IBM Software Group

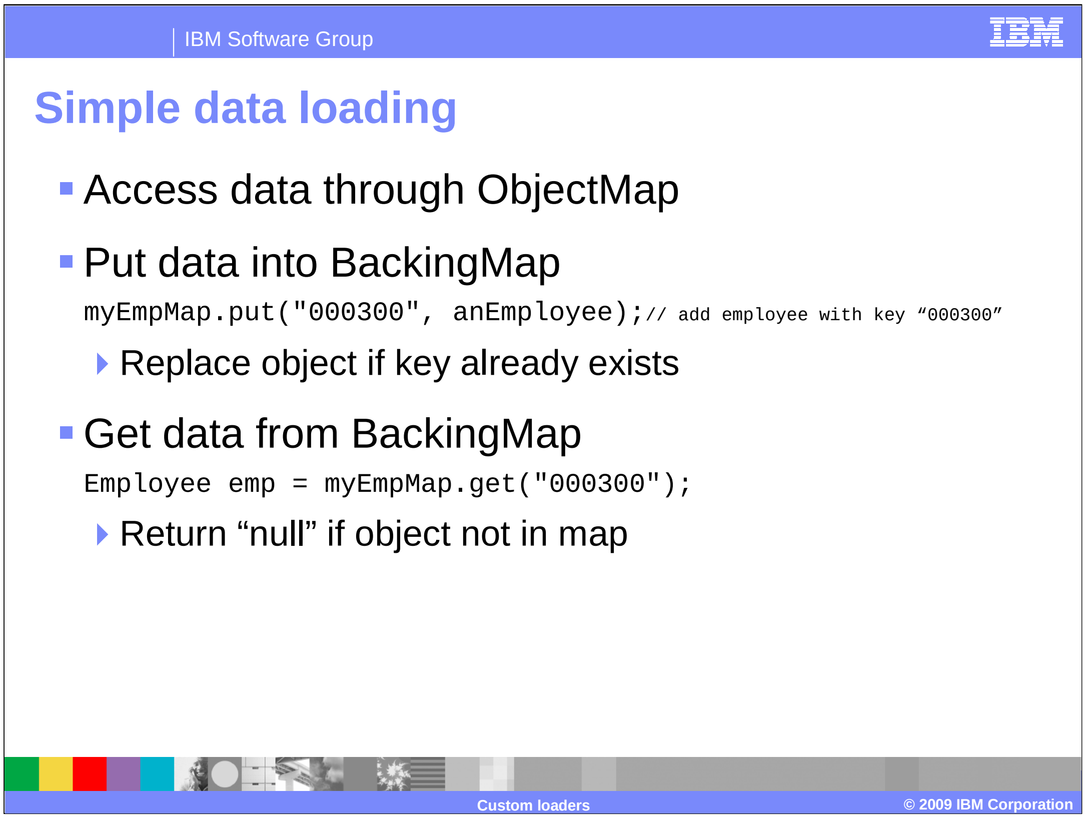
IBM WebSphere eXtreme Scale V7.0

Custom loaders

business on demand.

Updated July 7, 2009

WebSphere® eXtreme Scale provides the ability to populate the grid at start up (called preloading) or to perform dynamic access to the database at runtime through custom loaders.



IBM Software Group

Simple data loading

- Access data through ObjectMap
- Put data into BackingMap

```
myEmpMap.put("000300", anEmployee);// add employee with key “000300”
```

