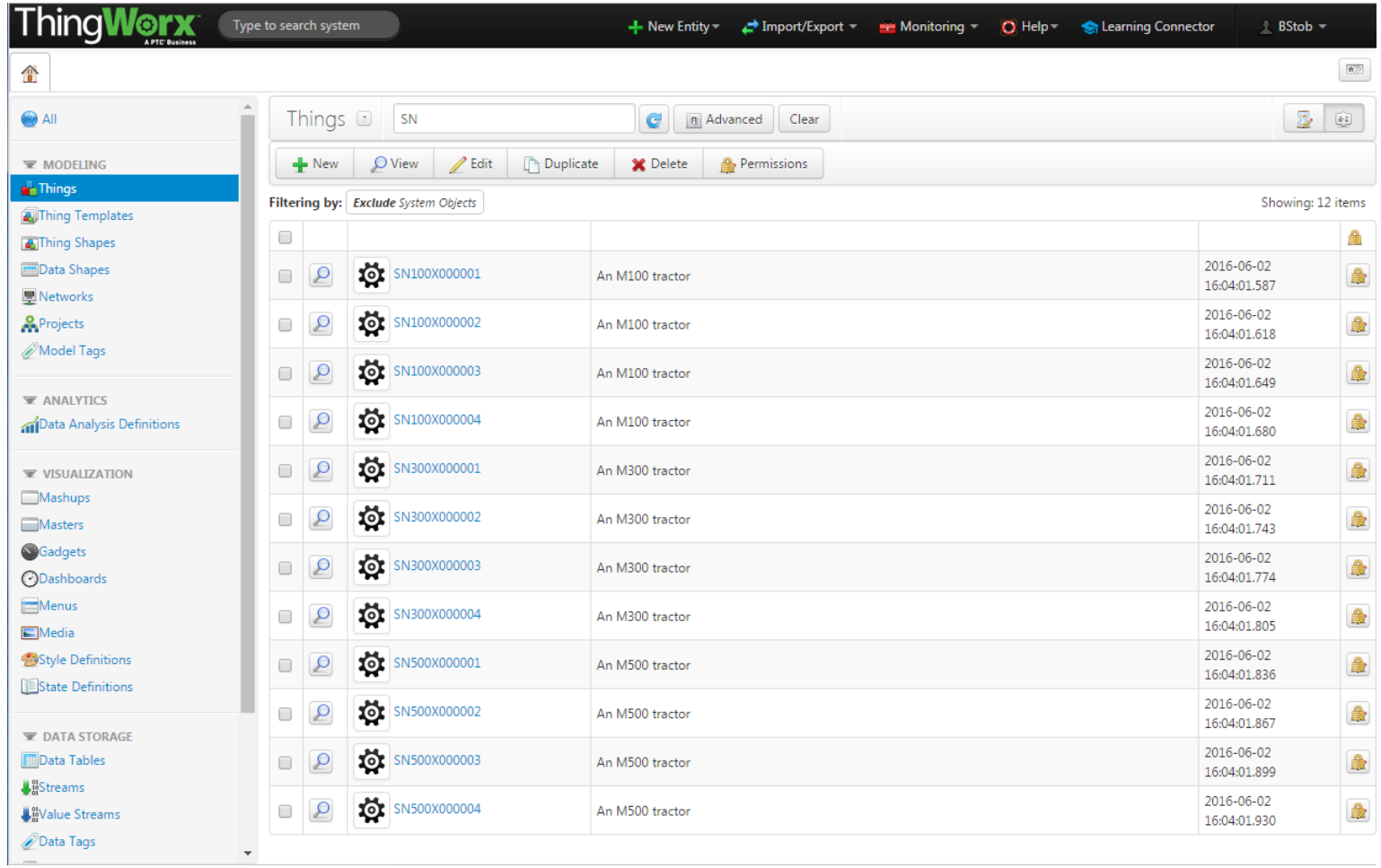
- ❑ Scenario: Acme Garden Tractor Co.
- ❑ The Model – Best Practices
- ❑ Connectivity - Edge Device Communication with ThingWorx®
- ❑ Apps – Rapid Application Development
- ❑ Creating an Extension using the Eclipse Plugin
- ❑ Best Practices for Team Development

- Projects
  - Provide a grouping capability for keeping a collection of entities together
  - Used in the export/import process to
    - Move the application through the landscape (development, QA, production, etc., environments)
    - Upload to source control
  - Add the ability to add dependencies between projects

- Exporting to Source Code Repository
  - Exports organized individual XML files in folders by project and entity type
  - Individual developers should develop against a local ThingWorx instance and import/export from a local directory that is set up for source control
    - Helps to avoid conflicting changes made at the same time on a server
  - Process:
    - Create a Thing using the “SourceControlRepository” Thing Template.
    - Configure the “rootPath” setting of the thing to point to the source control directory
      - Can be anywhere on server’s file system

- Diff Tool
  - During import, can be used to compare the contents of the import with what is already in the platform
  - Based on the differences, the user can select/deselect what to import

DEMO – PROJECTS, EXPORT FOR SOURCE CONTROL, DIFFERENCES TOOL



ADDITIONAL RESOURCES

BUILDING AN IOT SOLUTION – ADDITIONAL RESOURCES

- ThingWorx® Developer Zone
  - http://www.thingworx.com/developer
- ThingWorx® Community
  - https://community.thingworx.com/welcome
- ThingWorx® Marketplace
  - http://marketplace.thingworx.com/
- ThingWorx® Reference Documents
  - https://support.ptc.com/appserver/cs/doc/refdoc.jsp?p=browse_results&Product=ThingWorx
- PTC Support
  - https://support.ptc.com

BUILDING AN IOT SOLUTION – CONTACT US

- We'd love to hear from you!
- Ben Stob – Ben.Stob@thingworx.com
- Dave Larson - Dave.Larson@thingworx.com
- Jesse Docken - Jesse.Docken@thingworx.com
- Thearon Helgeson – Thearon.Helgeson@thingworx.com

QUESTIONS


LIVE
WORX16™
TAKE A FRESH LOOK AT THINGS
liveworx.com