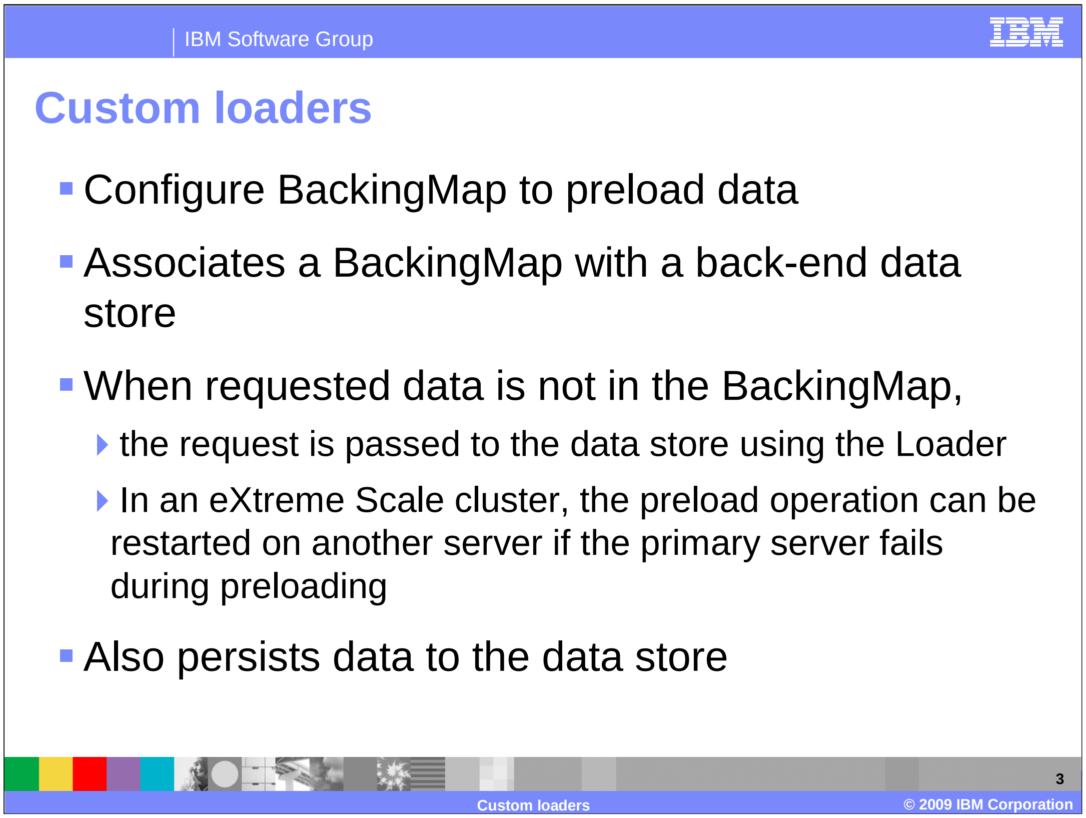
  - Replace object if key already exists
- Get data from BackingMap

```
Employee emp = myEmpMap.get("000300");
```

  - Return “null” if object not in map

Custom loaders

In the simplest case, an application loads data into an ObjectMap using the “put” method, and retrieves the data with the “get” method. If an application calls the ObjectMap’s “get” method with a key that does not exist in the BackingMap, the ObjectMap returns “null”. The application must then retrieve the data from an alternate source and add it to the map so it will be there the next time it is requested. This greatly increases the complexity of the eXtreme Scale client.



Custom loaders

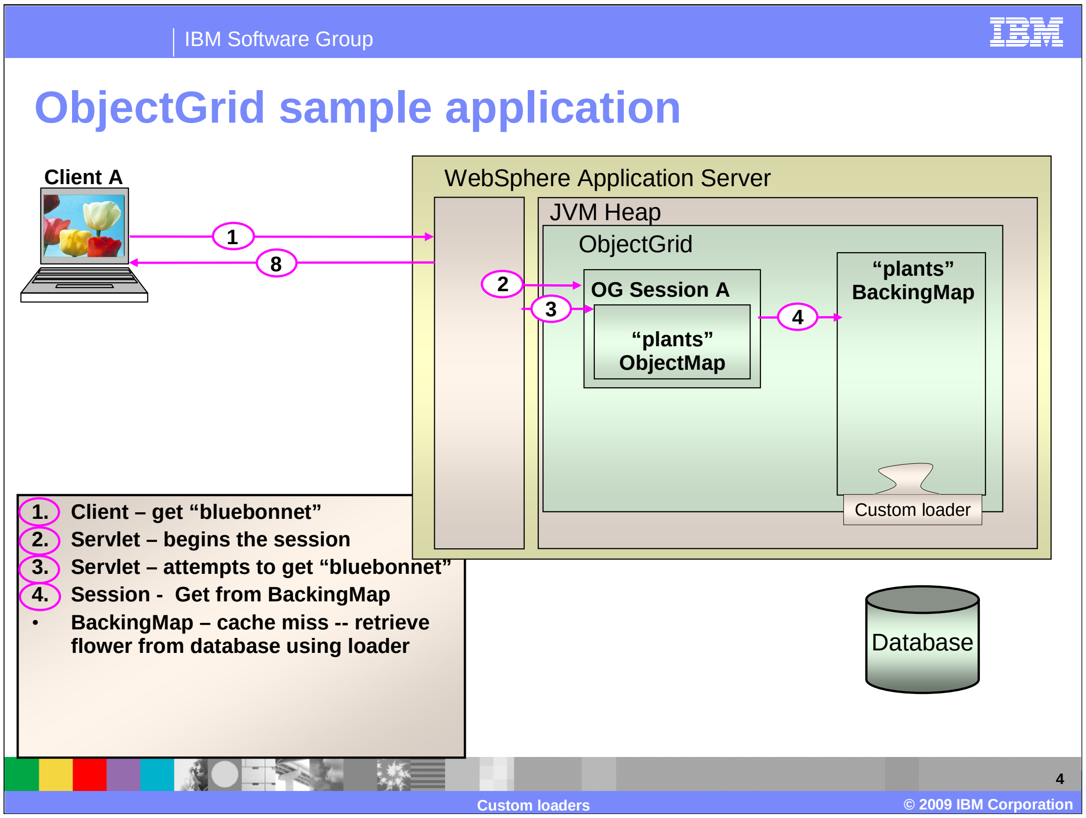
- Configure BackingMap to preload data
- Associates a BackingMap with a back-end data store
- When requested data is not in the BackingMap,
  - the request is passed to the data store using the Loader
  - In an eXtreme Scale cluster, the preload operation can be restarted on another server if the primary server fails during preloading
- Also persists data to the data store

By creating a custom loader object, you can easily preload data into a BackingMap during map initialization. The custom loader sits between the BackingMap and a back-end data store. The back-end data store can be a simple file in the file system, a hardened enterprise database, or even a random data generator. The loader can preload an entire data store, or a subset of data most likely to be requested by applications.

After the map is initially loaded, an application can call the ObjectMap’s “get” method with a key that does not exist in the BackingMap. Rather than returning “null”, the BackingMap passes the request to the custom loader. The Loader will then retrieve the data from the backing store.

By using a custom loader, the eXtreme scale client only has to use a single data access framework – the eXtreme Scale Grid. This results in simpler, more flexible, and more maintainable code.

Also, when data is inserted or modified in the grid, the BackingMap uses the Loader class to persist data to the data store.



IBM Software Group

ObjectGrid sample application

1. Client – get “bluebonnet”
2. Servlet – begins the session
3. Servlet – attempts to get “bluebonnet”
4. Session - Get from BackingMap

- BackingMap – cache miss -- retrieve flower from database using loader

4

Custom loaders

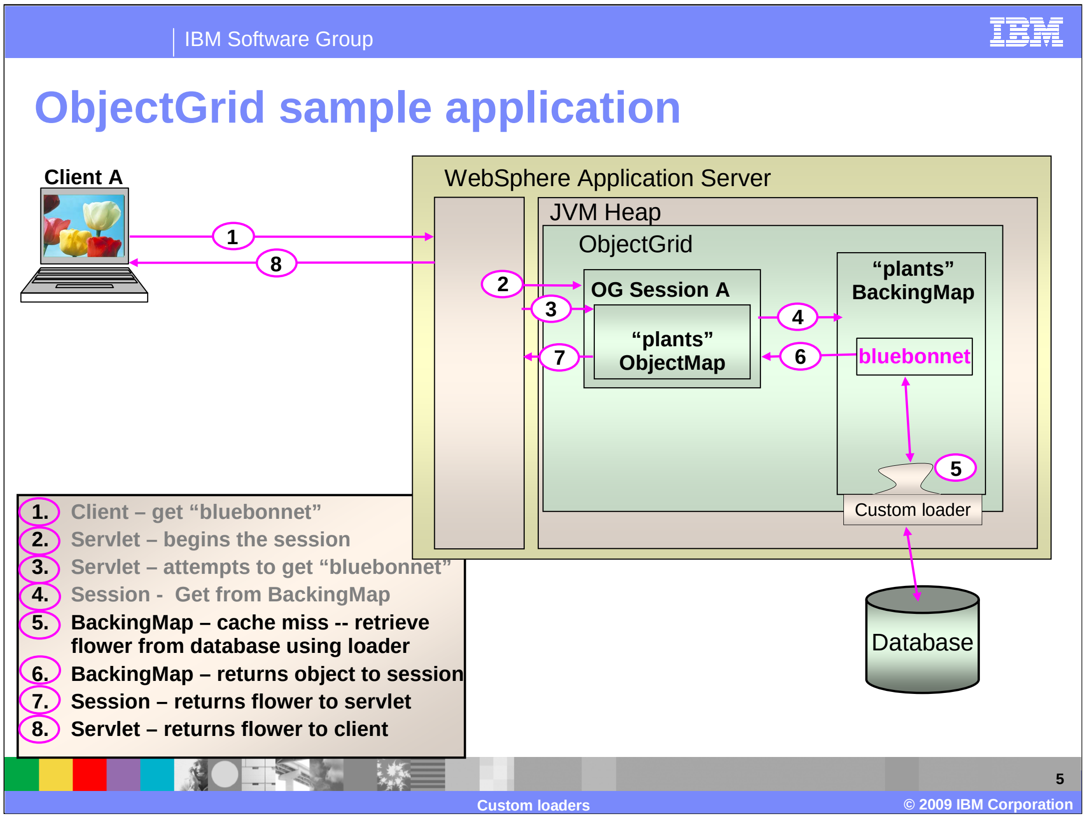
This example shows the general flow when an application requests an object that is not in the grid.

In this example, a client browser sends a request to a servlet to view details for a flower object.

The servlet begins an ObjectGrid session.

The servlet attempts to “get” the flower from the ObjectGrid.

In this case, the requested flower does not exist in the BackingMap. There can be several reasons for this. For example, the object was not preloaded during map initialization, or was previously loaded but evicted to make room for another object.



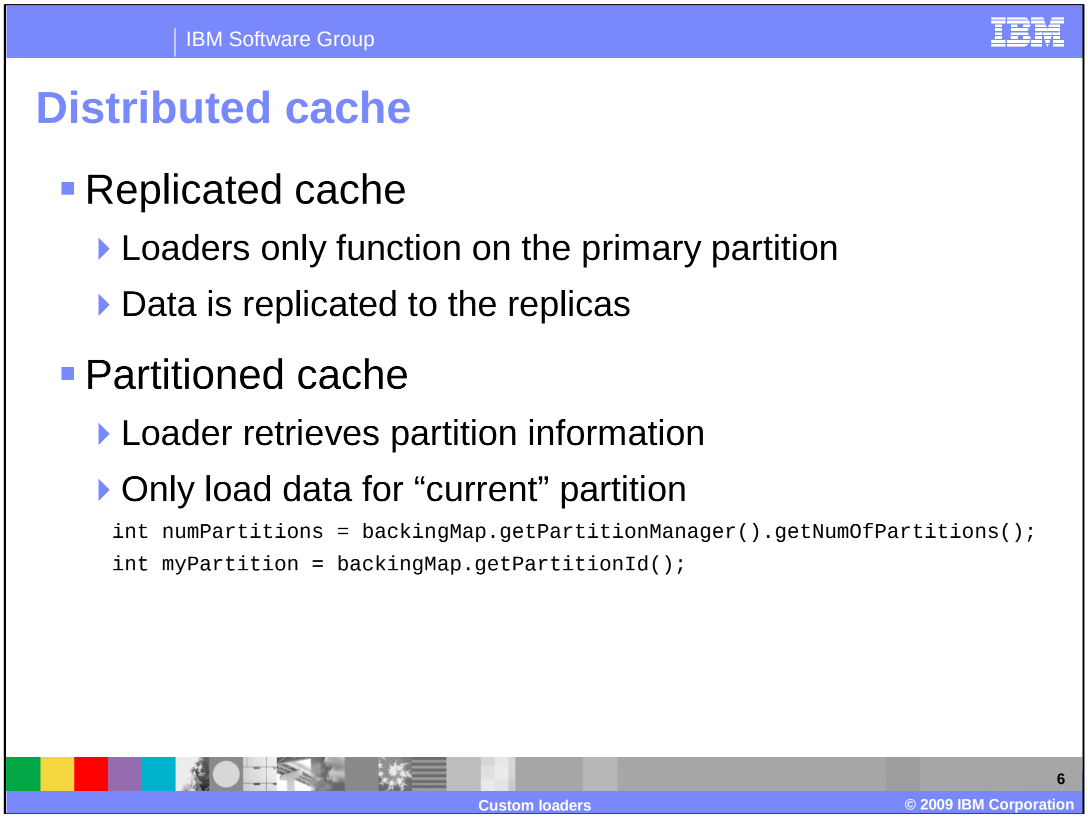
ObjectGrid sample application

1. Client – get "bluebonnet"
2. Servlet – begins the session
3. Servlet – attempts to get "bluebonnet"
4. Session - Get from BackingMap
5. BackingMap – cache miss -- retrieve flower from database using loader
6. BackingMap – returns object to session
7. Session – returns flower to servlet
8. Servlet – returns flower to client

The BackingMap forwards the request for the flower to the configured custom Loader, which issues an SQL "Select" to retrieve the flower's details from the database. The Loader constructs the plant object from the data returned and adds the flower to the BackingMap.

The BackingMap can now return the flower to the session, and the session to the servlet.

If another client requests details for the same flower object, the Loader is not used as the object is already in the grid.



IBM Software Group

Distributed cache

- Replicated cache
  - Loaders only function on the primary partition
  - Data is replicated to the replicas
- Partitioned cache
  - Loader retrieves partition information
  - Only load data for “current” partition

```
int numPartitions = backingMap.getPartitionManager().getNumOfPartitions();
int myPartition = backingMap.getPartitionId();
```

6

Custom loaders

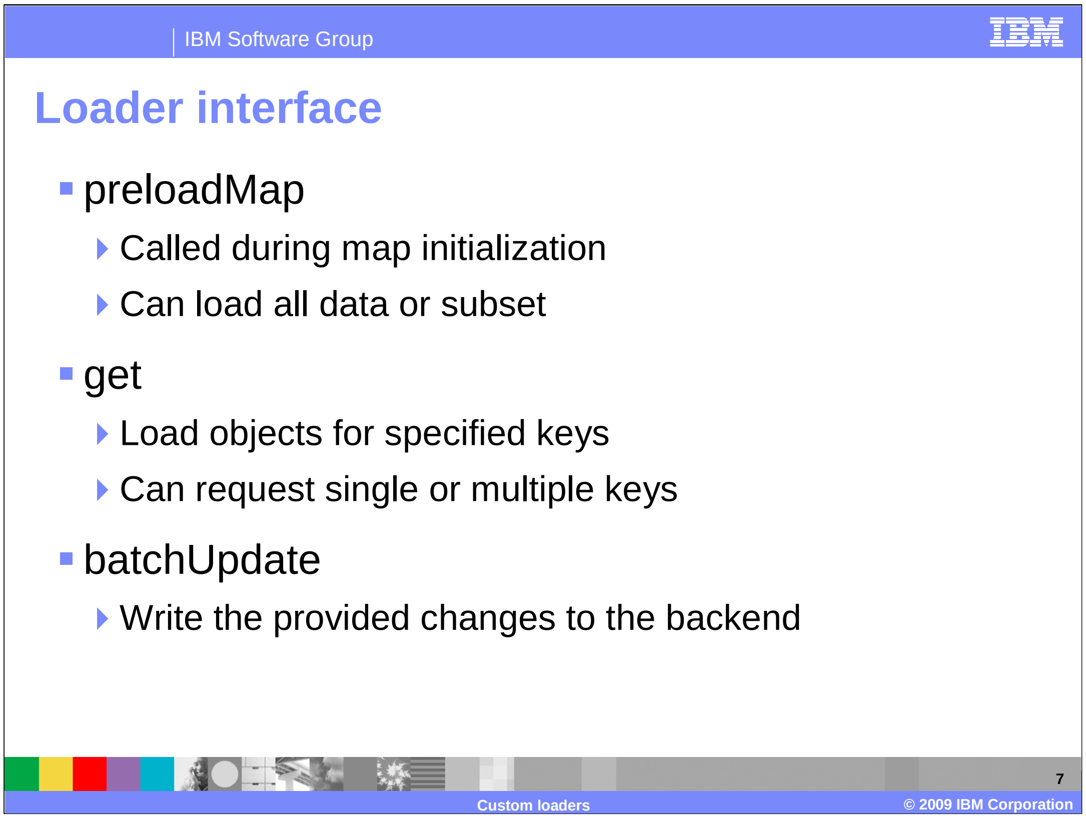
© 2009 IBM Corporation

In a distributed eXtreme Scale environment, a map can have replicas and might preload a large volume of data during initialization. In this configuration the loader only runs in the primary replica. As data is loaded it is distributed to replicas using the eXtreme Scale replication mechanism. This lowers the load on the back-end data store significantly as only one server is accessing the back-end.

A very large map can be partitioned, with data striped across multiple servers. Each entry is stored only on one of those servers. If the grid is partitioned, the loader must identify the subset of the data that it will preload.

This code example shows how an application can identify the subset of the data to load from the database. Applications should always use these methods even when the map is not initially partitioned. If the map is later partitioned by the administrators, then the loader will continue to work correctly.

The application must issue queries to retrieve the myPartition subset from the back-end. If a database is used, then it might be easier to have a column with the partition identifier for a given record unless there is some natural query that allows the data in the table to partition easily.



IBM Software Group

Loader interface

- preloadMap
  - Called during map initialization
  - Can load all data or subset
- get
  - Load objects for specified keys
  - Can request single or multiple keys
- batchUpdate
  - Write the provided changes to the backend

7

Custom loaders

© 2009 IBM Corporation

An application-provided custom loader class must implement the objectgrid Loader interface.

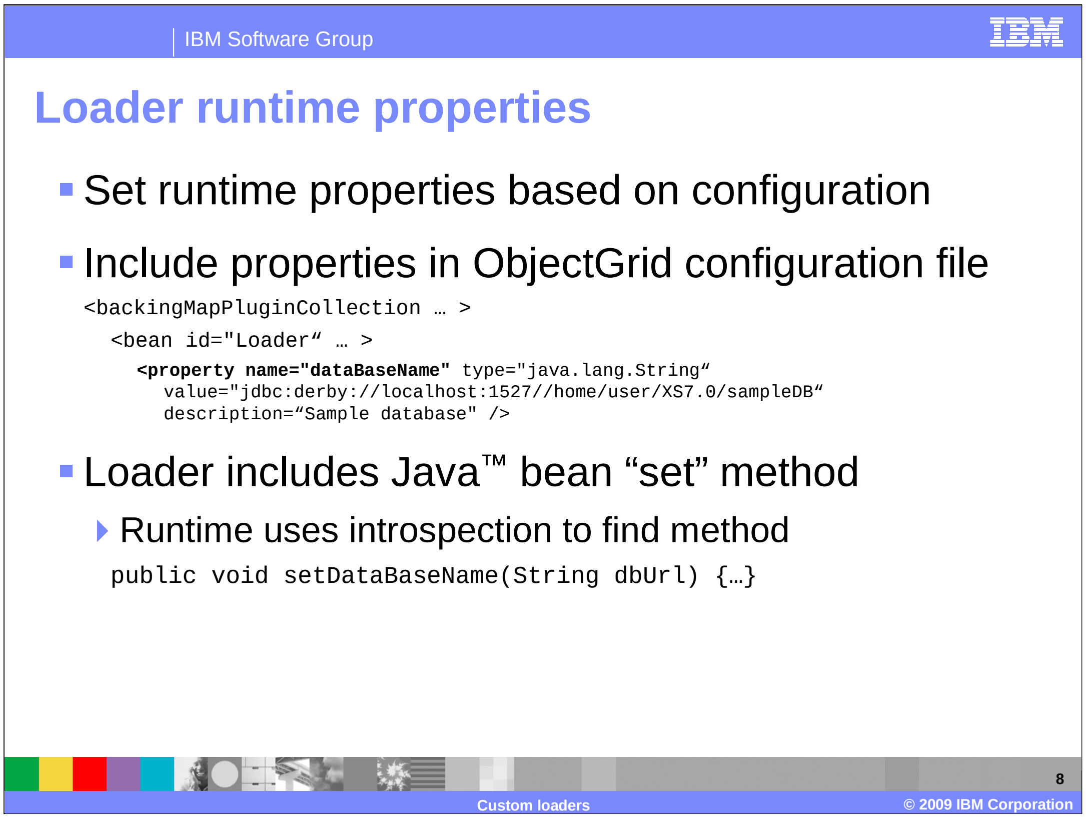
The “preloadMap” method is called during map initialization. It can load all the entries for a partition, or only the subset of data most likely to be requested by clients.

The “get” method is called when a client requests data for a key that does not exist in the grid. The “get” method receives a list of keys as an input parameter, and loads objects corresponding to those keys into the grid.

The batchUpdate” method is called whenever the ObjectGrid needs to apply a set of changes to the back-end data store.

A backing map can have only one loader and a client backing map (or near cache) cannot have a loader.

A loader definition can be applied to multiple backing maps, but each backing map has its own loader instance.


IBM Software Group

Loader runtime properties

- Set runtime properties based on configuration
- Include properties in ObjectGrid configuration file

```
<backingMapPluginCollection … >
  <bean id="Loader“ … >
    <property name="dataBaseName" type="java.lang.String“
      value="jdbc:derby://localhost:1527//home/user/XS7.0/sampleDB“
      description=“Sample database" />
```

- Loader includes Java™ bean “set” method
  - Runtime uses introspection to find method

```
public void setDataBaseName(String dbUrl) {…}
```

8

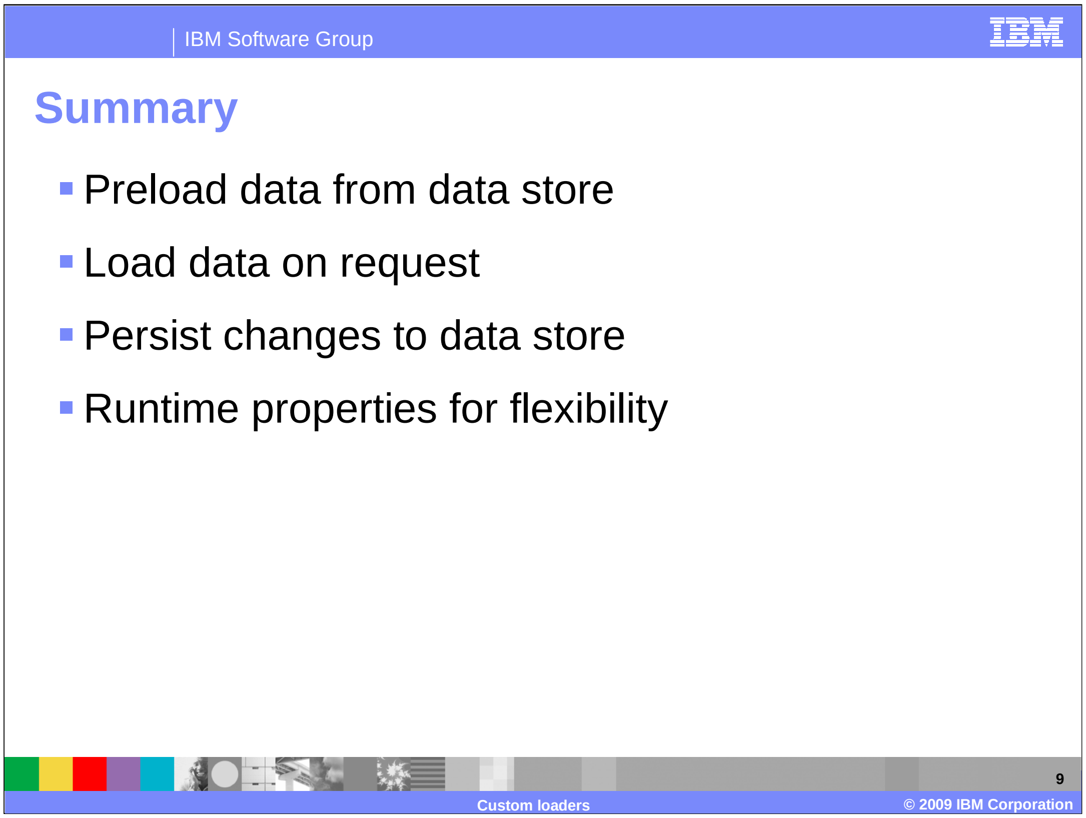
Custom loaders

© 2009 IBM Corporation

In your application-provided Loader, you might need to set one or more Loader properties to modify the Loader’s runtime behavior. For instance, if the Loader is retrieving data from a relational database, it will need the name of the database and the SQL isolation level. To use properties “dataBaseName” and “isoLationLevel”, the loader must include methods named “setDataBaseName” and “setIsolationLevel”. The grid runtime uses Java introspection to call these methods during loader initialization.

Using runtime properties allows administrators to easily modify a grid’s configuration without requiring code changes.

Parameters can be Java primitives, their java.lang counterparts, and java.lang.String. The name and type attributes must correspond to a method signature of the Loader. For example, if a property name is “size” and its type is “int”, then the Loader must include a method setSize(int).

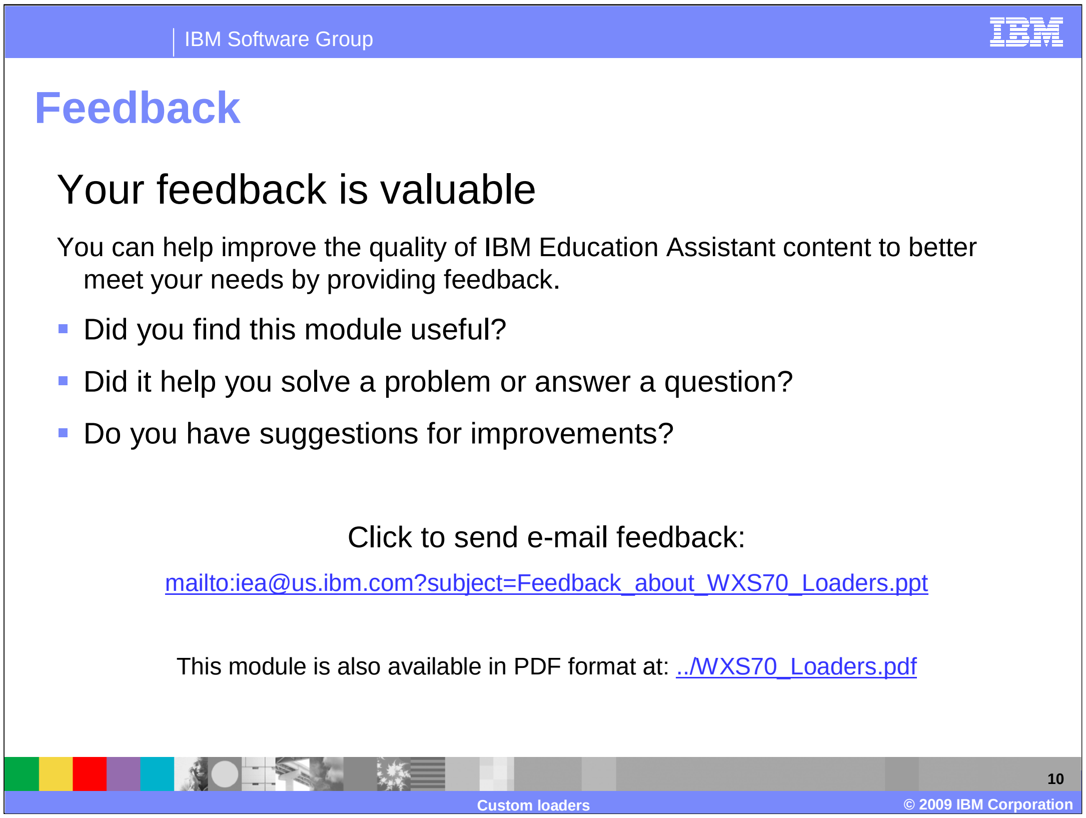


Summary

- Preload data from data store
- Load data on request
- Persist changes to data store
- Runtime properties for flexibility

In summary, WebSphere eXtreme Scale provides a Loader interface, which allows an administrator to associate a BackingMap to a back-end data store. The Loader class can preload data into the BackingMap during map initialization. When a client requests data that is not currently in the BackingMap, the BackingMap uses the Loader to get the data from the data store. The BackingMap also uses the Loader class to persist changes to data in the BackingMap to the data store.

The ability to specify custom runtime properties through the ObjectGrid configuration file makes custom loaders even more powerful and flexible.



IBM Software Group

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

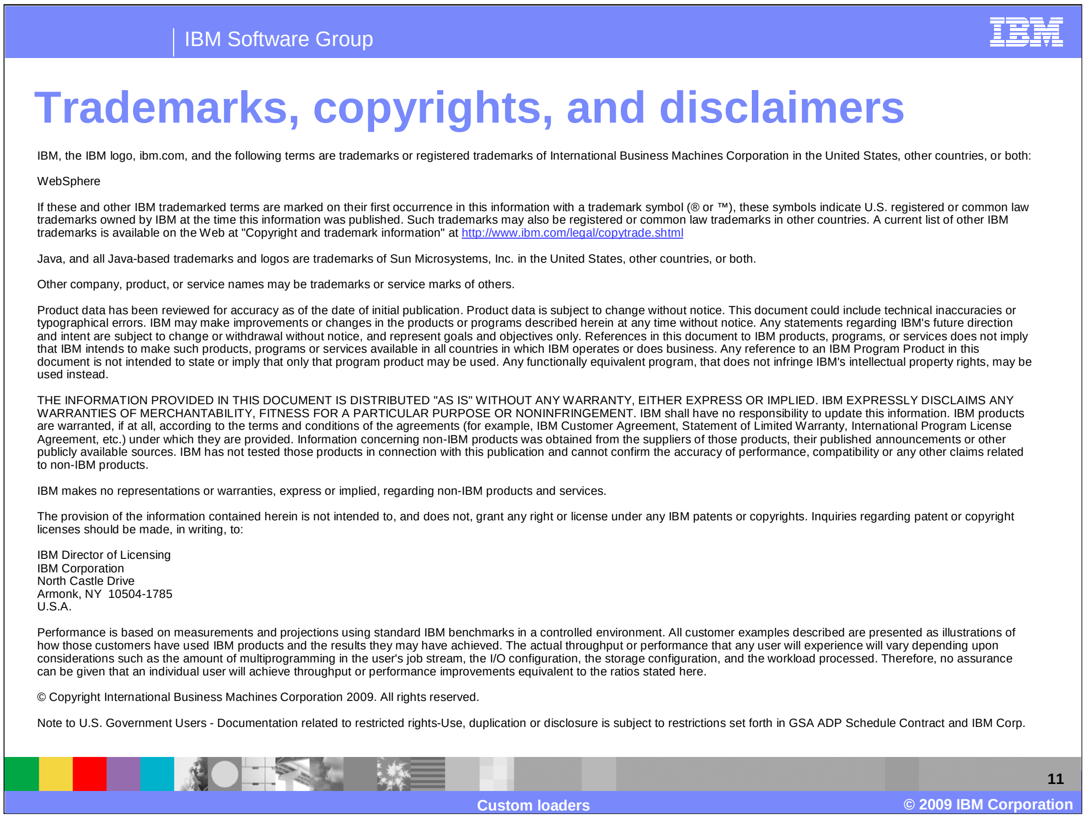
mailto:iea@us.ibm.com?subject=Feedback_about_WXS70_Loaders.ppt

This module is also available in PDF format at: ../WXS70_Loaders.pdf

10

Custom loaders

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers